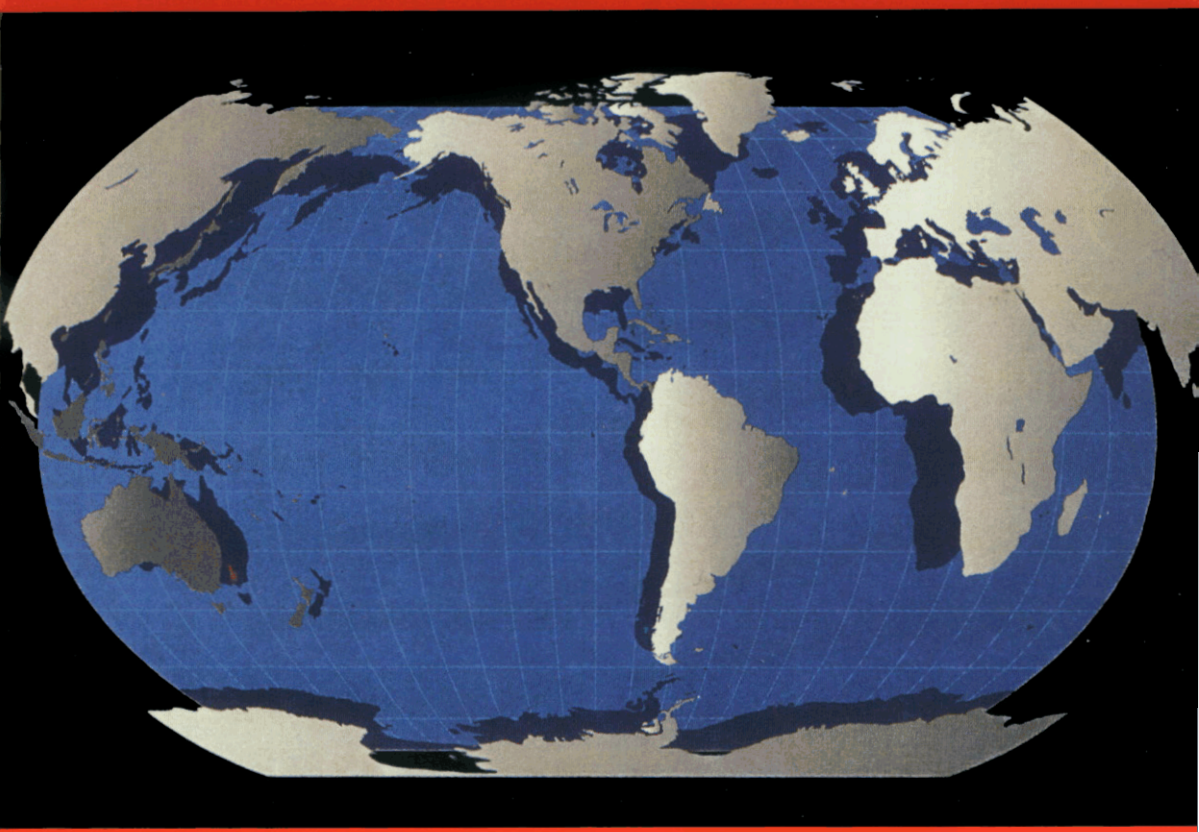
Tercera Edición

Incluye IPng e IP sobre ATM

REDES GLOBALES DE INFORMACION CON INTERNET Y

TCP/IP

Principios básicos, protocolos y arquitectura



PEARSON Prentice Hall ®

DOUGLAS E. COMER

Redes globales de información con Internet y TCP/IP

Principios básicos, protocolos y arquitectura

Redes globales de información con Internet y TCP/IP

Principios básicos, protocolos y arquitectura

Tercera edición

DOUGLAS E. COMER

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

TRADUCCIÓN:

Hugo Alberto Acuña Soto
Traductor Profesional

REVISIÓN TÉCNICA:

Gabriel Guerrero
Doctor en Informática
Universidad de París VI

PEARSON Educación®

México • Argentina • Brasil • Colombia • Costa Rica • Chile • Ecuador
España • Guatemala • Panamá • Perú • Puerto Rico • Uruguay • Venezuela

EDICIÓN EN INGLÉS:

Acquisitions editor: ALAN APT
Production editor: IRWIN ZUCKER
Cover designer: WENDY ALLING JUDY
Buyer: LORI BULWIN
Editorial assistant: SHIRLEY MCGUIRE

COMER: REDES GLOBALES DE INFORMACIÓN CON INTERNET Y TCP/IP
Principios básicos, protocolos y arquitectura, 3a. Ed.

Traducción de la obra en inglés: **Internetworking with TCP/IP, Vol. I: Principles, Protocols, and Architecture**

All rights reserved. Authorized translation from english language edition published by Prentice-Hall, Inc.

Todos los derechos reservados. Traducción autorizada de la edición en inglés publicada por Prentice-Hall, Inc.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage and retrieval system, without permission in writing from the publisher.

Prohibida la reproducción total o parcial de esta obra, por cualquier medio o método sin autorización por escrito del editor.

Derechos reservados © 1996 respecto a la primera edición en español publicada por
PRENTICE-HALL HISPANOAMERICANA, S. A.
Atlacomulco Núm. 500-5° Piso
Col. Industrial Atoto
53519, Naucalpan de Juárez, Edo. de México

ISBN 968-880-541-6

Miembro de la Cámara Nacional de la Industria Editorial, Reg. Núm. 1524

Original English Language Edition Published by Prentice-Hall, Inc.
Copyright © MCMXCV
All rights reserved

ISBN 0-13-216987-8

IMPRESO EN MÉXICO/PRINTED IN MEXICO

A Chris

Contenido

Prólogo xix

Prefacio xxiii

Capítulo 1 Introducción y panorama general 1

1.1 Motivación para trabajar con el enlace de redes 1
1.2 El TCP/IP de Internet 2
1.3 Servicios de Internet 3
1.4 Historia y alcance de Internet 6
1.5 Junta de arquitectura de Internet 8
1.6 Reorganización de IAB 9
1.7 Sociedad Internet 11
1.8 Solicitud de Comentarios de Internet 11
1.9 Protocolos y estandarización de Internet 12
1.10 Crecimiento y tecnologías del futuro 12
1.11 Organización del texto 13
1.12 Resumen 14

Capítulo 2 Reseña de las tecnologías subyacentes de red 17

2.1 Introducción 17
2.2 Dos enfoques de la comunicación por red 18
2.3 Redes de área amplia y local 19
2.4 Tecnología Ethernet 20
2.5 Interconexión de datos distribuida por fibra (FDDI) 33
2.6 Modalidad de transferencia asíncrona 36
2.7 Tecnología ARPANET 37
2.8 Red de la Fundación Nacional de Ciencias 40
2.9 ANSNET 44
2.10 Una red de columna vertebral de área amplia planeada 45

2.11 *Otras tecnologías en las que se ha utilizado el TCP/IP* 45
2.12 *Resumen y conclusión* 48

Capítulo 3 Concepto del enlace de redes y modelo arquitectónico 51

3.1 *Introducción* 51
3.2 *Interconexión de nivel de aplicación* 51
3.3 *Interconexión de nivel de red* 52
3.4 *Propiedades de Internet* 53
3.5 *Arquitectura de Internet* 54
3.6 *Interconexión a través de ruteadores IP* 54
3.7 *El punto de vista del usuario* 55
3.8 *Todas las redes son iguales* 56
3.9 *Las preguntas sin respuesta* 57
3.10 *Resumen* 58

Capítulo 4 Direcciones Internet 61

4.1 *Introducción* 61
4.2 *Identificadores universales* 61
4.3 *Tres tipos primarios de direcciones IP* 62
4.4 *Las direcciones especifican conexiones de red* 63
4.5 *Direcciones de red y de difusión* 63
4.6 *Difusión limitada* 64
4.7 *Interpretación de cero como "esto"* 65
4.8 *Debilidades del direccionamiento de Internet* 65
4.9 *Notación decimal con puntos* 67
4.10 *Dirección loopback* 68
4.11 *Resumen de reglas especiales de direccionamiento* 68
4.12 *Autoridad de direccionamiento Internet* 69
4.13 *Un ejemplo* 69
4.14 *Orden de octetos de red* 71
4.15 *Resumen* 72

Capítulo 5 Transformación de direcciones Internet en direcciones físicas (ARP) 75

5.1 *Introducción* 75
5.2 *El problema de la asociación de direcciones* 75
5.3 *Dos tipos de direcciones físicas* 76
5.4 *Asociación mediante transformación directa* 76
5.5 *Definición mediante enlace dinámico* 77
5.6 *Memoria intermedia para asociación de direcciones* 78
5.7 *Refinamientos ARP* 79

5.8 Relación de ARP con otros protocolos 79
5.9 Implantación de ARP 79
5.10 Encapsulación e identificación de ARP 81
5.11 Formato del protocolo ARP 81
5.12 Resumen 83

Capítulo 6 Determinación en el arranque de una dirección Internet (RARP) 85

6.1 Introducción 85
6.2 Protocolo de asociación de direcciones por réplica (RARP) 86
6.3 Temporización de las transacciones RARP 88
6.4 Servidores RARP primarios y de respaldo 88
6.5 Resumen 89

Capítulo 7 Protocolo Internet: entrega de datagramas sin conexión 91

7.1 Introducción 91
7.2 Una red virtual 91
7.3 Arquitectura y filosofía de Internet 92
7.4 El concepto de entrega no confiable 92
7.5 Sistema de entrega sin conexión 93
7.6 Propósito del protocolo Internet 93
7.7 El datagrama de Internet 94
7.8 Opciones para los datagramas Internet 103
7.9 Resumen 108

Capítulo 8 Protocolo Internet: ruteo de datagramas IP 111

8.1 Introducción 111
8.2 Ruteo en una red de redes 111
8.3 Entrega directa e indirecta 113
8.4 Ruteo IP controlado por tabla 115
8.5 Ruteo con salto al siguiente 115
8.6 Rutas asignadas por omisión 117
8.7 Rutas por anfitrión específico 117
8.8 El algoritmo de ruteo IP 118
8.9 Ruteo con direcciones IP 118
8.10 Manejo de los datagramas entrantes 120
8.11 Establecimiento de tablas de ruteo 121
8.12 Resumen 121

Capítulo 9 Protocolo Internet: mensajes de error y de control (ICMP) 125

9.1 Introducción 125

9.2 *El Protocolo de mensajes de control de Internet* 125
9.3 *Reporte de errores contra corrección de errores* 126
9.4 *Entrega de mensajes ICMP* 127
9.5 *Formato de los mensajes ICMP* 128
9.6 *Prueba de accesabilidad y estado de un destino (Ping)* 129
9.7 *Formato de los mensajes de solicitud de eco y de respuesta* 130
9.8 *Reporte de destinos no accesibles* 130
9.9 *Control de congestionamientos y de flujo de datagramas* 132
9.10 *Formato de disminución de tasa al origen* 132
9.11 *Solicitudes para cambio de ruta desde los ruteadores* 133
9.12 *Detección de rutas circulares o excesivamente largas* 135
9.13 *Reporte de otros problemas* 136
9.14 *Sincronización de relojes y estimación del tiempo de tránsito* 137
9.15 *Solicitud de información y mensajes de respuesta* 138
9.16 *Obtención de una máscara de subred* 138
9.17 *Resumen* 139

Capítulo 10 Extensiones de dirección de subred y superred **141**

10.1 *Introducción* 141
10.2 *Reseña de hechos importantes* 141
10.3 *Minimización de números de red* 142
10.4 *Ruteadores transparentes* 143
10.5 *ARP sustituto (proxy ARP)* 144
10.6 *Direccionamiento de subred* 146
10.7 *Flexibilidad en la asignación de direcciones de subred* 148
10.8 *Implantaciones de subredes con máscaras* 149
10.9 *Representación de máscaras de subred* 150
10.10 *Ruteo con la presencia de subredes* 151
10.11 *El algoritmo de ruteo de subred* 152
10.12 *Un algoritmo unificado de ruteo* 153
10.13 *Mantenimiento de las máscaras de subred* 154
10.14 *Difusión a las subredes* 154
10.15 *Direccionamiento de superred* 155
10.16 *El efecto de trabajar con superredes en el ruteo* 156
10.17 *Resumen* 158

Capítulo 11 Estratificación de protocolos por capas **161**

11.1 *Introducción* 161
11.2 *Necesidad de manejar varios protocolos* 161
11.3 *Las capas conceptuales del software de protocolo* 163
11.4 *Funcionalidad de las capas* 165
11.5 *X.25 y su relación con el modelo ISO* 166
11.6 *Diferencias entre X.25 y la estratificación por capas de Internet* 169

11.7 El principio de la estratificación por capas de protocolos 171
11.8 Estratificación por capas en presencia de una subestructura de red 173
11.9 Dos fronteras importantes en el modelo TCP/IP 175
11.10 La desventaja de la estratificación por capas 176
11.11 La idea básica detrás del multiplexado y el demultiplexado 177
11.12 Resumen 178

Capítulo 12 Protocolo de datagrama de usuario (UDP) 181

12.1 Introducción 181
12.2 Identificación del destino final 181
12.3 Protocolo de datagrama de usuario 182
12.4 Formato de los mensajes UDP 183
12.5 Pseudo-encabezado UDP 184
12.6 Encapsulación de UDP y estratificación por capas de protocolos 185
12.7 Estratificación por capas y cómputo UDP de suma de verificación 187
12.8 Multiplexado, demultiplexado y puertos de UDP 187
12.9 Números de puerto UDP reservados y disponibles 188
12.10 Resumen 190

Capítulo 13 Servicio de transporte de flujo confiable (TCP) 193

13.1 Introducción 193
13.2 Necesidad de la entrega de flujo 193
13.3 Características del servicio de entrega confiable 194
13.4 Proporcionando confiabilidad 195
13.5 La idea detrás de las ventanas deslizables 197
13.6 El protocolo de control de transmisión 199
13.7 Puertos, conexiones y puntos extremos 200
13.8 Aperturas pasivas y activas 202
13.9 Segmentos, flujos y números de secuencia 203
13.10 Tamaño variable de ventana y control de flujo 204
13.11 Formato del segmento TCP 205
13.12 Datos fuera de banda 207
13.13 Opción de tamaño máximo de segmento 207
13.14 Cómputo de suma de verificación TCP 208
13.15 Acuses de recibo y retransmisión 209
13.16 Tiempo límite y retransmisión 210
13.17 Medición precisa de muestras de viaje redondo 212
13.18 Algoritmo de Karn y anulación del temporizador 213
13.19 Respuesta a una variación alta en el retraso 214
13.20 Respuesta al congestionamiento 215
13.21 Establecimiento de una conexión TCP 217
13.22 Números de secuencia inicial 218
13.23 Terminación de una conexión TCP 219

13.24 Restablecimiento de una conexión TCP 220
13.25 Máquina de estado TCP 221
13.26 Forzando la entrega de datos 221
13.27 Números reservados de puerto TCP 223
13.28 Desempeño del TCP 223
13.29 Síndrome de ventana tonta y paquetes pequeños 224
13.30 Prevención del síndrome de ventana tonta 226
13.31 Resumen 229

Capítulo 14 Ruteo: núcleos, pares y algoritmos (GGP) 233

14.1 Introducción 233
14.2 Origen de las tablas de ruteo 234
14.3 Ruteo con información parcial 235
14.4 Arquitectura y núcleos de Internet originales 236
14.5 Ruteadores de núcleo 237
14.6 Más allá de la arquitectura de núcleo, hasta las columnas vertebrales pares 240
14.7 Difusión automática de ruta 242
14.8 Ruteo por vector-distancia (Bellman-Ford) 242
14.9 Protocolo pasarela-a-pasarela (GGP) 244
14.10 Formatos de los mensajes GGP 245
14.11 Ruteo enlace-estado (SPF) 247
14.12 Protocolos SPF 248
14.13 Resumen 249

Capítulo 15 Ruteo: sistemas autónomos (EGP) 251

15.1 Introducción 251
15.2 Agregar complejidad al modelo arquitectónico 251
15.3 Una idea fundamental: saltos adicionales (hops) 252
15.4 Concepto de los sistemas autónomos 254
15.5 Protocolo de pasarela exterior (EGP) 256
15.6 Encabezado de mensaje EGP 257
15.7 Mensajes de adquisición de vecino EGP 258
15.8 Mensajes de accesabilidad de vecino EGP 259
15.9 Mensajes de solicitud de sondeo EGP 260
15.10 Mensajes de actualización de enrutamiento EGP 261
15.11 Medición desde la perspectiva del receptor 263
15.12 La restricción clave de EGP 264
15.13 Problemas técnicos 265
15.14 Descentralización de la arquitectura Internet 266
15.15 Más allá de los sistemas autónomos 266
15.16 Resumen 267

Capítulo 16 Ruteo en un sistema autónomo (RIP, OSPF, HELLO) 269

16.1 Introducción 269
16.2 Rutas interiores dinámicas y estáticas 269
16.3 Protocolo de información de ruteo (RIP) 272
16.4 Protocolo Hello 278
16.5 Combinación de RIP, Hello y EGP 280
16.6 Protocolo de SPF abierto (OSPF) 281
16.7 Ruteo con información parcial 287
16.8 Resumen 288

Capítulo 17 Multidifusión Internet (IGMP) 291

17.1 Introducción 291
17.2 Difusión por hardware 291
17.3 Multidifusión por hardware 292
17.4 Multidifusión IP 293
17.5 Direcciones de multidifusión IP 294
17.6 Transformación de multidifusión IP en multidifusión Ethernet 294
17.7 Extensión de IP para manejar la multidifusión 295
17.8 Protocolo de gestión de grupos de Internet 296
17.9 Implantación IGMP 297
17.10 Transiciones del estado de la membresía de grupo 298
17.11 Formato de los mensajes IGMP 299
17.12 Asignación de direcciones de multidifusión 299
17.13 Difusión de información de ruteo 299
17.14 El programa mrouted 300
17.15 Resumen 302

Capítulo 18 TCP/IP en redes ATM 305

18.1 Introducción 305
18.2 Hardware ATM 306
18.3 Redes ATM grandes 306
18.4 El aspecto lógico de una red ATM 307
18.5 Los dos paradigmas de la conexión ATM 308
18.6 Rutas, circuitos e identificadores 309
18.7 Transporte de celdas ATM 310
18.8 Capas de adaptación ATM 310
18.9 Convergencia, segmentación y reensamblaje de AAL5 313
18.10 Encapsulación de datagramas y tamaño de MTU de IP 314
18.11 Tipos y multiplexión de paquetes 314
18.12 Enlace de direcciones IP en una red ATM 316
18.13 Concepto lógico de subred IP 316

18.14 Gestión de conexiones 317
18.15 Enlace de direcciones dentro de una LIS 318
18.16 Formato de los paquetes ATMARP 318
18.17 Utilización de paquetes ATMARP para determinar una dirección 321
18.18 Obtención de entradas para un servidor de base de datos 322
18.19 Finalización del tiempo de la información ATMARP en un servidor 323
18.20 Finalización del tiempo de la información ATMARP en un anfitrión o en un ruteador 323
18.21 Resumen 324

Capítulo 19 Modelo de interacción cliente-servidor **327**

19.1 Introducción 327
19.2 Modelo cliente-servidor 327
19.3 Un ejemplo simple: servidor de eco UDP 328
19.4 Servicio de fecha y hora 330
19.5 La complejidad de los servidores 331
19.6 Servidor RARP 333
19.7 Alternativas al modelo cliente-servidor 333
19.8 Resumen 334

Capítulo 20 La interfaz socket **337**

20.1 Introducción 337
20.2 El paradigma E/S de UNIX y la E/S de la red 338
20.3 Adición de la red E/S a UNIX 338
20.4 La abstracción de socket 339
20.5 Creación de un socket 340
20.6 Herencia y finalización del socket 341
20.7 Especificación de una dirección local 341
20.8 Conexión de socket con direcciones de destino 342
20.9 Envío de datos a través de un socket 343
20.10 Recepción de datos a través de un socket 345
20.11 Obtención de direcciones socket locales y remotas 347
20.12 Obtención y definición de opciones de socket 347
20.13 Especificación de una longitud de cola para un servidor 348
20.14 Cómo acepta conexiones un servidor 349
20.15 Servidores que manejan varios servicios 350
20.16 Obtención y especificación de nombres de anfitrión 351
20.17 Obtención y especificación del dominio de anfitrión interno 351
20.18 Llamadas de biblioteca de red BSD de UNIX 352
20.19 Rutinas de conversión del orden de red de los octetos 353
20.20 Rutinas de manipulación de direcciones IP 354
20.21 Acceso al sistema de nomenclatura de dominios ANS[5] 355
20.22 Obtención de información sobre anfitriones 357

20.23 Obtención de información sobre redes 357
20.24 Obtención de información sobre protocolos 358
20.25 Obtención de información sobre servicios de red 358
20.26 Ejemplo de un cliente 359
20.27 Ejemplo de un servidor 361
20.28 Resumen 365

Capítulo 21 Arranque y autoconfiguración (BOOTP, DHCP) 367

21.1 Introducción 367
21.2 La necesidad de una alternativa a RARP 368
21.3 Utilización de IP para determinar una dirección IP 369
21.4 Política de retransmisión BOOTP 369
21.5 Formato de los mensajes BOOTP 370
21.6 Procedimiento de arranque de dos pasos 371
21.7 Campo área de vendedor específico 372
21.8 La necesidad de una configuración dinámica 373
21.9 Configuración dinámica de anfitrión 374
21.10 Asignación dinámica de direcciones IP 375
21.11 Obtención de direcciones múltiples 376
21.12 Estados de adquisición de direcciones 376
21.13 Terminación temprana de arrendamiento 378
21.14 Estado de renovación de arrendamiento 378
21.15 Formato de los mensajes DHCP 380
21.16 Opciones y tipos de mensajes DHCP 381
21.17 Opción Overload 381
21.18 DHCP y nombres de dominios[3] 382
21.19 Resumen 382

Capítulo 22 Sistema de nombre de dominio (DNS) 385

22.1 Introducción 385
22.2 Nombres para las máquinas 385
22.3 Espacio de nombre plano 386
22.4 Nombres jerárquicos 387
22.5 Delegar autoridad para los nombres 388
22.6 Autoridad para los subconjuntos de nombres 388
22.7 Nombres de dominio TCP/IP de Internet 389
22.8 Nombres de dominio oficiales y no oficiales de Internet 390
22.9 Cosas por nombrar y sintaxis de los nombres 392
22.10 Asociación de nombres de dominio en direcciones 393
22.11 Resolución de nombres de dominio 395
22.12 Traducción eficiente 396
22.13 Desempeño del cache: la clave de la eficiencia 397
22.14 Formato de los mensajes del servidor de dominios 398

22.15 Formato de nombre comprimido 401
22.16 Abreviatura de nombres de dominio 401
22.17 Asociaciones inversas 403
22.18 Búsquedas de apuntador 403
22.19 Tipos de objetos y contenido del registro de recursos 404
22.20 Obtención de autoridad para un subdominio 405
22.21 Resumen 406

Capítulo 23 Aplicaciones: acceso remoto (TELNET, Rlogin) 409

23.1 Introducción 409
23.2 Computación remota interactiva 409
23.3 Protocolo TELNET 410
23.4 Adaptarse a la heterogeneidad 412
23.5 Transferencia de comandos que controlan el extremo remoto 414
23.6 Forzar al servidor a leer una función de control 416
23.7 Opciones de TELNET 417
23.8 Negociación de opciones de TELNET 418
23.9 Rlogin (BSD de UNIX) 418
23.10 Resumen 419

Capítulo 24 Aplicaciones: transferencia y acceso de archivos (FTP, TFTP, NFS) 423

24.1 Introducción 423
24.2 Acceso y transferencia de archivos 423
24.3 Acceso compartido en línea 424
24.4 Compartir mediante la transferencia de archivos 425
24.5 FTP: el mayor protocolo TCP/IP para transferencia de archivos 426
24.6 Características del FTP 426
24.7 Modelo de proceso FTP 426
24.8 Asignación de números de puerto TCP 428
24.9 El FTP desde el punto de vista del usuario 429
24.10 Ejemplo de una sesión con FTP anónimo 430
24.11 TFTP 431
24.12 NFS 433
24.13 Implantación NFS 434
24.14 Llamada de procedimiento remoto (RPC) 434
24.15 Resumen 435

Capítulo 25 Aplicaciones: correo electrónico (822, SMTP, MIME) 439

25.1 Introducción 439
25.2 Correo electrónico 439

25.3 Nombres y alias de los buzones de correo 441
25.4 Expansión de alias y direccionamiento de correspondencia 441
25.5 Relación entre el enlace de redes y el correo electrónico 442
25.6 Estándares TCP/IP para el servicio de correo electrónico 444
25.7 Direcciones de correo electrónico 445
25.8 Pseudo direcciones de dominio 446
25.9 Protocolo de transferencia de correo simple (SMTP) 447
25.10 La extensión MIME para datos no ASCII 449
25.11 Mensajes MIME multipart 450
25.12 Resumen 452

Capítulo 26 Aplicaciones: manejo de Internet (SNMP, SNMPv2) 455

26.1 Introducción 455
26.2 Nivel de los protocolos de manejo 455
26.3 Modelo arquitectónico 457
26.4 Arquitectura de protocolo 458
26.5 Ejemplos de variables MIB 459
26.6 Estructura de la información de administración 460
26.7 Definiciones formales mediante la ASN.1 461
26.8 Estructura y representación de nombres de objetos MIB 461
26.9 Protocolo de manejo de red simple 466
26.10 Formato de los mensajes SNMP 468
26.11 Ejemplo de un mensaje codificado SNMP 470
26.12 Resumen 471

Capítulo 27 Resumen de las dependencias de protocolos 473

27.1 Introducción 473
27.2 Dependencias de protocolos 473
27.3 Acceso de programas de aplicación 475
27.4 Resumen 476

Capítulo 28 Seguridad de Internet y diseño del muro de seguridad 479

28.1 Introducción 479
28.2 Recursos de protección 480
28.3 Necesidad de una política de información 480
28.4 Comunicación, cooperación y desconfianza mutua 482
28.5 Mecanismos para la seguridad de Internet 482
28.6 Muros de seguridad y acceso a Internet 484
28.7 Conexiones múltiples y vínculos más débiles 485
28.8 Implantación de muro de seguridad y hardware de alta velocidad 486
28.9 Filtros de nivel de paquete 487

28.10 Especificación de seguridad y de filtro de paquetes 488
28.11 Consecuencia del acceso restringido para clientes 489
28.12 Acceso de servicios a través de un muro de seguridad 489
28.13 Detalles de la arquitectura del muro de seguridad 491
28.14 Red Stub 492
28.15 Implantación alternativa de muro de seguridad 492
28.16 Monitoreo y establecimiento de conexión 493
28.17 Resumen 494

Capítulo 29 El futuro del TCP/IP (IPng, IPv6) 497

29.1 Introducción 497
29.2 ¿Por qué cambiar TCP/IP e Internet? 498
29.3 Motivos para el cambio del IPv4 499
29.4 El camino hacia una nueva versión del IP 500
29.5 Nombre del próximo IP 500
29.6 Características del IPv6 501
29.7 Forma general de un datagrama IPv6 502
29.8 Formato del encabezado base del IPv6 502
29.9 Encabezados de extensión del IPv6 505
29.10 Análisis de un datagrama IPv6 506
29.11 Fragmentación y reensamblaje del IPv6 506
29.12 Consecuencia de la fragmentación de extremo a extremo 507
29.13 Ruteamiento de origen del IPv6 508
29.14 Opciones del IPv6 508
29.15 Tamaño del espacio de dirección del IPv6 510
29.16 Notación hexadecimal con dos puntos del IPv6 511
29.17 Tres tipos básicos de dirección IPv6 512
29.18 Dualidad de difusión y multidifusión 513
29.19 Una elección de ingeniería y difusión simulada 513
29.20 Asignación propuesta de espacio de dirección IPv6 513
29.21 Codificación y transición de la dirección IPv4 514
29.22 Proveedores, suscriptores y jerarquía de direcciones 515
29.23 Jerarquía adicional 516
29.24 Resumen 517

Apéndice 1 Guía de RFC 519

Apéndice 2 Glosario de abreviaturas y términos de enlace de redes 565

Bibliografía 599

Índice 607

Prólogo

El libro del profesor Douglas Comer se ha convertido en *el* texto clásico de introducción al TCP/IP. Escribir una introducción al TCP/IP para los no iniciados es una tarea muy difícil. Al combinar la explicación de los principios generales de la comunicación entre computadoras con ejemplos específicos de la serie de protocolos TCP/IP, Douglas Comer nos proporciona un libro accesible y valioso.

Aun cuando esta obra trata específicamente sobre la serie de protocolos TCP/IP, es también un buen libro para aprender acerca de los protocolos de comunicación entre computadoras en general. Los principios en la arquitectura, estratificación por capas, multiplexado, encapsulación, direcciones y transformación de direcciones, ruteo y asignación de nombres, son exactamente los mismos en cualquier conjunto de protocolos, considerando, por supuesto, diferencias en los detalles.

Los protocolos de comunicación de computadoras como los sistemas operativos, *no hacen nada* por sí mismos, sólo están al servicio de los procesos de aplicación. Los procesos son los elementos activos que requieren de la comunicación y, en última instancia, los que envían y reciben los datos transmitidos. Las diversas capas de protocolo son como las diferentes capas en el sistema operativo de una computadora, en especial el sistema de archivos. Entender la arquitectura de un protocolo es como entender la arquitectura de un sistema operativo. En este libro, Douglas Comer eligió un acercamiento que va de lo básico a niveles superiores —comenzando por las redes físicas hasta llegar a niveles de abstracción de las aplicaciones cada vez mayores.

Ya que los procesos de aplicación son los elementos activos que utilizan la comunicación soportada por los protocolos, el TCP/IP es un mecanismo de "comunicación entre procesos" (interprocess communication o IPC por sus siglas en inglés). Aun cuando existen varios experimentos en curso con sistemas operativos respecto a la forma de transferencia de mensajes y el tipo de procedimientos de llamada de la IPC, basados en el IP, el enfoque en este libro se orienta más hacia las aplicaciones tradicionales que emplean datagramas de UDP o formas de conexión lógica de TCP de IPC. Por lo general en un sistema operativo hay un conjunto de funciones proporcionadas por el sistema mismo para los procesos de aplicación. Este sistema, conocido como interfaz, incluye, entre otras cosas llamadas de apertura, lectura, escritura y cierre de archivos. En muchos sistemas hay llamadas de sistema similares para IPC incluyendo la comunicación de redes. Como ejemplo de interfaz, Douglas Comer presenta una panorámica de la interfaz socket.

Una de las ideas clave inherentes al TCP/IP y que define el título de este libro es el "enlace de redes". El poder de un sistema de comunicación está directamente relacionado con el número de entidades en el sistema. La red telefónica es muy útil debido a que (casi) todos los teléfonos están

conectados a una sola red (así aparecen ante el usuario). Los sistemas de comunicación entre computadoras y redes en la actualidad están separados y fragmentados. Dado que cada vez más usuarios y compañías adoptan el TCP/IP como su tecnología de redes y se unen a Internet, este problema comienza a ser menor, pero queda todavía un largo camino por recorrer.

El objetivo de interconectar y enlazar redes a fin de contar con una sola y poderosa red de comunicación entre computadoras ha sido fundamental para el diseño del TCP/IP.

Un aspecto esencial para el enlace de redes es el direccionamiento y contar con un protocolo universal, el Protocolo Internet. Por supuesto, las redes individuales tienen sus propios protocolos, los cuales se utilizan para transferir datagramas IP, y estas direcciones se deben traducir entre la red individual y las direcciones IP. A lo largo de la existencia del TCP/IP, la naturaleza de este tipo de redes ha cambiado, desde los primeros días de ARPANET hasta las redes ATM, desarrolladas recientemente. En un nuevo capítulo de esta edición se analiza el IP en las redes ATM. En este libro, se incluyen ahora los desarrollos más recientes en Dynamic Host Configuration (Configuración Dinámica de Anfitrión o DHCP por sus siglas en inglés), la cual facilita la administración de redes y la instalación de computadoras nuevas.

Para tener un enlace de redes, las redes individuales deben conectarse unas a otras. Los dispositivos de conexión son conocidos como ruteadores. Por otra parte, los ruteadores tienen que contar con algunos procedimientos para enviar la información de una red a otra. La información a transmitir está en forma de datagramas IP y el destino se especifica en una dirección IP, pero el ruteador debe decidir la ruta con base en la dirección IP y lo que sabe de la conectividad de las redes que conforman Internet. Los procedimientos para distribuir información sobre la conectividad actual para los ruteadores se conocen como algoritmos de ruteo, y éstos son hoy en día objeto de gran estudio y desarrollo. En particular, es muy importante el desarrollo reciente de la técnica de Classless InterDomain Routing (CIDR) para reducir la cantidad de intercambios de información para ruteo.

Como todos los sistemas de comunicación, la serie de protocolos del TCP/IP no es un sistema acabado. Esto significa que seguirán dándose cambios en los requerimientos y que habrá nuevas oportunidades. Así pues, este libro es, en cierto sentido, una "instantánea" del TCP/IP. Como Douglas Comer lo señala, hay en esto muchos cabos sueltos. Debido al rápido crecimiento que se ha dado de manera reciente en Internet, existe preocupación acerca de un desbordamiento de las capacidades del protocolo TCP/IP, particularmente en relación con el espacio de direccionamiento. Como respuesta a esto la comunidad de investigadores e ingenieros ha desarrollado una versión para la "próxima generación" del protocolo de Internet conocida como IPng. Muchas de las empresas que se han unido ahora a Internet están preocupadas por la seguridad. En un nuevo capítulo de esta edición se analiza la seguridad y lo que se conoce como muros contra incendios o muros de seguridad (firewalls).

La mayor parte de los capítulos concluye con algunos señalamientos acerca del material "para estudio posterior". Gran parte de este material se refiere a memorandos de las series de notas de RFC. Estas series son resultado de una política para hacer que las ideas que surgen del trabajo y el desarrollo de las especificaciones de los protocolos estén disponibles para la comunidad de investigadores y desarrolladores del TCP/IP. Tal disponibilidad de información básica y detallada sobre los protocolos, y la disponibilidad de sus primeras implantaciones, ha contribuido en gran medida a la extensión actual de su uso. Este compromiso con la documentación pública, a este nivel de detalle, es poco usual en los esfuerzos de investigación y ha aportado beneficios al desarrollo de la comunicación entre computadoras.

Este libro unifica información acerca de varias partes de los protocolos y la arquitectura del TCP/IP, y los hace accesibles. Esta publicación es una aportación muy significativa y relevante a la evolución de la comunicación entre computadoras.

Jon Postel,
Director Asociado del
Networking Information Sciences Institute
Universidad del Sur de California
Enero de 1995

Prefacio

El mundo ha cambiado de manera dramática desde que se publicó la segunda edición de este libro. Es difícil creer que apenas han pasado cuatro años desde entonces. Cuando comenzaba la segunda edición, en el verano de 1990, Internet había crecido hasta llegar cerca de 300,000 computadoras anfitrionas, de 5,000 anfitriones con que contaba cuando se publicó la primera edición de este libro. Al mismo tiempo, estamos maravillados por la forma en que ha crecido y se ha desarrollado a partir de un oscuro proyecto de investigación. Los cínicos predijeron que, de continuar el crecimiento, se *produciría un colapso total para 1993. En lugar de colapsarse,* Internet ha continuado su expansión explosiva; la "gran" Internet de 1990 constituye únicamente el 7% de la Internet actual.

Internet y TCP/IP se han adaptado bien a los cambios. La tecnología básica ha sobrevivido una década de crecimiento exponencial asociado a un incremento en el tráfico. Los protocolos trabajan ahora con nuevas tecnologías de red de alta velocidad y los diseños han soportado aplicaciones que no se hubieran podido imaginar hace una década. Por supuesto, la serie de protocolos en su totalidad no se ha mantenido estática. Se han desarrollado tanto nuevos protocolos como nuevas técnicas para adaptar los protocolos existentes a las nuevas tecnologías de red. Los cambios están documentados en los RFC, los cuales se han incrementado en alrededor de un 50%.

A lo largo de toda la obra aparece información actualizada (incluyendo el uso del término *ruteador de IP* que se ha vuelto muy popular comercialmente, en lugar del tradicional *gateway de IP* que es el término científico tradicional). También, se incluye material nuevo que describe los cambios y los avances técnicos. Asimismo, en el capítulo sobre direccionamiento en redes se describe ahora las superredes y las subredes, y se muestra cómo las dos técnicas son motivadas por el mismo objetivo. El capítulo sobre *bootstrapping* (secuencia de iniciación) expone un avance significativo que eliminará la necesidad de configuración manual de las computadoras anfitrionas y permitirá que una computadora obtenga la dirección IP de manera automática —tal procedimiento se conoce como Dynamic Host Configuration (DHCP). El capítulo sobre TCP incluye una descripción del Silly Window Sindrome (síndrome de las ventanas tontas) y una explicación de la heurística utilizada por TCP para prevenir este problema. El capítulo sobre correo electrónico incluye una descripción de las Multipurpose Internet Mail Extensions (MIME), el cual permite que información no perteneciente a ASCII pueda enviarse en un mensaje de correo electrónico (e-mail) estándar.

Tres nuevos capítulos contienen información detallada acerca de desarrollos significativos. El capítulo 18 explica como el TCP/IP se utiliza en las redes ATM, la organización del hardware de ATM, el propósito de adaptación de los protocolos en capas, la encapsulación IP, la asignación de direcciones, el ruteo y el manejo de los circuitos virtuales. El capítulo ilustra cómo un protocolo sin conexión, como el IP, puede usar la interfaz orientada a la conexión que proporciona ATM. El capítulo 28 cubre un tema que es crucial para muchas organizaciones que están considerando conectarse a la red global de Internet —la seguridad. El capítulo describe el concepto de muro de seguridad y muestra cómo una arquitectura de muro de seguridad puede usarse para proteger las redes y las computadoras en una organización de la entrada de intrusos. En el capítulo también se analizará los principios subyacentes en el diseño del muro de seguridad de dos niveles y se considera los accesos exteriores desde el punto de vista de la seguridad de una computadora. Por último, hay un capítulo nuevo que está dedicado a lo que será el cambio más significativo desde el comienzo del TCP/IP: la inminente adopción del protocolo Internet de la próxima generación (IPng). El capítulo 29 describe el protocolo que ha sido desarrollado por IETF para servir como IPng. Aun cuando ésta no ha sido completamente probada o aprobada como estándar permanente, el nuevo diseño parece haber sido elegido por consenso. El capítulo presenta el diseño propuesto y el esquema de asignación de direcciones.

Esta tercera edición conserva el mismo contenido general y la misma organización de conjunto de la segunda edición. El texto en su totalidad se enfoca al concepto de enlace entre redes en general y de la tecnología de red de redes TCP/IP en particular. El enlace entre redes es una poderosa abstracción que nos permite tratar con la complejidad de múltiples tecnologías de comunicación subyacentes. Esta abstracción oculta los detalles del hardware de red y proporciona un ambiente de comunicación de alto nivel. En la obra se revisa la arquitectura de interconexión de redes y los principales protocolos subyacentes, que hacen que una red interconectada funcione como un solo sistema de comunicación unificado. También se muestra cómo un sistema de comunicación entre redes puede usarse para la computación distribuida.

Luego de leer este libro, entenderá cómo es posible interconectar múltiples redes físicas en un solo sistema coordinado, de qué manera operan los protocolos entre redes en el ambiente y cómo los programas de aplicación emplean el sistema resultante. Con un ejemplo específico, podrá aprender los detalles del TCP/IP global de Internet, incluyendo la arquitectura de su sistema de ruteo y los protocolos de aplicación que soporta. Además, comprenderá algunas de las limitaciones de la red de redes.

Diseñado como libro de texto y referencia profesional, la obra está escrita para niveles avanzados de estudiantes no graduados o graduados. Para los profesionales, el libro proporciona una introducción completa a la tecnología del TCP/IP y a la arquitectura de Internet. Aun cuando no se intenta reemplazar los estándares de los protocolos, el libro es un excelente punto de partida para aprender acerca del enlace de redes, dado que proporciona una panorámica completa que hace énfasis en los principios básicos. Más aún ofrece una perspectiva al lector que puede ser muy difícil de obtener desde los documentos de los protocolos por separado.

Si se emplea en un salón de clase, el texto puede proporcionar material más que suficiente para un curso de redes semestral, tanto para estudiantes graduados como no graduados. Tal curso puede extenderse a una secuencia de dos semestres si se acompaña con proyectos de programación y lecturas relacionadas. Para cursos de estudiantes no graduados, muchos de los detalles son innecesarios. Los estudiantes deberán asirse a los conceptos básicos descritos en el texto y ser capaces de describirlos o emplearlos. Los estudiantes graduados tendrán que utilizar el material presentado aquí como una base para posteriores investigaciones. Deberán comprender los detalles con la suficiente precisión para responder a los ejercicios o resolver problemas que impliquen una mayor sutileza o requieran de

investigaciones más amplias. Muchos de los ejercicios sugeridos son muy sutiles; con frecuencia, resolverlos requerirá que el estudiante lea los estándares de los protocolos y aplique su energía creativa a comprender las consecuencias.

En todos los niveles, la experiencia práctica afirmará los conceptos y ayudará al estudiante a obtener una mayor intuición. Por lo tanto, hago una exhortación a los instructores para que implanten proyectos que obliguen a los estudiantes a valerse de los servicios y protocolos de Internet. El proyecto semestral en el curso Internetworking en Purdue requiere que los estudiantes construyan un ruteador IP. Nosotros les proporcionamos el hardware y el código fuente para un sistema operativo, incluyendo controladores de dispositivos para interfaz de red; los estudiantes trabajan para construir un ruteador que interconecte 3 redes con diferentes MTU. El curso es muy riguroso, los estudiantes trabajan en equipo, y los resultados han sido inesperados (ya muchas empresas han contratado a graduados de nuestro curso). Aun cuando la experimentación es segura, dado que la instrucción en el laboratorio de red está aislada de la producción de la infraestructura de computación, hemos encontrado que los estudiantes muestran un gran entusiasmo y se benefician mucho cuando tienen acceso a un TCP/IP funcional entre redes.

El libro está organizado en cuatro partes principales. Los capítulos 1 y 2 forman una introducción que proporciona una panorámica y en ellos se analiza las tecnologías de red existentes. En particular, el capítulo 2 revisa el hardware físico de red. La intención es proporcionar una intuición básica acerca de lo que es posible sin perder un tiempo desproporcionado en los detalles del hardware. De los capítulos 3 a 13, se describe el TCP/IP de Internet desde el punto de vista de un solo anfitrión, mostrando los protocolos que contiene un anfitrión y la forma en que éstos operan. Tales capítulos cubren las bases del direccionamiento de Internet y de ruteo, así como la noción de protocolo de estratificación por capas. En los capítulos 14 a 18 y 28 se describe la arquitectura de una red de redes considerada globalmente. En ellos se explora la arquitectura del ruteo y se utilizan los protocolos de los ruteadores para intercambiar información de ruteo. Por último, en los capítulos 19 al 27 se trata la aplicación de los servicios de nivel disponibles en Internet. En ellos se presenta un modelo de interacción cliente-servidor y se proporcionan varios ejemplos de software cliente y servidor.

Los capítulos han sido organizados en un orden ascendente. Esto es, se comienza con una panorámica del hardware con la estructuración de nuevas funciones hasta llegar a los niveles superiores. Este tipo de presentación pondrá de manifiesto ante cualquiera la forma en que se ha desarrollado el software de Internet, dado que aquí se sigue el mismo modelo en cuanto a usos e implantación. El concepto de estratificación por capas no aparece sino hasta el capítulo 11. El análisis de la estratificación por capas enfatiza la distinción entre las capas conceptuales de funcionalidad y la realidad del software de protocolo por capas, en el cual aparecen múltiples objetos en cada capa.

Se requiere de pocos antecedentes para comprender el material aquí presentado. El lector deberá contar con conocimientos básicos de sistemas de computadoras y estar familiarizado con estructuras de datos como pilas, colas y árboles. El lector necesita una intuición básica sobre la organización del software de computadora dentro de un sistema operativo que soporta programación concurrente y programas de aplicación que utilizan llamadas para cumplir con los procesos de computación. No necesita contar con grandes conocimientos de matemáticas y tampoco conocer la teoría de la información o los teoremas de la comunicación de información; el libro describe la red física como una caja negra a partir de la cual se puede estructurar un enlace de redes. Se establecen principios de diseño en inglés y se discuten las motivaciones y las consecuencias.

Estoy agradecido con todas las personas que contribuyeron en la elaboración de esta nueva edición del libro. John Lin colaboró ampliamente en esta edición, al incluir la clasificación de los RFC. Ralph Droms revisó el capítulo sobre la secuencia de iniciación, Sandeep Kumar, Steve Lodin y Christoph Schuba, del proyecto de seguridad COAST de Purdue, comentaron el capítulo que trata sobre seguridad. Agradezco en particular a mi esposa, Chris, por haber editado cuidadosamente e introducido muchas mejoras en la redacción de este libro.

1

Introducción y panorama general

1.1 Motivación para trabajar con el enlace de redes

La comunicación de datos se ha convertido en una parte fundamental de la computación. Las redes globales reúnen datos sobre temas diversos, como las condiciones atmosféricas, la producción de cosechas y el tráfico aéreo. Algunos grupos establecen listas de correo electrónico para poder compartir información de interés común. Las personas que tienen pasatiempos intercambian programas para sus computadoras personales. En el mundo científico, las redes de datos son esenciales pues permiten a los científicos enviar programas y datos hacia supercomputadoras remotas para su procesamiento, recuperar los resultados e intercambiar información con sus colegas.

Por desgracia, la mayor parte de las redes son entidades independientes, establecidas para satisfacer las necesidades de un solo grupo. Los usuarios escogen una tecnología de hardware apropiada a sus problemas de comunicación. De manera más importante, es imposible construir una red universal desde una sola tecnología de hardware, debido a que ninguna red satisface todas las necesidades de uso. Algunos usuarios necesitan una red de alta velocidad para conectar máquinas, pero dichas redes no se pueden expandir para abarcar grandes distancias. Otros establecen una red de menor velocidad que conecta máquinas que se encuentran a miles de kilómetros de distancia.

Durante los pasados 15 años, ha evolucionado una nueva tecnología que hace posible interconectar muchas redes físicas diferentes y hacerlas funcionar como una unidad coordinada. Esta tecnología, llamada *internetworking*, unifica diferentes tecnologías de hardware subyacentes al proporcionar un conjunto de normas de comunicación y una forma de interconectar redes heterogéneas. La tecnología de red de redes oculta los detalles del hardware de red y permite que las computadoras se comuniquen en forma independiente de sus conexiones físicas de red.

La tecnología de red de redes que se describe en este libro es un ejemplo de la *interconexión del sistema abierto*. Se llama *sistema abierto* porque, a diferencia de los sistemas privados de comunicación disponibles por medio de vendedores particulares, las especificaciones están disponibles públicamente. Por lo tanto, cualquier persona puede desarrollar el software necesario para comunicarse a través de una red de redes. Algo muy importante es que toda la tecnología ha sido diseñada para permitir la comunicación entre máquinas que tengan arquitecturas diferentes de hardware, para utilizar cualquier hardware de red de paquetes conmutados y para incorporar muchos sistemas operativos de computadoras.

Para apreciar la tecnología de red de redes, piense en cómo afecta a un grupo de profesionistas. Considere, por ejemplo, el efecto de interconectar las computadoras utilizadas por científicos. Cualquier científico puede intercambiar los datos que resulten de un experimento con cualquier otro científico. Los centros nacionales pueden recolectar datos de los fenómenos naturales y poner dichos datos a disposición de los científicos. Los servicios de computación y los programas que estén disponibles en un sitio pueden utilizarse por otros científicos en otros sitios. Como resultado, la velocidad a la que se lleva a cabo la investigación científica aumenta; los cambios son dramáticos.

1.2 El TCP/IP de Internet

Las agencias gubernamentales de los Estados Unidos se han dado cuenta de la importancia y el potencial de la tecnología de red de redes desde hace muchos años y han proporcionado fondos para la investigación, con lo cual se ha hecho posible una red de redes global. En este libro se tratan los principios e ideas que subyacen en la tecnología de red de redes, producto de la investigación realizada con fondos de la *Agencia de Proyectos de Investigación Avanzada* (*ARPA*, por sus siglas en inglés).[1] La tecnología ARPA incluye un grupo de estándares de red que especifican los detalles de cómo se comunican las computadoras, así como un grupo de reglas para interconectar redes y para rutear el tráfico. Conocido de manera oficial como el grupo de protocolos Internet TCP/IP, pero llamado más comúnmente *TCP/IP* (siglas provenientes de sus dos principales estándares), éste puede utilizarse para comunicarse a través de cualquier grupo de redes interconectadas. Por ejemplo, algunas empresas utilizan el TCP/IP para interconectar todas las redes dentro de la corporación, aun cuando las empresas no tengan una conexión hacia redes externas. Otros grupos utilizan el TCP/IP para comunicarse entre sitios geográficamente alejados uno de otro.

Aunque la tecnología TCP/IP es significativa por sí misma, es especialmente interesante debido a que su viabilidad ha sido demostrada a gran escala. Ésta forma la tecnología base para una red de redes global que conecta hogares, campus universitarios y otras escuelas, corporaciones y laboratorios gubernamentales en 61 países. En Estados Unidos, la *Fundación Nacional de Ciencias (NSF)*, en el *Departamento de Energía (DOE)*, el *Departamento de Defensa (DOD)*, la *Agencia de Servicios Humanos y de Salud (HHS)* y la *Administración Nacional Aeronáutica y del Espacio (NASA)* han contribuido con los fondos para Internet, y utilizan el TCP/IP para conectar muchos de sus centros de investigación. Conocido como *Internet (ARPA/NSF)*, *Internet TCP/IP*, *Internet global* o tan sólo

[1] ARPA era conocida como la *Agencia de Proyectos Avanzados de Investigación de la Defensa* por muchos años durante la década de los ochenta.

Internet,[2] la red de redes resultante permite que los investigadores en las instituciones conectadas compartan información con sus colegas alrededor del mundo, tan fácilmente como si compartieran información con investigadores en un cuarto contiguo. Por su gran éxito, Internet demuestra la viabilidad de la tecnología TCP/IP y muestra cómo puede incorporar una amplia variedad de tecnologías subyacentes de red.

La mayor parte del material en este libro se aplica a cualquier red de redes que utilice el TCP/IP, pero algunos capítulos se refieren de manera específica a la Internet global. Los lectores que sólo estén interesados en la tecnología, deben tener cuidado en distinguir la diferencia entre la arquitectura de Internet por sí misma y las redes de redes TCP/IP generales que puedan existir. Sin embargo, sería un error ignorar por completo secciones del texto que describen a la Internet global muchas redes corporativas ya son más complejas que el Intermet global de hace diez años, y muchos de los problemas que enfrentan ya han sido resueltos en el Internet global.

1.3 Servicios de Internet

Una persona no puede apreciar los detalles técnicos subyacentes del TCP/IP sin entender los servicios que proporciona. En esta sección, se revisan de manera breve los servicios de una red de redes, resaltando los servicios que la mayoría de los usuarios utiliza, y se deja para los capítulos posteriores el análisis de cómo se conectan las computadoras a una red de redes TCP/IP y cómo se implementa su funcionalidad.

Casi todo el análisis de los servicios se enfocará en estándares llamados *protocolos*. Protocolos como el TCP y el IP proporcionan las reglas para la comunicación. Contienen los detalles referentes a los formatos de los mensajes, describen cómo responde una computadora cuando llega un mensaje y especifican de qué manera una computadora maneja un error u otras condiciones anormales. Un aspecto importante es que permite reflexionar sobre la comunicación por computadora de manera independiente de cualquier hardware de red de cualquier marca. En cierto sentido, los protocolos son para las comunicaciones lo que los algoritmos para la computación. Un algoritmo permite especificar o entender un cómputo aunque no se conozcan los detalles de un juego de instrucciones de CPU. De manera similar, un protocolo de comunicaciones permite especificar o entender la comunicación de datos sin depender de un conocimiento detallado de una marca en particular de hardware de red.

El hacer a un lado los detalles de bajo nivel de la comunicación nos ayuda a mejorar la productividad de muchas maneras. Primero, debido a que los programadores tienen que manejar abstracciones de protocolos de un nivel más elevado, no necesitan aprender o recordar tantos detalles sobre una configuración de hardware en particular. Pueden crear con rapidez nuevos programas. Segundo, como los programas hechos por medio de abstracciones de un nivel más elevado no se encuentran restringidos a una sola arquitectura de máquina o a un solo tipo de hardware de red, no se necesitan cambiar cuando se reconfiguran las máquinas o las redes. Tercero, puesto que los programas de aplicación hechos mediante protocolos de un nivel más elevado son independientes del hardware subyacente, pueden proporcionar comunicación directa entre un par arbitrario de máquinas. Los programadores no necesitan hacer versiones especiales de software de aplicación para mover y traducir datos entre cada par de máquinas posibles

[2] Seguiremos la convención usual escribiendo en mayúsculas la letra i de *Internet* al referirnos específicamente al Internet global, y usar minúsculas al referirnos a los internets privados que utilizan TCP/IP

Veremos que todos los servicios de red se encuentran descritos por protocolos. En las siguientes secciones, nos referimos a protocolos utilizados para especificar servicios de nivel de aplicación así como los utilizados para definir servicios a nivel de red. En los capítulos posteriores, se explica con mayor detalle cada uno de estos protocolos.

1.3.1 Servicios de Internet a nivel de aplicación

Desde el punto de vista de un usuario, una red de redes TCP/IP aparece como un grupo de programas de aplicación que utilizan la red para llevar a cabo tareas útiles de comunicación. Utilizamos el término *interoperabilidad* para referirnos a la habilidad que tienen diversos sistemas de computación para cooperar en la resolución de problemas computacionales. Los programas de aplicación de Internet muestran un alto grado de interoperabilidad. La mayoría de los usuarios que accesan a Internet lo hacen al correr programas de aplicación sin entender la tecnología TCP/IP, la estructura de la red de redes subyacente o incluso sin entender el camino que siguen los datos hacia su destino; los usuarios confían en los programas de aplicación y en el software subyacente de red para manejar esos detalles. Sólo los programadores que crean los programas de aplicación de red necesitan ver a la red de redes como una red, así como entender parte de la tecnología.

Los servicios de aplicación de Internet más populares y difundidos incluyen:

- *Correo electrónico.* El correo electrónico permite que un usuario componga memorandos y los envíe a individuos o grupos. Otra parte de la aplicación de correo permite que un usuario lea los memorandos que ha recibido. El correo electrónico ha sido tan exitoso que muchos usuarios de Internet dependen de él para su correspondencia normal de negocios. Aunque existen muchos sistemas de correo electrónico, al utilizar el TCP/IP se logra que la entrega sea más confiable debido a que no se basa en computadoras intermedias para distribuir los mensajes de correo. Un sistema de entrega de correo TCP/IP opera al hacer que la máquina del transmisor contacte directamente la máquina del receptor. Por lo tanto, el transmisor sabe que, una vez que el mensaje salga de su máquina local, se habrá recibido de manera exitosa en el sitio de destino.
- *Transferencia de archivos.* Aunque los usuarios algunas veces transfieren archivos por medio del correo electrónico, el correo está diseñado principalmente para mensajes cortos de texto. Los protocolos TCP/IP incluyen un programa de aplicación para transferencia de archivos, el cual permite que los usuarios envíen o reciban archivos arbitrariamente grandes de programas o de datos. Por ejemplo, al utilizar el programa de transferencia de archivos, se puede copiar de una máquina a otra una gran base de datos que contenga imágenes de satélite, un programa escrito en Pascal o C++, o un diccionario del idioma inglés. El sistema proporciona una manera de verificar que los usuarios cuenten con autorización o, incluso, de impedir el acceso. Como el correo, la transferencia de archivos a través de una red de redes TCP/IP es confiable debido a que las dos máquinas comprendidas se comunican de manera directa, sin tener que confiar en máquinas intermedias para hacer copias del archivo a lo largo del camino.
- *Acceso remoto.* El acceso remoto permite que un usuario que esté frente a una computadora se conecte a una máquina remota y establezca una sesión interactiva. El acceso remoto hace aparecer una ventana en la pantalla del usuario, la cual se conecta directamente con la máquina remota al enviar cada golpe de tecla desde el teclado del usuario a una máquina remota y muestra en la ventana del usuario cada caracter que la computadora remota genere. Cuando termina la sesión de acceso remoto, la aplicación regresa al usuario a su sistema local.

Nos referiremos a estas y a otras aplicaciones en capítulos posteriores para examinarlas con mayor detalle. Veremos cómo utilizan exactamente los protocolos subyacentes TCP/IP y por qué tener estándares para protocolos de aplicación ayuda a garantizar que sean difundidos de manera amplia.

1.3.2 Servicios de Internet a nivel de red

Un programador que crea programas de aplicación que utilizan protocolos TCP/IP tiene una visión totalmente diferente de una red de redes, con respecto a la visión que tiene un usuario que únicamente ejecuta aplicaciones como el correo electrónico. En el nivel de red, una red de redes proporciona dos grandes tipos de servicios que todos los programas de aplicación utilizan. Aunque no es importante en este momento entender los detalles de estos servicios, no se pueden omitir del panorama general del TCP/IP:

- *Servicio sin conexión de entrega de paquetes.* Este servicio, tratado en detalle más adelante en el texto, forma la base de todos los otros servicios de red de redes. La entrega sin conexión es una abstracción del servicio que la mayoría de las redes de conmutación de paquetes ofrece. Simplemente significa que una red de redes TCP/IP rutea mensajes pequeños de una máquina a otra, basándose en la información de dirección que contiene cada mensaje. Debido a que el servicio sin conexión rutea cada paquete por separado, no garantiza una entrega confiable y en orden. Como por lo general se introduce directamente en el hardware subyacente, el servicio sin conexión es muy eficiente. Algo muy importante es que tener una entrega de paquetes sin conexión como la base de todos los servicios de red de redes, hace que los protocolos TCP/IP sean adaptables a un amplio rango de hardware de red.
- *Servicio de transporte de flujo confiable.* La mayor parte de las aplicaciones necesitan mucho más que sólo la entrega de paquetes, debido a que requieren que el software de comunicaciones se recupere de manera automática de los errores de transmisión, paquetes perdidos o fallas de conmutadores intermedios a lo largo del camino entre el transmisor y el receptor. El servicio de transporte confiable resuelve dichos problemas. Permite que una aplicación en una computadora establezca una "conexión" con una aplicación en otra computadora, para después enviar un gran volumen de datos a través de la conexión como si ésta fuera permanente y directa del hardware. Debajo de todo esto, por supuesto, los protocolos de comunicación dividen el flujo de datos en pequeños mensajes y los envían, uno tras otro, esperando que el receptor proporcione un acuse de recibo de la recepción.

Muchas redes proporcionan servicios básicos similares a los que se indican arriba, así que usted se podrá preguntar qué es lo que distingue a los servicios TCP/IP de los otros. Las principales características distintivas son:

- *Independencia de la tecnología de red.* Ya que el TCP/IP está basado en una tecnología convencional de conmutación de paquetes, es independiente de cualquier marca de hardware en particular. La Internet global incluye una variedad de tecnologías de red que van de redes diseñadas para operar dentro de un solo edificio a las diseñadas para abarcar grandes distancias. Los protocolos TCP/IP definen la unidad de transmisión de datos, llamada *datagrama*, y especifican cómo transmitir los datagramas en una red en particular.

- *Interconexión universal.* Una red de redes TCP/IP permite que se comunique cualquier par de computadoras conectadas a ella. Cada computadora tiene asignada una *dirección* reconocida de manera universal dentro de la red de redes. Cada datagrama lleva en su interior las direcciones de su fuente y su destino. Las computadoras intermedias de conmutación utilizan la dirección de destino para tomar decisiones de ruteo.
- *Acuses de recibo punto-a-punto.* Los protocolos TCP/IP de una red de redes proporcionan acuses de recibo entre la fuente y el último destino en vez de proporcionarlos entre máquinas sucesivas a lo largo del camino, aun cuando las dos máquinas no estén conectadas a la misma red física.
- *Estándares de protocolo de aplicación.* Además de los servicios básicos de nivel de transporte (como las conexiones de flujo confiable), los protocolos TCP/IP incluyen estándares para muchas aplicaciones comunes, incluyendo correo electrónico, transferencia de archivos y acceso remoto. Por lo tanto, cuando se diseñan programas de aplicación que utilizan el TCP/IP, los programadores a menudo se encuentran con que el software ya existente proporciona los servicios de comunicación que necesitan.

En los capítulos posteriores, se tratarán los detalles de los servicios proporcionados a los programadores, así como muchos de los estándares de protocolos de aplicación.

1.4 Historia y alcance de Internet

Parte de lo que hace tan interesante a la tecnología TCP/IP es su adopción casi universal, así como el tamaño y el índice de crecimiento de la Internet global. ARPA comenzó a trabajar con una tecnología de red de redes a mediados de los años setenta; su arquitectura y protocolos tomaron su forma actual entre 1977 y 1979. En ese tiempo, ARPA era conocida como la principal agencia en proporcionar fondos para la investigación de redes de paquetes conmutados y fue pionera de muchas ideas sobre la conmutación de paquetes con su bien conocida *ARPANET*. ARPANET utilizaba interconexión convencional de línea rentada punto-a-punto, pero ARPA también ofreció fondos para la exploración de conmutación de paquetes a través de redes de radio y mediante canales de comunicación por satélite. De hecho, la diversidad creciente de tecnologías de hardware de red obligó a ARPA a estudiar la interconexión de redes y alentó el enlace de redes.

La disponibilidad de ARPA en cuanto a fondos para la investigación, atrajo la atención y la imaginación de muchos grupos de investigación, en especial de los investigadores que ya tenían experiencia previa utilizando conmutación de paquetes en ARPANET. ARPA llevó a cabo reuniones informales de investigadores para compartir ideas y discutir los resultados de los experimentos. En 1979, había tantos investigadores involucrados en los esfuerzos del TCP/IP, que ARPA formó un comité informal para coordinar y guiar el diseño de los protocolos y la arquitectura del Internet que surgía. Llamada Junta de Control y Configuración de Internet (ICCB), el grupo se reunía con regularidad hasta 1983, año en que fue reorganizado.

La Internet global se inició alrededor de 1980 cuando ARPA comenzó a convertir las máquinas conectadas a sus redes de investigación en máquinas con el nuevo protocolo TCP/IP. ARPANET, una vez en su lugar, se convirtió rápidamente en la columna vertebral del nuevo Internet, y fue utilizada para realizar muchos de los primeros experimentos con el TCP/IP. La transición hacia la tecnología

Internet se completó en enero de 1983, cuando la Oficina del Secretario de Defensa ordenó que todas las computadoras conectadas a redes de largo alcance utilizaran el TCP/IP. Al mismo tiempo, la *Agencia de Comunicación de la Defensa* (DCA), dividió ARPANET en dos redes separadas, una para la investigación futura y otra para la comunicación militar. La parte de investigación conservó el nombre de ARPANET; la parte militar, que era un poco más grande, se conoció como *red militar MILNET.*

Para alentar a los investigadores universitarios a que adoptaran y utilizaran los nuevos protocolos, ARPA puso a su disposición una implementación de bajo costo. En ese tiempo, la mayor parte de los departamentos universitarios de ciencias de la computación utilizaban una versión del sistema operativo UNIX, disponible en *Distribución Berkeley de Software* de la Universidad de California, en general conocido como *UNIX Berkeley* o *UNIX BSD.* Al proporcionar fondos a Bolt Beranek de Newman, Inc. (BBN), para implementar sus protocolos TCP/IP en la utilización de UNIX y al proporcionar fondos a Berkeley para integrar los protocolos a su sistema de distribución de software, ARPA fue capaz de llegar a más del 90% de los departamentos universitarios de ciencias de la computación. El nuevo software de protocolo llegó en un momento particularmente significativo pues muchos departamentos estaban adquiriendo una o dos computadoras adicionales y sólo las conectaban mediante redes de área local. Los departamentos necesitaban protocolos de comunicación y no había otros generalmente disponibles.

La distribución Berkeley de software se volvió popular ya que ofrecía mucho más que protocolos básicos TCP/IP. Además de los programas normales de aplicación TCP/IP, Berkeley ofrecía un grupo de utilidades para servicios de red que se parecían a los servicios de UNIX utilizados en una sola máquina. La principal ventaja de las utilidades Berkeley reside en su parecido con el UNIX normal. Por ejemplo, un usuario experimentado de UNIX puede aprender rápidamente a utilizar la utilidad de copia remota de archivos de Berkeley (*rcp*), debido a que se comporta de la misma manera que la utilidad UNIX de copia de archivo, a excepción de que permite que los usuarios copien archivos hacia y desde máquinas remotas.

Además de un grupo de programas de utilidades, UNIX Berkeley proporcionó una nueva abstracción de sistema operativo conocida como *socket*, la cual permite que programas de aplicación accesen a protocolos de comunicación. Como generalización del mecanismo UNIX para I/O, el socket tiene opciones para muchos tipos de protocolo de red además del TCP/IP. Su diseño ha sido motivo de debate desde su introducción, y muchos investigadores de sistemas operativos han propuesto alternativas. Sin embargo, independientemente de sus méritos generales, la introducción de la abstracción socket fue importante ya que permitió a los programadores utilizar protocolos TCP/IP sin mucho esfuerzo. Por lo tanto, alentó a los investigadores a experimentar con el TCP/IP.

El éxito de la tecnología TCP/IP y de Internet entre los investigadores de ciencias de la computación guió a que otros grupos la adoptaran. Dándose cuenta de que la comunicación por red pronto sería una parte crucial de la investigación científica, la Fundación Nacional de Ciencias tomó un papel activo al expandir el Internet TCP/IP para llegar a la mayor parte posible de científicos. Iniciando en 1985, se comenzó un programa para establecer redes de acceso distribuidas alrededor de sus seis centros con supercomputadoras. En 1986 se aumentaron los esfuerzos para el enlace de redes al proporcionar fondos para una nueva red de columna vertebral de área amplia, llamada *NSFNET*,[1] que eventualmente alcanzó todos los centros con supercomputadoras y los unió a ARPA-

[1] El término *NSFNET* se utiliza a veces en forma amplia para hacer alusión a todas las actividades de red de las NSF fundadas, pero nosotros lo usaremos para referirnos a la red de columna vertebral. En el próximo capítulo, aportaremos más detalles sobre la tecnología.

NET. Por último, en 1986, la NSF proporcionó fondos para muchas redes regionales, cada una de las cuales conecta en la actualidad importantes instituciones científicas de investigación en cierta área. Todas las redes con fondos de la NSF utilizan los protocolos TCP/IP y todas forman parte de la Internet global.

A siete años de su concepción, Internet había crecido hasta abarcar cientos de redes individuales localizadas en los Estados Unidos y en Europa. Conectaba casi 20,000 computadoras en universidades, así como a centros de investigación privados y gubernamentales. El tamaño y la utilización de Internet ha seguido creciendo mucho más rápido de lo esperado. A finales de 1987, se estimó que el crecimiento había alcanzado un 15% mensual. En 1994, la Internet global incorporaba más de 3 millones de computadoras en 61 países.

La adopción de los protocolos TCP/IP y el crecimiento de Internet no se ha limitado a proyectos con fondos del gobierno. Grandes corporaciones computacionales se conectaron a Internet, así como muchas otras grandes corporaciones, incluyendo: compañías petroleras, automovilísticas, empresas electrónicas, compañías farmaceúticas y de telecomunicaciones. Las compañías medianas y pequeñas se comenzaron a conectar en los años noventa. Además, muchas compañías han utilizado los protocolos TCP/IP en sus redes corporativas, aunque no han optado por ser parte de la Internet global.

La rápida expansión ha presentado problemas de escala no contemplados en el diseño original y motivado a los investigadores a encontrar técnicas para manejar grandes recursos distribuidos. Por ejemplo, en el diseño original, los nombres y direcciones de todas las computadoras conectadas a Internet se guardaban en un solo archivo que se editaba a mano y luego se distribuía a cada sitio en Internet. A mediados de los ochenta, fue obvio que una base central no sería suficiente. Primero, las solicitudes de actualización del archivo pronto excederían la capacidad de procesamiento del personal disponible. Segundo, aunque existiera un archivo central apropiado, la capacidad de la red era insuficiente para permitir la distribución frecuente a cada sitio o el acceso por línea de cada sitio.

Se desarrollaron nuevos protocolos y se estableció un nuevo sistema de nombres en la Internet global, que permite que cualquier usuario deduzca de manera automática el nombre de una máquina remota. Conocido como *Sistema de Nomenclatura de Dominios*, el mecanismo se apoya en máquinas llamadas *servidores* para responder a solicitudes de nombres. Ninguna máquina en sí misma contiene toda la base de datos de nombres de dominio. Los datos por el contrario, se encuentran distribuidos entre un grupo de máquinas que utiliza protocolos TCP/IP para comunicarse entre ellas cuando responden a una solicitud de búsqueda.

1.5 Junta de arquitectura de Internet

Debido a que el grupo de protocolos TCP/IP para red de redes no surgió de una marca específica o de una sociedad profesional reconocida, es natural preguntar ¿quién establece la dirección técnica y quién decide cuándo los protocolos se convierten en estándares? La respuesta es: un grupo conocido como la *Internet Architecture Board* (Junta de Arquitectura de Internet o *IAB* por sus siglas en inglés).[4] IAB proporciona el enfoque y coordinación para gran parte de la investigación y desarrollo subyacentes de los protocolos TCP/IP, y también guía la evolución de Internet. Decide qué protocolos son parte obligatoria del grupo TCP/IP y establece políticas oficiales.

[4] IAB originalmente eran las siglas de *Internet Activities Board* (Junta de Actividades de Internet).

Conformado en 1983 cuando ARPA reorganizó la Junta de Control y Configuración de Internet, la IAB heredó mucho de su esquema del grupo anterior. Sus metas inciales fueron alentar el intercambio de ideas entre los principios comprendidos en la investigación relacionada con el TCP/IP e Internet, así como mantener a los investigadores enfocados en los objetivos comunes. Durante los primeros seis años, la IAB pasó de ser un grupo de investigación específica de ARPA a una organización autónoma. Durante esos años, cada miembro de la IAB dirigió una *Fuerza de Tarea de Internet*, asignada a investigar un problema o un grupo de aspectos considerados importantes. La IAB consistía en unas diez fuerzas de tarea, con esquemas que iban de, por ejemplo, uno que investigaba cómo la carga de tráfico de varias aplicaciones afectaba a Internet, a otro que manejaba problemas de ingeniería a corto plazo de Internet. La IAB se reunía muchas veces durante el año para escuchar reportes de estado de cada fuerza de tarea, escuchar y revisar directivas técnicas, discutir políticas e intercambiar información con los representantes de agencias como ARPA y NSF, quienes proporcionaban fondos para las operaciones y la investigación de Internet.

El presidente de la IAB ostentaba el título de *Arquitecto de Internet* y era responsable de la sugerencia de directivas técnicas y de la coordinación de actividades de las fuerzas de tarea. El presidente de la IAB establecía nuevas fuerzas de tarea con el consejo de la Junta y también representaba a la IAB ante otros organismos.

Los que apenas comienzan a conocer el TCP/IP a veces se sorprenden al saber que la IAB no maneja un gran presupuesto; aunque establecía las directivas, no proporcionaba fondos para la mayor parte de la investigación e ingeniería que realizaba. Por el contrario, eran los voluntarios los que realizaban casi todo el trabajo. Todos los miembros de la IAB eran responsables de reclutar voluntarios que sirvieran en sus fuerzas de tarea, convocar y realizar reuniones, y reportar los progresos a la IAB. Por lo general, los voluntarios llegaban de la comunidad de investigación o de organizaciones comerciales que producían o utilizaban el TCP/IP. Los investigadores activos participaban en las actividades de las fuerzas de tarea de Internet por dos razones. Por una parte, prestar sus servicios en una fuerza de tarea proporcionaba oportunidades para aprender sobre nuevos problemas de investigación. Por otra, debido a que las nuevas ideas y las soluciones a problemas, diseñadas y probadas por las fuerzas de tarea a menudo se convertían en parte de la tecnología del TCP/IP de Internet, los miembros se dieron cuenta de que su trabajo tenía una influencia directa y positiva en el campo de trabajo.

1.6 Reorganización de IAB

Para el verano de 1989, la tecnología TCP/IP así como Internet habían crecido más allá del proyecto inicial de investigación y se convirtieron en medios de producción de los que miles de personas dependen para sus negocios diarios. Ya no fue posible introducir nuevas ideas al cambiar algunas instalaciones fuera de las horas de trabajo. Hasta cierto punto, los cientos de compañías comerciales que ofrecían productos TCP/IP determinaban si los productos interoperarían, al decidir cuándo incorporar cambios en su software. Los investigadores que bosquejaban las especificaciones y que probaban nuevas ideas en los laboratorios ya no podían esperar la aceptación y el uso inmediato de las ideas. Era irónico que los investigadores que diseñaron y vieron cómo se desarrolló el TCP/IP se encontraran rebasados por el éxito comercial de su creación. De manera breve, el TCP/IP se convirtió en una tecnología de producción exitosa y el mercado comenzó a dominar su evolución.

Para reflejar la realidad política y comercial tanto del TCP/IP como de Internet, la IAB fue reorganizada en el verano de 1989. La presidencia cambió. Los investigadores fueron transferidos de la IAB a un grupo subsidiario, y se constituyó una nueva IAB que incluía a los representantes de la comunidad ahora más amplia.

La figura 1.1 ilustra la nueva organización de la IAB y la relación de los subgrupos.

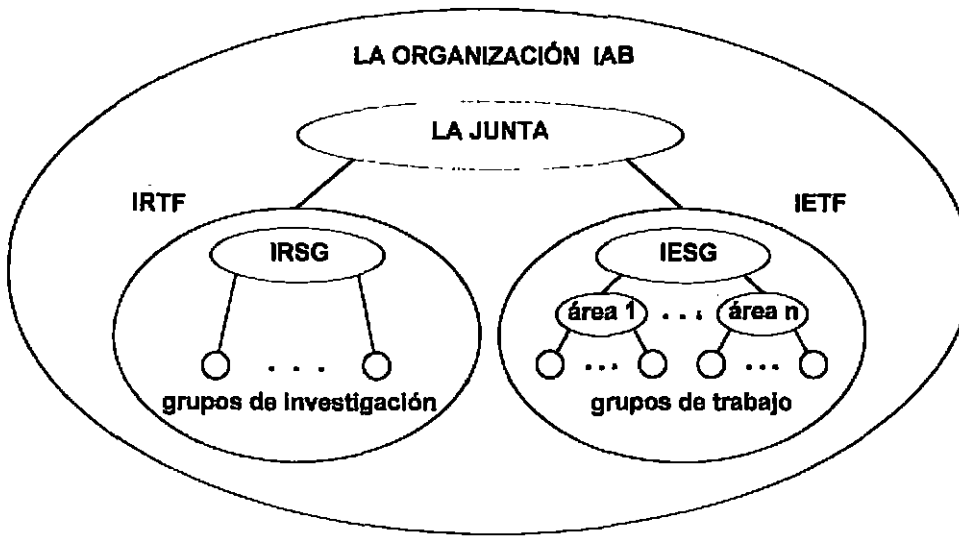


Figura 1.1 Estructura de la IAB después de la reorganización de 1989.

Como se muestra en la figura 1.1, además de la Junta en sí misma, la IAB contiene dos grupos mayores: la *Fuerza de Tarea de Investigación Internet (IRTF)* y la *Fuerza de Tarea de Ingeniería Internet (IETF)*.

Como su nombre lo indica, la IETF se concentra en problemas de ingeniería a corto y mediano plazo. La IETF existía en la estructura original de la IAB y su éxito proporcionó parte de la motivación para la reorganización. A diferencia de la mayor parte de las fuerzas de tarea de la IAB, las cuales estaban limitadas a pocas personas que se enfocaban en un tema específico, la IETF creció hasta incluir a docenas de miembros activos que trabajaban juntos en muchos problemas. Antes de la reorganización, la IETF estaba dividida en más de 20 *grupos de trabajo*, cada uno enfocado en un problema específico. Los grupos de trabajo mantenían reuniones individuales para formular soluciones a los problemas. Además, toda la IETF se reunía de manera regular para escuchar reportes de los grupos de trabajo y para discutir cambios o adiciones propuestas a la tecnología TCP/IP. Generalmente realizadas tres veces al año, las reuniones de la IETF completa atraían a cientos de participantes y espectadores. La IETF se había vuelto demasiado grande para que el presidente la manejara.

Debido a que la IETF era conocida a través de Internet, y ya que sus reuniones eran ampliamente reconocidas y atendidas, la estructura reorganizada de la IAB mantiene a la IETF, pero la divide en aproximadamente doce áreas, cada una con su propio gerente. El presidente de la IETF y los gerentes

de área forman el *Grupo de Control de Ingeniería de Internet (IESG)*, que son las personas responsables de coordinar los esfuerzos de los grupos de trabajo de la IETF. En la actualidad, el nombre "IETF" se refiere al cuerpo completo, incluyendo al presidente, los gerentes de área y todos los miembros de los grupos de trabajo.

Creada durante la reorganización, la Fuerza de Tarea de Investigación de Internet (IRTF) es la contraparte de investigación de la IETF. La IRTF coordina las actividades de investigación relacionadas con los protocolos TCP/IP y con la arquitectura de la red de redes en general. Al igual que la IETF, la IRTF cuenta con un pequeño grupo llamado *Grupo de Control de Investigación de Internet* o *IRSG*, que establece prioridades y coordina las actividades de investigación. A diferencia de la IETF, la IRTF actualmente es una organización mucho más pequeña y mucho menos activa. Cada miembro de la IRSG preside un Grupo de Investigación de Internet voluntario, análogo a los grupos de trabajo de la IETF; la IRTF no está dividida en áreas.

1.7 Sociedad Internet

En 1992, cuando Internet se alejó de las raíces del gobierno de los Estados Unidos, se formó una sociedad para alentar la participación en Internet. Llamada *Sociedad Internet*, el grupo es una organización interna inspirada por la National Geographic Society. Como anfitriona de la IAB, la Sociedad Internet ayuda a que las personas se unan y utilicen Internet alrededor del mundo.

1.8 Solicitud de comentarios de Internet

Hemos dicho que ninguna marca es dueña de la tecnología TCP/IP, ni tampoco ninguna sociedad profesional o cuerpo de estándares. Por lo tanto, la documentación de protocolos, estándares y políticas no se puede obtener con un distribuidor. Por el contrario, es la Fundación Nacional de Ciencias la que proporciona fondos a un grupo en AT&T para que mantenga y distribuya información sobre el TCP/IP y sobre la Internet global. Conocido como *Centro de Información de la Red Internet (INTERNIC)*,[5] maneja muchos detalles administrativos para Internet, además de distribuir la documentación.

La documentación del trabajo en Internet, las propuestas para protocolos nuevos o revisados, así como los estándares del protocolo TCP/IP, aparecen en una serie de reportes técnicos llamados *Solicitudes de Comentarios de Internet o RFC*. (Versiones anteriores de los RFC se conocen como *Bosquejos Internet*.) Los RFC pueden ser cortos o largos, pueden incluir conceptos mayores o detalles, y pueden ser estándares o simples propuestas sobre nuevos protocolos.[6] El editor RFC es un miembro de la IAB. Mientras se editan los RFC, no se refiere a ellos de la misma forma que a papeles académicos de investigación. También, algunos reportes pertinentes a Internet fueron

[5] Pronunciado "Inter-Nick" debido a sus siglas, la organización es sucesora del Centro de Información de Red original (NIC).

[6] El Apéndice *I* contiene una introducción a los RFC, la cual examina su diversidad, incluyendo algunas bromas que han surgido.

publicados en una serie de reportes más antigua y paralela llamada *Notas de Ingeniería de Internet* o *IEN*. Aunque la serie IEN ya no está activa, no todos los IEN aparecen en la serie RFC. A través del texto, se encuentran referencias a RFC y a unos pocos IEN.

La serie RFC está numerada en forma secuencial en el orden cronológico en que se escriben los RFC. Cada RFC, nuevo o revisado, tiene asignado un nuevo número, por lo que los lectores deben tener cuidado en obtener la versión con número más alto de un documento; está disponible un índice para ayudar a identificar la versión correcta.

Para auxiliar a INTERNIC y para hacer más rápida la recuperación de documentos, muchos sitios alrededor del mundo almacenan copias de los RFC y las ponen a disposición de la comunidad. Una persona puede obtener un RFC por correo postal, electrónico o directamente a través de Internet utilizando un programa de transferencia de archivos. Además, INTERNIC y otras organizaciones ponen a disposición versiones preliminares de documentos RFC, conocidas como *bosquejos Internet*. Pregunte a un experto en redes locales cómo obtener los RFC o bosquejos Internet en su localidad, o consulte el Apéndice *I* para obtener más instrucciones de cómo recuperarlos.

1.9 Protocolos y estandarización de Internet

Los lectores familiarizados con las redes de comunicación de datos se dan cuenta que existen muchos estándares para protocolos de comunicación. Muchos de ellos existían antes que Internet, así que surge una pregunta: ¿por qué los diseñadores de Internet inventaron nuevos protocolos cuando ya existían muchos estándares internacionales? La respuesta es compleja, pero a continuación se encuentra una regla simple:

> *Utilizar estándares existentes de protocolo siempre que dichos estándares se puedan aplicar; inventar nuevos protocolos sólo cuando los estándares existentes no sean suficientes, y estar preparado a utilizar nuevos estándares cuando estén disponibles y proporcionen una funcionalidad equivalente.*

Así que, a pesar de parecer lo contrario, el Grupo de Protocolos TCP/IP de Internet no fue diseñado para ignorar o evitar los estándares ya existentes. Surgió solamente porque ninguno de los protocolos ya existentes satisfacía la necesidad de un sistema de comunicación interoperable para enlace de redes.

1.10 Crecimiento y tecnologías del futuro

Tanto la tecnología TCP/IP como Internet continúan evolucionando. Se siguen proponiendo nuevos protocolos; los más antiguos se están revisando. La NSF añadió una considerable complejidad al sistema al introducir una red de columna vertebral, redes regionales y cientos de redes a nivel de campus. Otros grupos alrededor del mundo se conectan día con día a Internet. Sin embargo, el cambio más significativo no viene de la adición de conexiones de redes, sino del tráfico adicional.

Cuando nuevos usuarios se conectan a Internet y aparecen nuevas aplicaciones, los patrones de tráfico cambian. Cuando los físicos, químicos y biólogos comenzaron a utilizar Internet, intercambiaban archivos de datos sobre sus experimentos. Dichos archivos parecían muy grandes comparados con los mensajes de correo electrónico. Cuando Internet se volvió más popular y los usuarios comenzaron a rastrear información utilizando servicios como *gopher* y *World Wide Web*, el tráfico se incrementó de nuevo.

Para incorporar el crecimiento de tráfico, la capacidad de la columna vertebral NSFNET ya se había incrementado tres veces, aumentando su capacidad aproximadamente 840 veces en comparación con la original; se prevé para 1995 otro incremento con factor de 3. En la actualidad, es difícil visualizar un fin de la necesidad de mayor capacidad.

El crecimiento en las demandas para las redes no debe ser una sorpresa. La industria de la computación ha disfrutado por muchos años de una demanda continua de mayor poder de procesamiento y de mayor almacenamiento de datos. Los usuarios apenas han comenzado a entender cómo utilizar las redes. En el futuro podemos esperar incrementos continuos en la demanda de comunicaciones. Por lo tanto, se necesitarán tecnologías de comunicación con mayor capacidad para incorporar el crecimiento.

En la figura 1.2 se resume la expansión de Internet y se ilustra un componente importante del *crecimiento: el cambio en la complejidad surge porque muchos grupos autónomos manejan partes de* la Internet global. Los diseños iniciales para muchos subsistemas dependen de un manejo centralizado. Se necesita mucho esfuerzo para extender dichos diseños e incorporar el manejo descentralizado.

	número de redes	número de computadoras	número de gerentes
1980	10	10^2	10^0
1990	10^3	10^5	10^1
1997	10^8	10^8	10^2

Figura 1.2 Crecimiento de la Internet conectada. Además de los incrementos en el tráfico que resultaron del incremento en tamaño, Internet afronta la complejidad resultante del manejo descentralizado del desarrollo y operaciones.

1.11 Organización del texto

El *material sobre el* TCP/IP se escribió en tres volúmenes. En este volumen se presenta la tecnología TCP/IP, las aplicaciones que la utilizan y la arquitectura de la Internet global con mayor detalle. En él se tratan los fundamentos de los protocolos como el TCP y el IP, y se muestra cómo se conjugan en una red de redes. Además de proporcionar detalles, el texto resalta los principios generales que subyacen bajo los protocolos de red y explica por qué los protocolos TCP/IP se adaptan fácilmente a tantas tecnologías subyacentes de redes físicas. En el volumen II, se analizan más a fondo los detalles internos de los protocolos TCP/IP y se muestra cómo se implementan. En él se presenta el código de un sistema de trabajo para ilustrar cómo los protocolos individuales trabajan juntos y contiene detalles útiles para las personas responsables de la construcción de una red de redes corpo-

rativa. En el volumen III se muestra cómo las aplicaciones distribuidas utilizan el TCP/IP para comunicarse. En él se enfoca el paradigma cliente-servidor, la base para toda la programación distribuida. También se discute la interfaz entre programas y protocolos,[7] y se muestra cómo se organizan los programas clientes y servidores. Además, en el volumen III se describe el concepto del procedimiento remoto y se muestra cómo los programadores utilizan herramientas para elaborar software cliente y servidor.

Hasta ahora hemos hablado sobre la tecnología TCP/IP e Internet en términos generales, resumiendo los servicios proporcionados y la historia de su desarrollo. El siguiente capítulo proporciona un breve resumen sobre el tipo de hardware de red utilizado a través de todo Internet. Su propósito no es el de iluminar los matices del hardware de una marca en particular, sino enfocarse en las características de cada tecnología, que son de primordial importancia para un arquitecto de red de redes. Los capítulos siguientes ahondan en los protocolos y en Internet, cumpliendo tres propósitos: en ellos se explora los conceptos generales y se revisa el modelo arquitectónico de Internet, se examinan los detalles de los protocolos TCP/IP y se tratan los estándares para servicios de alto nivel como correo electrónico y transferencia electrónica de archivos. De los capítulos *3* a *12*, se revisan los principios fundamentales y se describe el software de protocolo de red encontrado en cualquier máquina que utilice el TCP/IP. En los siguientes capítulos, se describen servicios que abarcan muchas máquinas, incluyendo la propagación de información de ruteo, definición de nombres y aplicaciones como el correo electrónico.

Al final de los capítulos, se encuentran dos apéndices. El primero contiene una guía a los RFC. En él se describe en forma mayor los RFC encontrados en este capítulo y se proporcionan ejemplos de la información que se puede encontrar en los RFC. También se describe a detalle cómo obtener los RFC por medio de correo electrónico, correo postal y transferencia de archivos. Por último, debido a que el índice estándar de los RFC está en orden cronológico, en el apéndice se presenta una lista de los RFC organizados por tema, para facilitar a los novatos encontrar un RFC correspondiente a cierto tema.

El segundo apéndice contiene una lista alfabética de términos y abreviaturas utilizados a lo largo de todo el texto. Debido a que los novatos a menudo encuentran la nueva terminología abrumadora y difícil de recordar, se les alienta a utilizar la lista alfabética en vez de volver a buscar en el texto.

1.12 Resumen

Una red de redes consiste en un grupo de redes conectadas que actúan como un todo coordinado. La mayor ventaja de una red de redes es que proporciona interconexión universal y permite que grupos individuales utilicen cualquier hardware de red que satisfaga sus necesidades. Examinaremos los principios subyacentes de la comunicación mediante red de redes en general y los detalles de un grupo de protocolos de una red de redes en particular. También discutiremos cómo se utilizan los protocolos para red de redes. Nuestra tecnología de ejemplo, llamada TCP/IP, cuyo nombre proviene de sus dos protocolos principales, fue desarrollada en la Agencia de Proyectos Avanzados de Investigación. Proporciona la base para la Internet global, una gran red de redes operacional que

[7] El Volumen III está disponible en dos versiones: una que utiliza la interfaz socket y otra que emplea la interfaz de capa de transporte.

interconecta universidades, corporaciones y dependencias gubernamentales en muchos países alrededor del mundo. La Internet global se está expandiendo con rapidez.

PARA CONOCER MÁS

La Cerf's *A History of the ARPANET* (1989) y la *History of the Internet Activities Board* (RFC 1160) proporcionan una lectura fascinante y encaminan al lector hacia documentos antiguos de investigación sobre el TCP/IP y sobre el enlace de redes. Denning (Nov-Dic 1989) proporciona una perspectiva distinta de la historia de ARPANET. Jennings et. al. (1986) analiza la importància para los científicos de las redes de computadoras. Denning (Sept-Oct 1989) también resalta la importancia del enlace de redes y proporciona un posible escenario para una red de redes a nivel mundial. El Comité Federal de Coordinación para la Ciencia, Ingeniería y Tecnología (FCCSET), sugiere que el enlace de redes debería ser una prioridad nacional.

La IETF publica minutas de sus reuniones regulares las cuales están disponibles en la Corporación para las Iniciativas de Investigación Nacional en Reston, VA. El *Journal of Internetworking: Research and Experience* proporciona reportes sobre la investigación del enlace de redes, haciendo énfasis en la validación experimental de ideas. El periódico *Connexions* (Jacobsen 1987-), contiene artículos sobre el TCP/IP e Internet, así como declaraciones oficiales políticas hechas por la IAB. Por último, se alienta al lector a recordar que el grupo de protocolos TCP/IP así como Internet continúan cambiando; se puede encontrar nueva información en los RFC, así como en conferencias anuales como el Simposio ACM SIGCOMM y en los eventos NETWORLD+INTEROP de la Compañía Interop.

EJERCICIOS

1.1 Explore los programas de aplicación en su sitio de trabajo que utilicen el TCP/IP.

1.2 Averigüe si su sitio de trabajo tiene conexión a Internet.

1.3 Los productos TCP/IP obtienen ganancias brutas de más de mil millones de dólares al año. Lea publicaciones sobre comercio para encontrar una lista de fabricantes que ofrezcan dichos productos.

2

Reseña de las tecnologías subyacentes de red

2.1 Introducción

Es importante entender que Internet no es un nuevo tipo de red física, sino un método de interconexión de redes físicas y un conjunto de convenciones para el uso de redes que permite a las computadoras conectadas a éstas interactuar unas con otras. Si bien el hardware de las redes desempeña un papel de menor importancia en el diseño total, entender la tecnología utilizada para enlazar redes de redes requiere de la distinción entre sus mecanismos de bajo nivel proporcionados por el hardware mismo y la infraestructura de alto nivel proporcionada por el software de protocolo de TCP/IP. También es importante entender cómo la infraestructura proporcionada por la tecnología de conmutación de paquetes afecta nuestra selección de las abstracciones de alto nivel.

Este capítulo introduce el concepto y la terminología básica de conmutación de paquetes y revisa algunas de las tecnologías de hardware de red subyacentes que se han utilizado para el enlace entre redes TCP/IP. En capítulos posteriores se describe cómo esas redes se interconectan y cómo el protocolo TCP/IP se adapta a la gran diversidad existente en el hardware. La lista que se presenta aquí en realidad no es completa, esto demuestra claramente la variedad existente entre redes físicas que operan con el TCP/IP. El lector puede omitir con seguridad muchos de los detalles técnicos pero deberá tratar de asimilar la idea de la conmutación de paquetes y también imaginar la construcción de sistemas de comunicación homogéneos que usen tecnología de hardware heterogénea. Es muy importante que el lector observe de cerca los detalles de los esquemas de direccionamiento físico de las diversas tecnologías en uso. En capítulos posteriores, se tratará con detalle la forma en que los protocolos de alto nivel utilizan el direccionamiento físico.

2.2 Dos enfoques de la comunicación por red

Si se realiza una conexión entre una computadora y otra o entre terminales y computadoras, la comunicación entre redes puede dividirse en dos tipos básicos: de *circuitos conmutados* (a veces llamada *orientada a la conexión*) y por *conmutación de paquetes*[1] (a veces llamada *sin conexión*). Las redes conmutadas de circuitos operan formando una conexión delicada (circuito) entre dos puntos. El sistema telefónico de Estados Unidos utiliza tecnología de circuitos conmutados —una llamada telefónica establece un circuito desde el teléfono que la origina a través de la oficina local de conmutación, a través de las líneas troncales, hacia la oficina remota de conmutación y finalmente hasta el teléfono destino. Mientras este circuito se mantenga, el equipo telefónico tomará muestras del micrófono continuamente, codificará las muestras en forma digital y las transmitirá a través del circuito hasta el receptor. El emisor garantiza que las muestras pueden ser enviadas y reproducidas dado que los circuitos proporcionan una trayectoria de envío de datos de 64 Kbps (miles de bits por segundo), ésta es la cifra necesaria para garantizar el envío de la voz digitalizada. La ventaja de los circuitos conmutados reside en su capacidad garantizada: una vez que un circuito se establece, ninguna otra actividad de la red se verá disminuida en su capacidad. Una desventaja de la conmutación de circuitos es el costo: el costo de un circuito es fijo, independientemente del tráfico. Por ejemplo, se debe pagar una cuota fija por una llamada telefónica, sin importar si las dos partes que se comunicaron hablaron o no en todo momento.

Las redes de conmutación de paquetes, normalmente utilizadas para conectar computadoras, funcionan de una manera por completo diferente. En una red de conmutación de paquetes, la información es transferida a través de la red dividida en pequeñas unidades llamadas *paquetes* que son multiplexadas en conexiones entre máquinas de alta capacidad. Un paquete por lo general contiene sólo unos cuantos cientos de octetos de datos y transporta información de identificación que permite al hardware de la red saber cómo enviar el paquete hacia un destino específico. Por ejemplo, un archivo grande que será transmitido entre dos máquinas debe ser fragmentado en muchos paquetes que serán enviados a través de la red en un momento dado. El hardware de red envía los paquetes al destino especificado, donde el software los reensamblará de nuevo en un solo archivo. Una gran ventaja de la conmutación de paquetes es que comunicaciones múltiples entre computadoras pueden procesarse de manera concurrente, con conexiones entre máquinas compartidas por todos los pares de máquinas que se están comunicando. La desventaja, por supuesto, es que si la actividad se incrementa, un par de computadoras que se esté comunicando en un momento dado dispondrá de una menor capacidad de la red. Esto significa que cada vez que una red de conmutación de paquetes se sobrecarga, las computadoras que están usando la red deberán esperar para poder continuar enviando paquetes.

A pesar de las dificultades potenciales que no garantizan la capacidad de la red, las redes de conmutación de paquetes se han vuelto muy populares. Los motivos para adoptar la conmutación de paquetes son el costo y el desempeño. Dado que múltiples máquinas pueden compartir el hardware de red, se requiere de pocas conexiones y el costo se reduce. Como los ingenieros han sido hábiles en construir hardware de red de alta velocidad, la capacidad normalmente no es un problema. Como muchas interconexiones entre computadoras utilizan conmutación de paquetes, en este libro nos referiremos con el término *red* a las redes de conmutación de paquetes.

---

[1] De hecho, es posible construir tecnologías híbridas de hardware; para nuestros propósitos, sólo la diferencia en la funcionalidad es importante.

2.3 Redes de área amplia y local

Las redes de conmutación de paquetes que deben recorrer distancias geográficas grandes (por ejemplo, el territorio de Estados Unidos) son fundamentalmente diferentes de las que deben recorrer distancias cortas (como, una habitación). Para ayudar a caracterizar las diferencias en la capacidad y las proyecciones de uso, la tecnología de conmutación de paquetes se divide con frecuencia en dos grandes categorías: *Wide Area Networks* (*redes de área amplia* o *WAN* por sus siglas en inglés) y *Local Area Networks* (*redes de área local* o *LAN* por sus siglas en inglés). Las dos categorías no tienen una definición formal. Tal vez por ello, los vendedores aplican los términos con cierta vaguedad para auxiliar a los clientes a distinguir entre las dos tecnologías.

La tecnología WAN, a veces llamadas *long haul networks* (*redes de gran alcance*), proporcionan comunicación que cubre grandes distancias. Muchas tecnologías WAN no tienen un límite de distancia de recorrido; una WAN puede permitir que dos puntos inmediatamente lejanos se comuniquen. Por ejemplo, una WAN puede recorrer un continente o unir computadoras a través de un océano. Por lo común las WAN operan más lentamente que las LAN y tienen tiempos de retraso mucho mayores entre las conexiones. La velocidad normal para una WAN llega a un rango que va de los 56 Kbps a 155 Mbps (millones de bits por segundo). Los retardos para una WAN pueden variar de unos cuantos milisegundos a varias decenas de segundos.[2]

Las tecnologías LAN proporcionan las velocidades de conexión más altas entre computadoras, pero sacrifican la capacidad de recorrer largas distancias. Por ejemplo, una LAN común recorre un área pequeña, como un edificio o un pequeño campus, y opera dentro de un rango que va de los 10 Mbps a los 2 Gbps (billones de bits por segundo). Debido a que la tecnología LAN cubre distancias cortas, ofrece tiempos de retraso mucho menores que las WAN. Los tiempos de retardo en una LAN pueden ser cortos, como unas cuantas decenas de milisegundos, o largos, 10 milisegundos.

Ahora, podemos mencionar un principio general de la relación entre la velocidad y la distancia: las tecnologías que proporcionan altas velocidades de comunicación operan en distancias cortas. Existen otras diferencias entre las tecnologías de las categorías señaladas. En la tecnología LAN, cada computadora por lo general contiene un dispositivo de interfaz de red que conecta la máquina directamente con el medio de la red (por ejemplo, un alambre de cobre o cable coaxial). Con frecuencia la red por sí misma es pasiva, depende de dispositivos electrónicos conectados a las computadoras para generar y recibir las señales eléctricas necesarias. En la tecnología WAN, una red por lo común consiste en una serie de computadoras complejas, llamadas *conmutadoras de paquetes*, interconectadas por líneas de comunicación y modems. El tamaño de una red puede extenderse si se le añade un nuevo conmutador y otras líneas de comunicación. La conexión de una computadora de usuario a una WAN significa conectarla a uno de los conmutadores de paquetes. Cada conmutador extiende la ruta de la WAN e introduce un retardo cuando recibe un paquete y lo envía al siguiente conmutador. De esta manera, la extensión de una WAN hace que la ruta del tráfico que pasa a través de ella se extienda.

Este libro trata el software que oculta las diferencias tecnológicas entre las redes y hace que la interconexión sea independiente del hardware subyacente. Para apreciar las selecciones de diseño en el software, es necesario entender cómo interactúa con el hardware de red. La sección siguiente presenta ejemplos de tecnologías de red que han sido utilizadas en Internet y muestra algunas de las diferencias entre ellas. En capítulos posteriores se muestra como el software del TCP/IP aísla cada diferencia y hace que el sistema de comunicación sea independiente de la tecnología de hardware subyacente.

[2] Estos retardos se deben a que las WAN se comunican por medio de envío de señales a los satélites en órbita alrededor de la Tierra..

2.3.1 Direcciones de hardware de red

Cada tecnología de hardware de red define un *mecanismo de direccionamiento* que las computadoras utilizan para especificar el destino de cada paquete. A cada computadora conectada a una red se le asigna una dirección única, por lo general un número entero. Un paquete enviado a través de una red incluye *un campo de dirección de destino* que contiene la dirección del recipiente de destino. La dirección de destino aparece en el mismo lugar en todos los paquetes, haciendo posible que el hardware de red localice la dirección de destino fácilmente. Cuando se envía información se debe conocer la dirección del recipiente de destino y colocar la dirección del recipiente en el campo de dirección de destino del paquete, antes de transmitir el paquete.

Cada tecnología de hardware especifica cómo las computadoras son asignadas a una dirección. El hardware especifica, por ejemplo, el número de bits en la dirección así como la localización del campo de dirección de destino en un paquete. Aun cuando algunas tecnologías utilizan esquemas de direccionamiento compatibles, muchas no lo hacen así. Este capítulo contiene unos cuantos ejemplos de esquemas de direccionamiento de hardware; en capítulos posteriores se explica como el TCP/IP se adapta a los diversos esquemas de direccionamiento de hardware.

2.4 Tecnología Ethernet

Ethernet es el nombre que se le ha dado a una popular tecnología LAN de conmutación de paquetes inventada por Xerox PARC a principios de los años setenta. Xerox Corporation, Intel Corporation y Digital Equipment Corporation estandarizaron Ethernet en 1978; IEEE liberó una versión compatible del estándar utilizando el número 802.3. Ethernet se ha vuelto una tecnología LAN popular; muchas compañías, medianas o grandes, utilizan Ethernet. Dado que Ethernet es muy popular existen muchas variantes; analizaremos el diseño original primero y después cubriremos algunas variantes. Cada cable Ethernet tiene aproximadamente 1/2 pulgada de diámetro y mide hasta 500 m de largo. Se añade una resistencia entre el centro del cable y el blindaje en cada extremo del cable para prevenir la reflexión de señales eléctricas.

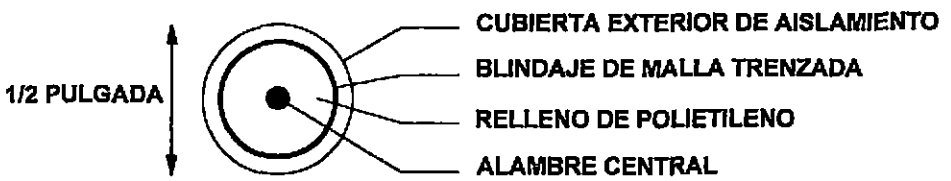


Figura 2.1 Sección transversal del cable coaxial utilizado en la red Ethernet original.

El diseño original de Ethernet utilizaba un cable coaxial como el mostrado en la figura 2.1. Llamado *ether*, el cable por sí mismo es completamente pasivo; todos los componentes electrónicos activos que hacen que la red funcione están asociados con las computadoras que se comunican a la red.

La conexión entre una computadora y un cable coaxial Ethernet requiere de un dispositivo de hardware llamado *transceptor*. Físicamente la conexión entre un transceptor y el cable Ethernet requiere de una pequeña perforación en la capa exterior del cable como se muestra en la figura 2.2. Los técnicos con frecuencia utilizan el término *tap* para describir la conexión entre un transceptor Ethernet y el cable. Por lo general, una pequeña aguja de metal montada en el transceptor atraviesa la perforación y proporciona el contacto eléctrico con el centro del cable y el blindaje trenzado. Algunos fabricantes de conectores hacen que el cable se corte y se inserte una "T".

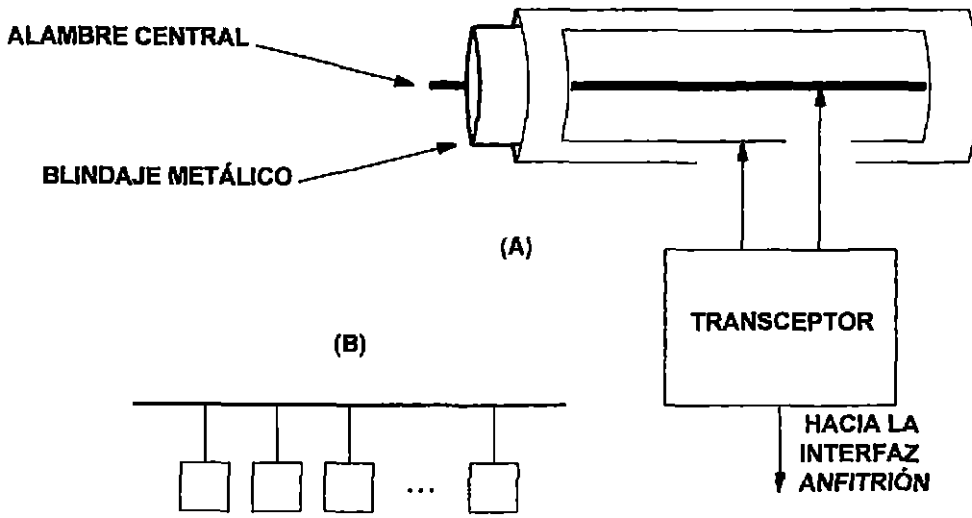


Figura 2.2. (a) Vista recortada de un cable de red Ethernet en la que se muestra los detalles de las conexiones eléctricas entre un transceptor y el cable, y (b) diagrama esquemático de una red Ethernet con varias computadoras conectadas.

Cada conexión a una red Ethernet tiene dos componentes electrónicos mayores. Un *transceptor* es conectado al centro del cable y al blindaje trenzado del cable, por medio del cual recibe y envía señales por el cable ether. Una *interfaz anfitrión o adaptador anfitrión* se conecta dentro del bus de la computadora (por ejemplo, en una tarjeta madre) y se conecta con el transceptor.

Un transceptor es una pequeña pieza de hardware que por lo común se encuentra físicamente junto al cable ether. Además del hardware análogo que envía y controla las señales eléctricas en el cable ether, un transceptor contiene circuitería digital que permite la comunicación con una computadora digital. El transceptor, cuando el cable ether está en uso, puede recibir y traducir señales eléctricas analógicas hacia o desde un formato digital en el cable ether. Un cable llamado *Attachment Unit Interface (AUI)* conecta el transceptor con la tarjeta del adaptador en una computadora anfitrión. Informalmente llamado *cable transceptor*, el cable AUI contiene muchos cables. Los cables transportan la potencia eléctrica necesaria para operar el transceptor, las señales de control para la operación

del transceptor y el contenido de los paquetes que se están enviando o recibiendo. La figura 2.3 ilustra cómo los componentes forman una conexión entre el bus del sistema de una computadora y un cable Ethernet.

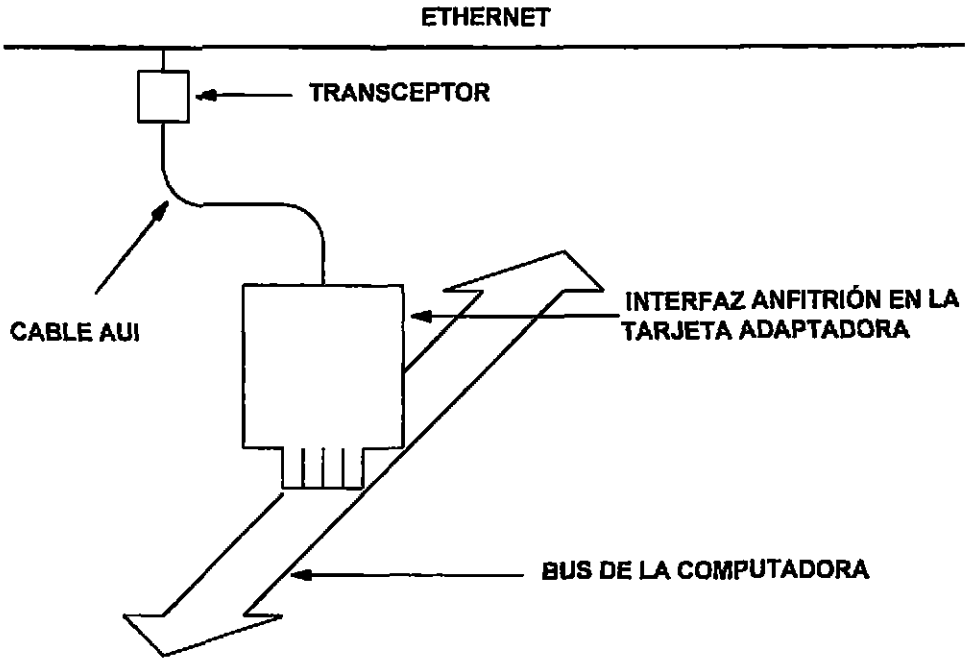


Figura 2.3 Los dos componentes electrónicos principales que forman una conexión entre el bus de una computadora y la red Ethernet. El cable AUI que conecta la interfaz de anfitrión al transceptor transporta corriente de alimentación y señales de control para la operación del transceptor, así como paquetes que se envían o reciben.

Cada interfaz de anfitrión controla la operación de un transceptor de acuerdo a las instrucciones que recibe del software de la computadora. Para el software del sistema operativo, la interfaz aparece como un dispositivo de entrada/salida que acepta instrucciones de transferencia de datos básicas desde la computadora, controla la transferencia del transceptor e interrumpe el proceso cuando éste ha concluido, finalmente reporta la información de estado. Aun cuando el transceptor es un simple dispositivo de harware, la interfaz de anfitrión puede ser compleja (por ejemplo, puede contener un microprocesador utilizado para controlar la transferencia entre la memoria de la computadora y el cable ether).

En la práctica las organizaciones que utilizan el Ethernet original en el ambiente de una oficina convencional extienden el cable Ethernet por el techo de las habitaciones e instalan una conexión para cada oficina conectándola de este modo con el cable. La figura 2.4 ilustra el esquema de cableado físico resultante.

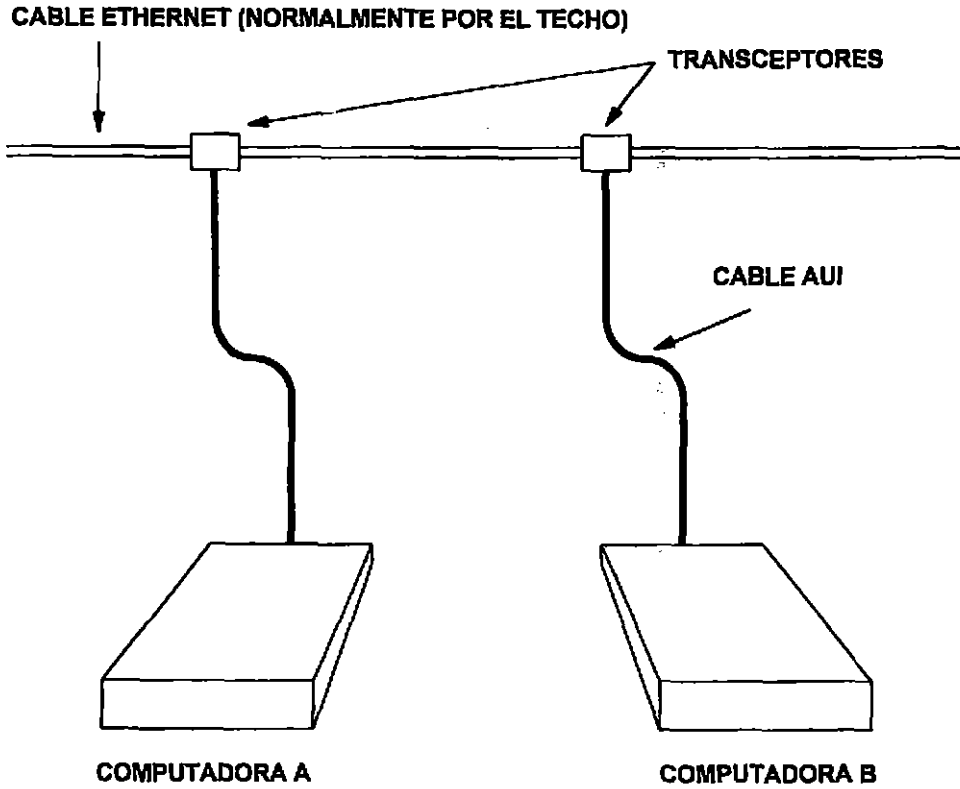


Figura 2.4 Conexión física de dos computadoras a una red Ethernet mediante el uso del esquema de cableado original. En el ambiente de una oficina, el cable Ethernet por lo general se coloca formando una trayectoria en el techo: cada oficina tiene un cable AUI que conecta una computadora en la oficina con el transceptor conectado al cable Ethernet.

2.4.1 Ethernet de cable delgado

Varios componentes de la tecnología Ethernet original tenían propiedades indeseables. Por ejemplo, un transceptor contenía componentes electrónicos, su costo no era *insignificante*. Además, ya que el transceptor estaba localizado en el cable y no en la computadora, éstos podían ser difíciles de accesar o reemplazar. El cable coaxial que forma el ether puede también ser difícil de instalar. En particular, para proporcionar la máxima protección contra la interferencia eléctrica el cable contiene un blindaje pesado que hace que el cable sea difícil de doblar. Por último un cable AUI también es grueso y difícil de doblar.

Para reducir costos en el caso de ambientes como el de las oficinas, en donde no existe mucha interferencia eléctrica, los ingenieros desarrollaron una alternativa de esquema de cableado Ethernet.

Llamada *thin-wire Ethernet* o *thinnet*,[3], el cable coaxial alternativo es más delgado, menos caro y más flexible. Sin embargo, un cable delgado Ethernet tiene algunas desventajas. Dado que no proporciona mucha protección contra la interferencia eléctrica, el cable delgado Ethernet no puede ser colocado junto a equipo eléctrico potente, como podría suceder en el caso de una fábrica. Además, el cable delgado Ethernet cubre distancias algo más cortas y soporta un menor número de conexiones de computadoras por red que el cable grueso Ethernet.

Para reducir aún más los costos con el cable delgado Ethernet, los ingenieros reemplazaron el costoso transceptor con circuitería digital de alta velocidad especial y proporcionaron una conexión directa desde una computadora hasta el cable ether. De esta forma, en el esquema de cable delgado, una computadora contiene tanto la interfaz de anfitrión como la circuitería necesaria para conectar la computadora con el cable. Los fabricantes de pequeñas computadoras y estaciones de trabajo encontraron el esquema del cable delgado Ethernet especialmente atractivo, debido a que podían integrar el hardware de Ethernet en una sola tarjeta de computadora y hacer las conexiones necesarias de manera directa en la parte posterior de la computadora.

Como el cable delgado Ethernet conecta directamente una computadora con otra, el esquema de cableado trabaja bien cuando muchas computadoras ocupan una sola habitación. El cable delgado conecta en forma directa una computadora con otra. Para añadir una nueva computadora sólo es necesario enlazarla con la cadena. En la figura 2.5 se ilustra la conexión utilizada en el esquema de cable delgado de Ethernet.

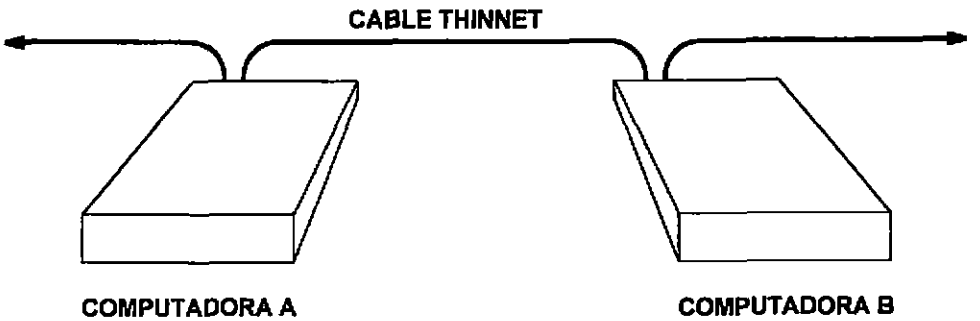


Figura 2.5 Conexión física de dos computadoras que se valen del esquema de cableado thinnet (cable de red delgado). El cable ether pasa directamente de una computadora a otra; no requiere del hardware de transceptores externos.

El esquema de cable delgado de Ethernet está diseñado para conectarse y desconectarse fácilmente. El esquema de cable delgado utiliza *conectores BNC*, los cuales no requieren de herramientas para conectar una computadora con el cable. Así, un usuario puede conectar una computadora al cable delgado Ethernet sin ayuda de un técnico. Por supuesto, permitir que el usuario manipule el cable ether tiene sus desventajas: si un usuario desconecta el cable ether, esto provocará

[3] Para diferenciarlo del cable thin-wire (cable delgado), al cable original de las redes Ethernet se le conoce a veces como *thick-Ethernet* o *thicknet* (cable de red grueso).

que todas las máquinas en el ether queden incomunicadas. En muchos casos, sin embargo, las ventajas superan a las desventajas.

2.4.2 Ethernet de par trenzado

Los avances en la tecnología han hecho posible construir redes Ethernet que no necesitan del blindaje eléctrico de un cable coaxial. Llamada *twisted pair Ethernet* (*Ethernet de par trenzado*), esta tecnología permite que una computadora accese una red Ethernet mediante un par de cables de cobre convencionales sin blindaje, similares a los cables utilizados para conectar teléfonos. La ventaja de usar cables de par trenzado es que reducen mucho los costos y protegen a otras computadoras conectadas a la red de los riesgos que se podrían derivar de que un usuario desconecte una computadora. En algunos casos, la tecnología de par trenzado hace posible que una organización instale una red Ethernet a partir del cableado telefónico existente sin tener que añadir cables nuevos.

Conocido con el nombre técnico de *10Base-T*, el esquema de cableado de par trenzado conecta cada computadora con un *hub* (*concentrador*) Ethernet como se muestra en la figura 2.6.

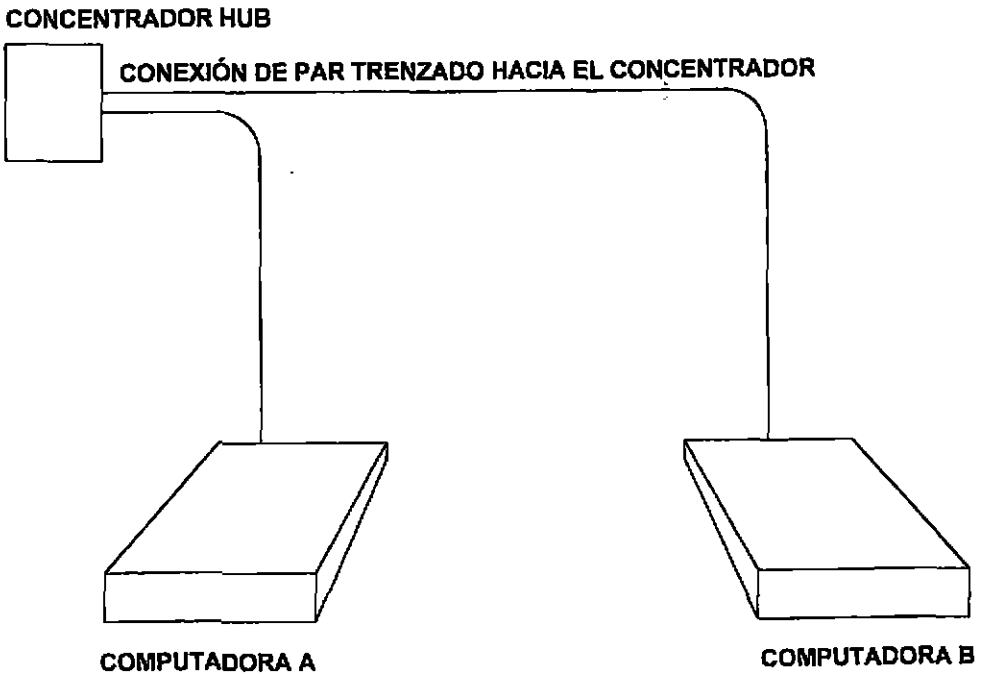


Figura 2.6 Ilustración de una red Ethernet que emplea cableado de par trenzado. Cada computadora se conecta a un concentrador mediante un par de cables convencionales.

El concentrador es un dispositivo electrónico que simula la señal en un cable Ethernet. Físicamente, un concentrador consiste en una pequeña caja que por lo general se aloja en un gabinete para cableado; la conexión entre un concentrador y una computadora debe tener una longitud menor a 100 m. Un concentrador requiere de alimentación eléctrica y puede permitir que el personal autorizado monitoree y controle la operación de la red. Para la interfaz anfitrión, en una computadora, la conexión hacia un concentrador parece operar de la misma forma que la conexión hacia un transceptor. Esto es, un concentrador Ethernet proporciona la misma capacidad de comunicación que un Ethernet delgado o grueso; los concentradores sólo ofrecen una alternativa al esquema de cableado.

2.4.3 Esquemas de cableado múltiple y adaptadores

Una conexión Ethernet de cable grueso requiere de un conector AUI, una conexión para Ethernet de cable delgado requiere de un conector BNC y un conector para 10Base-T requiere de un conector RJ45 que recuerda a los conectores modulares utilizados en los teléfonos. Muchos productos Ethernet permiten que cada usuario seleccione el esquema de cableado. Por ejemplo, las tarjetas adaptadoras para computadoras personales con frecuencia cuentan con los 3 conectores como se muestra en la figura 2.7. Dado que sólo un conector puede usarse a la vez, una computadora que cuenta con un adaptador determinado puede cambiarse de un esquema de cableado a otro con facilidad.

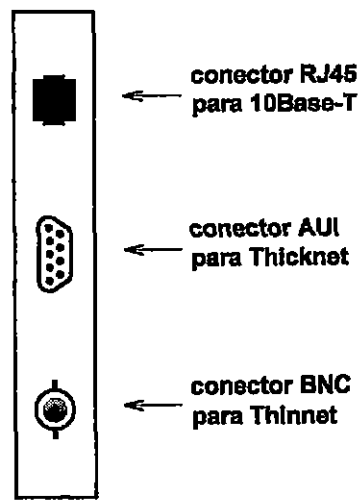


Figura 2.7 Una tarjeta adaptadora Ethernet común con tres conectores para los tres esquemas de cableado Ethernet. Aun cuando el adaptador contiene tres tipos de conector, sólo se puede utilizar un esquema de cableado a la vez.

2.4.4 Propiedades de una red Ethernet

La red Ethernet es una tecnología de bus de difusión de 10 Mbps que se conoce como "entrega con el mejor esfuerzo" y un control de acceso distribuido. Es un *bus* debido a que todas las estaciones comparten un solo canal de comunicación, es de *difusión* porque todos los transceptores reciben todas las transmisiones. El método utilizado para dirigir paquetes de una estación a otra únicamente o al subconjunto de todas las estaciones se analizará más adelante. Por ahora, es suficiente con entender que los transceptores no distinguen las transmisiones —transfiere todos los paquetes del cable a la interfaz anfitrión, la cual selecciona los paquetes que la computadora debe recibir y filtra todos los demás. Las redes Ethernet cuentan con un mecanismo llamado *entrega con el mejor esfuerzo* debido a que el hardware no proporciona información al emisor acerca de si el paquete ha sido recibido. Por ejemplo, si la máquina de destino es apagada, los paquetes enviados se perderán y el emisor no será notificado. Más adelante veremos cómo el protocolo TCP/IP se adapta al hardware de entrega con el mejor esfuerzo.

El control de acceso en las redes Ethernet es distribuido porque, a diferencia de algunas tecnologías de red, Ethernet no tiene una autoridad central para garantizar el acceso. El esquema de acceso de Ethernet es conocido como *Carrier Sense Multiple Access* con *Collision Detect (CSMA/CD)*. Es un CSMA debido a que varias máquinas pueden accesar la red Ethernet de manera simultánea y cada máquina determina si el cable ether está disponible al verificar si está presente una onda portadora. Cuando una interfaz anfitrión tiene un paquete para transmitir verifica el cable ether para comprobar si un mensaje se está transmitiendo (por ejemplo, verificando si existe una portadora). Cuando no se comprueba la presencia de una transmisión, la interfaz de anfitrión comienza a transmitir. Cada transmisión está limitada en duración (dado que hay un tamaño máximo para los paquetes). Además, el hardware debe respetar un tiempo mínimo de inactividad entre transmisiones, esto significa que no se dará el caso de que un par de computadoras que se comuniquen puedan utilizar la red sin que otras máquinas tengan la oportunidad de accesarla.

2.4.5 Recuperación y detección de colisiones

Cuando un transceptor comienza a transmitir, la señal no alcanza todas las partes de la red de manera simultánea. En lugar de ello, la señal viaja a lo largo del cable a una velocidad aproximada al 80% de la velocidad de la luz. De esta forma, es posible que dos transceptores perciban que la red está desocupada y comiencen a transmitir en forma simultánea. Cuando las dos señales eléctricas se cruzan, se produce una perturbación y ninguna de las dos señales será significativa. Este tipo de incidentes se conoce como *colisiones*.

El manejo de las colisiones en Ethernet se resuelve de manera ingeniosa. Cada transceptor monitorea el cable mientras está transmitiendo para explorar si hay alguna señal eléctrica exterior que interfiera con su transmisión. Técnicamente, el monitoreo se conoce como *detección de colisiones* (*CD*), esto hace de Ethernet una red CSMA/CD. Cuando se detecta una colisión, la interfaz de anfitrión aborta la transmisión y espera a que la actividad disminuya, luego intenta de nuevo transmitir. Se debe tener mucho cuidado pues de otra forma la red podría caer en una situación en la que todos los transceptores se ocuparían de intentar transmitir y todas las transmisiones producirían colisiones. Para ayudar a evitar este tipo de situaciones, las redes Ethernet utilizan un procedimiento de retroceso exponencial binario mediante el cual el emisor espera un lapso de tiempo aleatorio, después de la primera colisión esperará el doble de tiempo para intentar transmitir de nuevo, si de nuevo se produce

una colisión esperará cuatro veces el lapso de tiempo inicial antes de realizar un tercer intento y así sucesivamente. El retroceso exponencial evita que se pueda producir un congestionamiento intenso cuando estaciones diferentes tratan de transmitir en forma simultánea. En caso de que se dé un congestionamiento, existe una alta probabilidad de que dos estaciones seleccionen retrocesos aleatorios muy cercanos. Así, la probabilidad de que se produzca otra colisión es alta. Al duplicar el retardo aleatorio, la estrategia de retroceso exponencial distribuye con rapidez los intentos de las estaciones para retransmitir en un intervalo de tiempo razonablemente largo, haciendo que la probabilidad de que se produzcan nuevas colisiones sea muy pequeña.

2.4.6 Capacidad de las redes Ethernet

El estándar Ethernet se define en 10 Mbps, lo cual significa que los datos pueden transmitirse por el cable a razón de 10 millones de bits por segundo. A pesar de que una computadora puede generar datos a la velocidad de la red Ethernet, la velocidad de la red no debe pensarse como la velocidad a la que dos computadoras pueden intercambiar datos. La velocidad de la red debe ser pensada como una medida de la capacidad del tráfico total de la red. Pensemos en una red como en una carretera que conecta varias ciudades y pensemos en los paquetes como en coches en la carretera. Un ancho de banda alto hace posible transferir cargas de tráfico pesadas, mientras que un ancho de banda bajo significa que la carretera no puede transportar mucho tráfico. Una red Ethernet a 10 Mbps, por ejemplo, puede soportar unas cuantas computadoras que generan cargas pesadas o muchas computadoras que generan cargas ligeras.

2.4.7 Direccionamiento de hardware Ethernet

Las redes Ethernet definen un esquema de direccionamiento de 48 bits. Cada computadora conectada a una red Ethernet es asignada a un número único de 48 bits conocido como *dirección Ethernet*. Para asignar una dirección, los fabricantes de hardware de Ethernet adquieren bloques de direcciones Ethernet[4] y las asignan en secuencia conforme fabrican el hardware de interfaz Ethernet. De esta manera no existen dos unidades de hardware de interfaz que tengan la misma dirección Ethernet.

Por lo general, las direcciones Ethernet se fijan en las máquinas en el hardware de interfaz de anfitrión de forma que se puedan leer. Debido a que el direccionamiento Ethernet se da entre dispositivos de hardware, a estos se les llama a veces *direccionamientos* o *direcciones físicas*. Tómese en cuenta la siguiente propiedad importante de las direcciones físicas Ethernet:

> *Las direcciones físicas están asociadas con el hardware de interfaz Ethernet; cambiar el hardware de interfaz a una máquina nueva o reemplazar el hardware de interfaz que ha fallado provocará cambios en la dirección física de la máquina.*

Conociendo la dirección física Ethernet se pueden hacer cambios con facilidad porque los niveles superiores del software de red están diseñados para adaptarse a estos cambios.

[4] El Institute for Electrical and Electronic Engineers (IEEE) maneja el espacio de direcciones Ethernet y asigna las direcciones conforme se necesitan.

El hardware de interfaz anfitrión examina los paquetes y determina qué paquetes deben enviarse al anfitrión. Debe recordarse que cada interfaz recibe una copia de todos los paquetes aun cuando estén direccionados hacia otras máquinas. La interfaz de anfitrión utiliza el campo de dirección de destino de un paquete como filtro. La interfaz ignora los paquetes que están direccionados hacia otras máquinas y selecciona sólo los paquetes direccionados hacia el anfitrión. El mecanismo de direccionamiento y filtrado de hardware es necesario para prevenir que una computadora sea abrumada con la entrada de datos. Aun cuando el procesador central de la computadora podría realizar la verificación, ésta se realiza en la interfaz de anfitrión haciendo que el tráfico en la red Ethernet sea un proceso menos lento en todas las computadoras.

Una dirección Ethernet de 48 bits puede hacer más que especificar una sola computadora destino. Una dirección puede ser de alguno de los tres tipos siguientes:

- La dirección física de una interfaz de red (dirección de *unidifusión*).
- La dirección de *publidifusión* de la red.
- Una dirección de *multidifusión*.

Convencionalmente, la dirección de difusión se reserva para envíos simultáneos a todas las estaciones. Las direcciones de multidifusión proporcionan una forma limitada de difusión en la cual un subconjunto de computadoras en una red acuerda recibir una dirección de multidifusión dada. El conjunto de computadoras participantes se conoce como *grupo de multidifusión*. Para unirse a un grupo de multidifusión, una computadora debe instruir a la interfaz anfitrión para aceptar las direcciones de multidifusión del grupo. La ventaja de la multidifusión reside en la capacidad para limitar la difusión: todas las computadoras en un grupo de multidifusión pueden ser alcanzadas con un solo paquete de transmisión, pero las computadoras que eligen no participar en un grupo de multidifusión en particular no recibirán los paquetes enviados al grupo.

Para adaptarse al direccionamiento de multidifusión y difusión, el hardware de interfaz Ethernet debe reconocer más que la dirección física. Una interfaz anfitrión por lo general acepta hasta dos clases de paquete: los direccionados a la dirección física de la interfaz (esto es, unidifusión) y las direcciones hacia la dirección de difusión de la red. Algunos tipos de interfaz pueden programarse para reconocer direcciones de multidifusión o para alternar entre direcciones físicas. Cuando el sistema operativo comienza a trabajar, éste inicia la interfaz Ethernet, haciendo que se reconozca un conjunto de direcciones. La interfaz entonces examina el campo de direcciones de destino en cada paquete, pasando hacia el anfitrión sólo las transmisiones destinadas a una de las direcciones específicas.

2.4.8 Formato de la trama de Ethernet

La red Ethernet podría pensarse como una conexión de niveles enlazados entre máquinas. De esta manera, la información transmitida podría tener el aspecto de una *trama*.[5] La trama de Ethernet es

[5] El término controlador de *trama (frame)* proviene de las comunicaciones en líneas seriales en las que el emisor estructura los datos al añadir caracteres especiales antes y después de los datos por transmitir.

de una longitud variable pero no es menor a 64 octetos[6] ni rebasa los 1518 octetos (encabezado, datos y CRC). Como en todas las redes de conmutación de paquetes, cada trama de Ethernet contiene un campo con la información de la dirección de destino. La figura 2.8 muestra que la trama de Ethernet contiene la dirección física de la fuente y también la dirección física del destino.

Preámbulo	Dirección de destino	Dirección fuente	Datos de la trama	Tipo de trama	CRC
8 octetos	6 octetos	6 octetos	2 octetos	64-1500 octetos	4 octeto

Figura 2.8 Formato de una trama (paquete) que viaja através de Ethernet, precedida por un preámbulo. Los campos no se dibujaron a escala.

Además de la información para identificar la fuente y el destino, cada trama transmitida a través de Ethernet contiene un *preámbulo, un campo tipo, un campo de datos* y una *Cyclic Redundancy Check* (*verificación por redundancia cíclica* o *CRC*, por sus siglas en inglés). El preámbulo consiste en 64 bits que alternan ceros y unos para ayudar a la sincronización de los nodos de recepción. El CRC de 32 bits ayuda a la interfaz a detectar los errores de transmisión: el emisor computa el CRC como una función de los datos en la trama y el receptor computa de nuevo el CRC para verificar que el paquete se ha recibido intacto.

El campo de tipo de trama contiene un entero de 16 bits que identifica el tipo de datos que se están transfiriendo en la trama. Desde el punto de vista de Internet, el campo de tipo de trama es esencial porque significa que las tramas de Ethernet se *autoidentifican*. Cuando una trama llega a una máquina dada, el sistema operativo utiliza el tipo de trama para determinar qué módulo de software de protocolo se utilizará para procesar la trama. La mayor ventaja de que las tramas se autoidentifiquen es que éstas permiten que múltiples protocolos se utilicen juntos en una sola máquina y sea posible entremezclar diferentes protocolos en una sola red física sin interferencia. Por ejemplo, uno podría tener un programa de aplicación que utiliza protocolos de Internet mientras otro utiliza un protocolo experimental local. El sistema operativo utiliza el campo de tipo de una trama entrante para decidir como procesar el contenido. Veremos que los protocolos TCP/IP utilizan tramas Ethernet autoidentificables para hacer una selección entre varios protocolos.

2.4.9 Extensión de una red Ethernet con repetidores

Aun cuando el cable Ethernet tiene una longitud máxima, las redes pueden extenderse de dos formas: utilizando repetidores y puentes. Un dispositivo de hardware llamado *repetidor* puede emplearse para difundir señales eléctricas de un cable a otro. Sin embargo, sólo un máximo de 2 repetidores puede colocarse entre 2 máquinas dadas, de esta forma la longitud total de una red Ethernet

[6] Técnicamente el término *octeto* se refiere a un tamaño de carácter dependiente del hardware; los profesionales de redes utilizan *octeto* porque se trata de una cantidad de 8 bits en todas las computadoras.

sigue siendo relativamente corta (3 segmentos de 500 m. cada una). La figura 2.9 muestra un uso común de repetidores en un edificio de oficinas. Un solo cable corre en forma vertical hacia la parte superior del edificio, y se conecta un repetidor a la columna vertebral para derivar cables adicionales hacia cada piso. Las computadoras se conectan a los cables en cada piso.

2.4.10 Extensión de una red Ethernet con puentes

Los puentes son superiores a los repetidores debido a que no reproducen el ruido, los errores o tramas erróneas; una trama completamente válida se debe recibir antes de que el puente la acepte y la transmita hacia otro segmento. Dado que la interfaz de puente sigue las reglas de Ethernet CSMA/CD, las colisiones y los retardos de propagación en un cable se mantienen aislados unos de otros. Como resultado de ello, un (casi) arbitrario número de redes Ethernet se pueden conectar juntas con puentes. Un punto importante es que:

> *Los puentes ocultan los detalles de interconexión: un conjunto de segmentos puenteados actúan como una sola red Ethernet.*

Una computadora utiliza exactamente el mismo hardware para comunicarse con otra computadora a través de un puente que el que utiliza para comunicarse con una computadora en un segmento local.

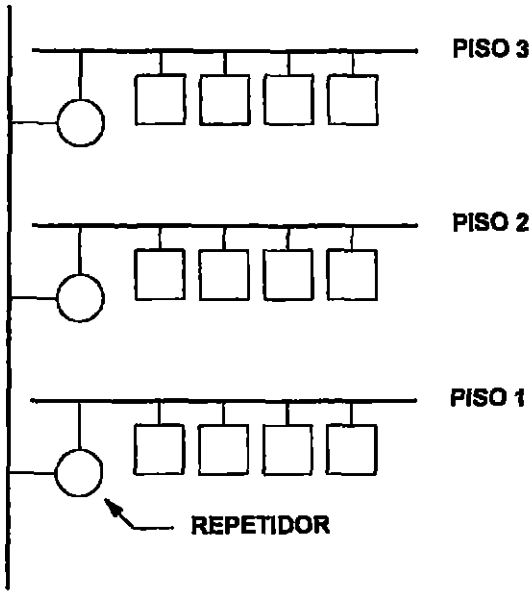


Figura 2.9 Repetidores utilizados para unir cables Ethernet en un edificio. Se pueden colocar un máximo de dos repetidores entre dos máquinas que se comunican.

Varios puentes hacen mucho más que transmitir tramas de un cable a otro: tales puentes son capaces de tomar decisiones inteligentes acerca de qué tramas enviar. A algunos puentes se les conoce como *adaptables* o *puentes con capacidad de aprendizaje*. Un puente adaptable consiste en una computadora con dos interfaces Ethernet. El software en un puente adaptable cuenta con dos listas de direcciones, una para cada interfaz. Cuando una trama llega desde una red Ethernet E_1, el puente adaptable añade la dirección de la *fuente* Ethernet de 48 bits a una lista asociada con E_1. De la misma forma, cuando una trama llega desde una red Ethernet E_2, el puente añade la dirección fuente a una lista asociada con E_2. De esta manera, con el paso del tiempo el puente adaptable irá aprendiendo qué máquinas se pueden direccionar en E_1 y cuáles en E_2.

Luego de grabar la dirección fuente de una trama, el puente adaptable utiliza la dirección de destino para determinar hacia donde debe enviar la trama. Si la lista de direcciones muestra que el destino se localiza en la red Ethernet de la cual proviene la trama, el puente no enviará la trama hacia otra red. Si el destino no está en la lista de direcciones (esto es, si el destino es una dirección de difusión o multidifusión o el puente aún no ha aprendido la localización del destino), el puente enviará la trama hacia otra red Ethernet.

Las ventajas de los puentes adaptables son obvias. Dado que el puente utiliza las direcciones en un tráfico normal, es completamente automático —no se necesita que los humanos configuren el puente con direcciones específicas. Dado que no genera un tráfico innecesario, un puente puede ayudar a mejorar el desempeño de una red sobrecargada aislando el tráfico en segmentos específicos. Los puentes trabajan excepcionalmente bien si una red puede dividirse físicamente en dos segmentos donde cada uno de ellos contenga un conjunto de computadoras que se comunican con frecuencia (por ejemplo, si cada segmento contiene un conjunto de estaciones de trabajo con un servidor y las estaciones de trabajo dirigen la mayor parte del tráfico hacia el servidor). En resumen:

> *Un puente adaptable Ethernet conecta dos segmentos Ethernet, enviando tramas entre uno y otro. Utiliza la dirección fuente para aprender qué máquinas están localizadas en un segmento Ethernet dado y combina la información aprendida con la dirección de destino para eliminar envíos cuando no son necesarios.*

Desde el punto de vista del TCP/IP, los puentes Ethernet no son más que otra forma de conexión física de red. Es importante resaltar lo siguiente:

> *Como la conexión entre cables físicos proporcionada por los puentes y los repetidores es transparente para las máquinas que utilizan la red Ethernet, podemos imaginar los múltiples segmentos Ethernet conectados por puentes y repetidores como un solo sistema físico de red.*

Muchos puentes comerciales son más sofisticados y robustos de lo que se indica en nuestra descripción. Cuando se les inicia, estos puentes verifican la existencia de otros puentes y aprenden la topología de la red. Utilizan un algoritmo de distribución de árbol de extensión para decidir cómo enviar las tramas. En particular, los puentes deciden cómo propagar paquetes de difusión de manera que sólo una copia de la trama de difusión se envíe por cada cable. Sin este algoritmo, las redes Ethernet y los puentes conectados en un ciclo podría producir resultados catastróficos dado que enviarían paquetes de difusión en todas direcciones de manera simultánea.

2.5 Interconexión de datos distribuida por fibra (FDDI)

FDDI es una tecnología de red de área local muy popular que proporciona un ancho de banda mayor que las redes Ethernet. A diferencia de las redes Ethernet y otras tecnologías LAN que utilizan cables para transportar las señales eléctricas, en la tecnología FDDI se utilizan fibras de vidrio y se transfiere la información codificada en pulsos de luz.[7]

La fibra óptica tiene dos ventajas con respecto a los cables de cobre. En primer lugar, como el ruido eléctrico no interfiere con una conexión óptica, la fibra se puede colocar junto a dispositivos eléctricos de potencia. En segundo lugar, dado que las fibras ópticas utilizan luz, la cantidad de datos que pueden enviarse por unidad de tiempo es mucho mayor que en los cables que transportan señales eléctricas.

Podría parecer que las fibras de vidrio son difíciles de instalar y se rompen fácilmente. Sin embargo, un cable óptico posee una flexibilidad sorprendente. La fibra de vidrio por sí misma tiene un diámetro muy pequeño y el cable incluye una cubierta plástica que protege a la fibra de las rupturas. El cable no se puede doblar en un ángulo de 90° pero se puede doblar en un arco con un diámetro de unas cuantas pulgadas. Por lo tanto, su instalación no es difícil.

2.5.1 *Propiedades de una red FDDI*

Una red FDDI es una tecnología token ring a 100 Mbps con una capacidad de auto reparación. Una red FDDI es un *ring (anillo)* dado que la red forma un ciclo que comienza desde una computadora, pasa a través de todas las demás computadoras y termina en el mismo punto en que inició. La FDDI es una tecnología *token ring* porque utiliza un token (o prenda) para controlar la transmisión. Cuando la red está desocupada, una trama especial llamada *token* pasa de una estación a otra. Cuando una estación tiene un paquete para enviar, espera a que llegue el token, envía el paquete y, entonces, transfiere el token a la siguiente estación. La circulación del token garantiza la equidad: asegura que todas las estaciones tengan una oportunidad para enviar un paquete antes de que cualquier estación envíe un segundo paquete.

Tal vez la propiedad más interesante de un FDDI reside en su capacidad para detectar y corregir problemas. La red se conoce como red con capacidad de *autorreparación* ya que el hardware puede adaptarse de manera automática a las fallas.

2.5.2 Anillos dobles de rotación contraria

Para proporcionar una recuperación automática de fallas, el hardware FDDI utiliza dos anillos independientes estando ambos conectados a cada computadora. La figura 2.10 ilustra la topología.

Los anillos FDDI son conocidos como anillos de *rotación contraria* dado que el tráfico circula en direcciones opuestas en cada anillo. La razón para utilizar la rotación en sentidos opuestos se hace clara si consideramos cómo el FDDI maneja las fallas.

---

[7] Una tecnología relacionada, conocida como *Copper Distributed Data Interface* (Interfaz de datos distribuidos por cobre o CDDI por sus siglas en inglés) trabaja como FDDI, pero utiliza cables de cobre para transportar señales.

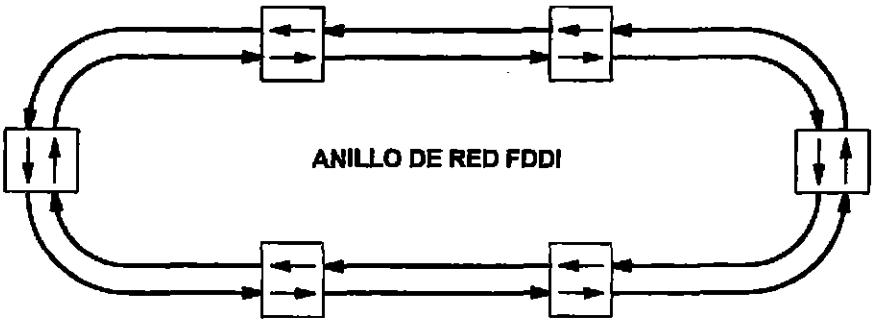


Figura 2.10 Red FDDI que interconecta seis computadoras con fibras ópticas. Las flechas muestran la dirección del tráfico en las fibras y a través de las computadoras conectadas.

A menos que se presente un error, el hardware FDDI no necesita ambos anillos. De hecho una interfaz FDDI se comporta como cualquier interfaz de red que transfiere un token hasta que se presenta un error. La interfaz examina todos los paquetes que circulan en el anillo, comparando la dirección de destino y la dirección de la computadora en cada paquete. La interfaz toma una copia de cualquier paquete destinado a la computadora local, pero también envía el paquete hacia el anillo.

Cuando una computadora necesita transmitir un paquete espera la llegada del token, temporalmente deja de enviar bits, y envía el paquete. Luego de enviar un paquete la interfaz transmite el token y comienza a enviar bits de nuevo. Si una estación tiene más de un paquete listo para ser enviado cuando recibe el token, la estación sólo enviará un paquete antes de pasar el token. Por lo tanto, el esquema del token circulante garantiza que todas las estaciones tengan un acceso franco hacia la red.

El hardware FDDI comienza a ser más interesante cuando ocurre un error de hardware. Cuando una interfaz detecta que no se puede comunicar con la computadora adyacente, la interfaz utiliza el anillo de respaldo para derivar la transmisión y evitar la falla. Por ejemplo, la figura 2.11 muestra un anillo FDDI en el cual una interfaz ha fallado, y las dos interfaces adyacentes la han suprimido del anillo.

El propósito del segundo anillo y la razón por la que la información fluye en dirección opuesta se aclara ahora: una falla puede significar que la fibra ha sido desconectada (por ejemplo, si se cortó accidentalmente). Si la fibra en ambos anillos sigue el mismo trayecto físico, la posibilidad de que la segunda fibra también haya sido desconectada será muy alta. El hardware FDDI de manera automática utiliza el anillo en rotación opuesta para formar un ciclo cerrado en la dirección que aún se mantiene trabajando. Esto permite que las otras computadoras continúen comunicándose a pesar de la falla.

Cuando el hardware FDDI detecta una falla en la red, automáticamente dirige la información hacia el anillo de respaldo para permitir la comunicación entre las estaciones restantes.

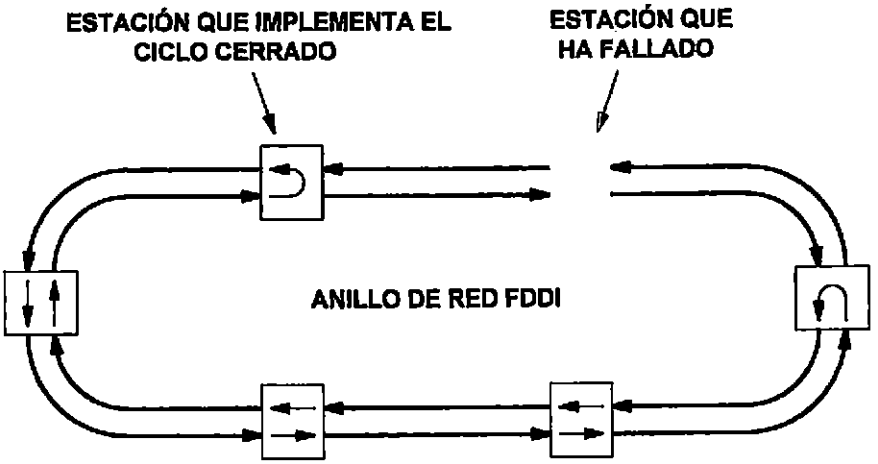


Figura 2.11 Anillo FDDI después de una falla. Cuando el hardware FDDI detecta una falla, utiliza el segundo anillo para derivar el tráfico y permitir que las estaciones restantes se comuniquen.

2.5.3 Formato de la trama de FDDI

Los estándares FDDI especifican el formato exacto de las tramas utilizadas en la red. La tabla en la figura 2.12 muestra una lista de los campos en la trama FDDI.

Campo	Longitud en unidades de 4 bits	Contenido
PA	4 o más	Preámbulo
SD	2	Delimitador de inicio
FC	2	Control de trama
DA	4 o 12	Dirección de destino
SA	4 o 12	Dirección fuente
RI	0 o 60	Información de ruteo
DATA	0 o más	Datos
FCS	8	Verificación de secuencia de trama
DE	1	Delimitador de final
FS	3 o más	Estado de trama

Figura 2.12 Formato de las tramas utilizadas por las redes FDDI, en el que se señala el tamaño de los campos en unidades de cuatro bits, llamadas *symbols*. La máxima longitud de trama es de 9,000 symbols.

Como sucede en otras tecnologías, cada computadora conectada a una red FDDI es asignada a una dirección y cada trama contiene un campo de dirección de destino. Sin embargo, para hacer más flexible a la FDDI y proporcionar una forma estándar de interconexión de dos anillos FDDI, los diseñadores permiten más de un formato de trama. Por ejemplo, el campo de dirección de destino es tanto de 4 como de 12 símbolos de longitud, donde un *símbolo* es una unidad de 4 bits. La trama también incluye un pequeño campo utilizado para el ruteo. El emisor puede emplear el campo de ruteo para especificar que una trama debe enviarse primero a un punto de conexión y después a un destino en un anillo conectado.

Una de las ventajas del FDDI reside en el gran tamaño de trama. Debido a que una trama puede contener 9,000 símbolos de 4 bits, el total de la trama puede ser de 4,500 octetos de longitud. Dado que el encabezado de información ocupa cuando mucho unos pocos cientos de octetos, una trama sola puede transportar 4K octetos de datos de usuario. En aplicaciones que transfieren grandes volúmenes de datos (por ejemplo, en la transferencia de archivos), la gran longitud de la trama significa una menor sobrecarga y consecuentemente un alto rendimiento efectivo total.

2.6 Modalidad de transferencia asíncrona

Asynchronous Transfer Mode (*modo de transferencia asíncrono* o *ATM*, por sus siglas en inglés) es el nombre dado a una tecnología de red orientada a la conexión de alta velocidad, que ha sido utilizada tanto en redes de área local como en redes de área amplia. Para los estándares actuales, alta velocidad se refiere a las redes que operan a 100 Mbps o más; el ATM puede conmutar datos a velocidades en gigabits.[8] Por supuesto, cada red de alta velocidad requiere de equipo complejo y de vanguardia. Como resultado de ello, las redes ATM son más caras que las de otras tecnologías.

Para obtener una velocidad de transferencia alta, una red ATM utiliza técnicas de software y hardware de propósito especial. Primero, una red ATM consiste en uno o más conmutadores de alta velocidad que se conectan con cada computadora anfitrión y con los otros conmutadores ATM. Segundo, la tecnología ATM utiliza fibra óptica para las conexiones, incluyendo las conexiones de computadoras anfitrión hacia los conmutadores ATM. La fibra óptica proporciona una razón de transferencia alta, mayor que la de los alambres de cobre; por lo común, la conexión entre un anfitrión y un conmutador ATM opera entre los 100 y los 155 Mbps. Tercero, las capas más bajas de una red ATM utilizan tramas de tamaño fijo llamadas *cells (celdas)*. Dado que cada celda es exactamente del mismo tamaño, el hardware de conmutador ATM puede procesar las celdas con rapidez.

2.6.1 Tamaño de las celdas ATM

Sorprendentemente, cada celda ATM tiene sólo 53 octetos de largo. La celda contiene 5 octetos de encabezado, seguido por 48 octetos de datos. En capítulos posteriores veremos, sin embargo, que cuando se utiliza el ATM para hacer envíos en el tráfico del IP, el tamaño de 53 octetos es irrelevante —una red ATM acepta y envía paquetes mucho más largos.

[8] Un gigabits por segundo (Gbps) es igual a 1000 millones de bits por segundo. Muchas computadoras no pueden generar o recibir datos a esta velocidad; un conmutador ATM opera en velocidades de gigabits para sostener el tráfico generado por muchas computadoras.

2.6.2 Redes orientadas a la conexión

ATM difiere de las redes de conmutación de paquetes descritas al principio debido a que ofrece un servicio *connection oriented* (*orientado a la conexión*). Antes de que una computadora anfitrión conectada a un ATM pueda enviar celdas, el anfitrión debe interactuar primero con el conmutador para especificar un destino. La interacción es análoga a la que se realiza en una llamada telefónica.[9] El anfitrión especifica la dirección de la computadora remota, y espera a que el conmutador ATM contacte el sistema remoto y establezca una ruta. Si la computadora remota rechaza la solicitud, no responde o el conmutador ATM no puede llegar a la computadora remota, la solicitud para establecer la comunicación no tendrá éxito.

Cuando una conexión se establece con éxito, el conmutador ATM local selecciona un identificador para la conexión y transfiere el identificador de conexión al anfitrión con un mensaje que informa al anfitrión del éxito de la comunicación. El anfitrión utiliza el identificador de conexión cuando envía o recibe celdas.

Cuando se termina de usar la conexión, el anfitrión se comunica nuevamente con el conmutador ATM para solicitar que la conexión se interrumpa. El conmutador desconecta las dos computadoras. La desconexión es equivalente a "colgar" en una llamada telefónica al terminar la llamada; después de la desconexión, el conmutador puede reutilizar el identificador de conexión.

2.7 Tecnología ARPANET

Una de las primeras redes de conmutación de paquetes de área amplia, ARPANET, fue construida por ARPA, la Advanced Research Projects Agency. ARPA otorgó un contrato para el desarrollo del software ARPANET a Bolt, Beranek y Newman de Cambridge, MA. hacia fines de 1968. En septiembre de 1969, las primeras piezas de ARPANET habían sido colocadas en su lugar.

ARPANET sirvió como campo de prueba para muchas de las investigaciones sobre conmutación de paquetes. Además de utilizarla como una red de investigación, los investigadores en varias universidades, bases militares y laboratorios gubernamentales, utilizaban con regularidad ARPANET para intercambiar archivos y correo electrónico y para proporcionar una conexión remota entre estos sitios. En 1975, el control de la red se transfirió de ARPA a la U.S. Defense Communications Agency (DCA). La DCA hizo que ARPANET fuera parte de la Defense Data Network (Red de Datos de la Defensa o DDN, por sus siglas en inglés), un programa que proporciona redes múltiples como parte de un sistema de comunicación alrededor del mundo para el Departamento de Defensa.

En 1983 el Departamento de Defensa dividió ARPANET en dos redes conectadas, dejando ARPANET para investigaciones experimentales y formando la *MILNET* para usos militares. MILNET está restringida al manejo de datos no clasificados. Aun cuando bajo condiciones normales, tanto ARPANET como MILNET mantienen un tráfico entre una y otra, el control se ha establecido para permitir que éstas se puedan desconectar.[10] Dado que ARPANET y MILNET utilizan la misma

[9] Debido a que ATM fue diseñada para transportar voz, así como datos, existe una relación muy fuerte entre ATM y los conmutadores telefónicos.

[10] Es posible que el mejor ejemplo conocido de desconexión ocurrió en noviembre de 1988, cuando un programa *worm* (*write-once read-many, o escribir-una vez leer-muchas*) atacó Internet y se replegó a sí mismo tan rápido como fue posible.

tecnología de hardware, nuestra descripción de los detalles técnicos se aplica a ambas, de manera que nos referiremos únicamente a ARPANET. De hecho, la tecnología está disponible comercialmente y es utilizada por varias corporaciones para establecer redes privadas de conmutación de paquetes.

Dado que ARPANET se había ya instalado y muchos de los investigadores que trabajaban en la arquitectura de Internet la utilizaban diariamente, tuvo una influencia profunda en su trabajo. Ellos pensaban en ARPANET como en una red de columna vertebral de área amplia, confiable y segura alrededor de la cual Internet podría construirse. La influencia de una sola red de columna vertebral de área amplia es todavía penosamente obvia en algunos de los protocolos de Internet que trataremos más adelante, y éstos han obstaculizado que Internet pueda adaptarse con facilidad a redes adicionales de columna vertebral.

Físicamente, ARPANET consiste en aproximadamente 50 minicomputadoras C30 y C300 de la BBN Corporation, llamadas *Packet Switching Nodes* (*nodos de conmutación de paquetes* o *PSN*, por sus siglas en inglés)[11] distribuidas en el territorio continental de Estados Unidos y de Europa Occidental (MILNET contiene unas 160 PSN, incluyendo 34 en Europa y 18 en el Pacífico y en el Lejano Oriente). Una PSN se ubica en cada localidad que participa en la red y está dedicada a la tarea de la conmutación de paquetes. La PSN no pueden utilizarse para computación de propósito general. De hecho, la PSN fue considerada parte de ARPANET y era propiedad de *Network Operations Center (NOC)*, localizada en la BBN en Cambridge, Massachusetts.

Desde las compañías de telecomunicaciones se conectaban los circuitos arrendados de datos de tipo punto a punto junto con las PSN para formar una red. Por ejemplo, los circuitos arrendados de datos conectaban la PSN de ARPANET de la Universidad de Purdue con la PSN de ARPANET en Carnegie Mellon y la Universidad de Wisconsin. Al principio, muchos de los circuitos arrendados de datos en ARPANET operaban a 56 Kbps, una velocidad considerada extremadamente alta en 1968 pero baja para los estándares actuales. Debemos recordar que la velocidad es una medida de la capacidad más que una medida del tiempo que toma el envío de paquetes. Conforme más computadoras utilizaban ARPANET, la capacidad se fue incrementando para adaptarse a la carga. Por ejemplo, durante el último año de existencia de ARPANET, muchos de los enlaces entre países operaron con canales que trabajaban en millones de bits por segundo.

La idea de no contar con un solo recurso, vulnerable a las fallas del sistema, es común en las aplicaciones militares dado que en éstas la confiabilidad es importante. Cuando se construyó ARPANET, ARPA decidió seguir los requerimientos militares en cuanto a confiabilidad, por lo tanto cada PSN debía tener al menos dos líneas de conexión arrendadas hacia otras PSN, y el software debía adaptarse automáticamente a las fallas y seleccionar rutas alternativas. Como resultado, ARPANET continuaba operando incluso si uno de los circuitos de datos fallaba.

Además, para la conexión con los circuitos arrendados de datos, cada PSN de ARPANET tenía hasta 22 *puertos* que la conectaba con las computadoras de los usuarios, llamadas *hosts*. Originalmente, todas las computadoras que accesaban ARPANET se conectaban de manera directa con uno de los puertos en una PSN. Por lo general, las conexiones directas eran formadas con una tarjeta de interfaz de propósito especial que se conectaba dentro del bus de entrada y salida de la computadora y se conectaba con un puerto anfitrión PSN. Si se programaba en forma adecuada, la interfaz permitía a la computadora ponerse en contacto con la PSN para enviar y recibir paquetes.

[11] Los PSN fueron llamados inicialmente *Interface Message Processors (IMP)*; algunas publicaciones todavía utilizan el término IMP como sinónimo de conmutación de paquetes.

El hardware de puerto original de PSN utilizaba un protocolo complejo para transferir datos a través de ARPANET. Conocido como 1822, debido al número del reporte técnico que lo describía, el protocolo permitía a un huésped enviar un paquete a través de ARPANET hacia un destino específico de PSN y un puerto específico en PSN. Realizar la transferencia es complicado, sin embargo, el 1822 es confiable debido a que realiza la transmisión con un control de flujo. Para prevenir que un anfitrión dado saturara la red, el 1822 limita el número de paquetes que pueden encontrarse en tránsito. Para garantizar que cada paquete llegue a su destino, el 1822 hace que el emisor espere una señal *Ready for Next Message (RFNM)* desde la PSN antes de transmitir cada paquete. La RFNM actúa como un acuse de recibo. Éste incluye un esquema de reservación de búfer que requiere el emisor para reservar un búfer en el destino PSN antes de enviar un paquete.

A pesar de que hay muchos aspectos del 1822 no discutidos aquí, la idea que hay que comprender es que independientemente de todos los detalles, ARPANET era sólo un mecanismo de transferencia. Cuando una computadora conectada a un puerto envía un paquete a otro puerto, el dato transmitido es exactamente el dato enviado. Dado que ARPANET no proporciona un encabezado específico de red, los paquetes se envían a través de ella sin contar con un campo específico donde se determine el tipo de paquete. De esta forma, a diferencia de algunas otras tecnologías de red, ARPANET no transmite paquetes que se autoidentifiquen.

En resumen:

> *Las redes como ARPANET o ATM no tienen tramas que se autoidentifiquen. Las computadoras conectadas deben ponerse de acuerdo en cuanto al formato y el contenido de los paquetes que enviarán a o recibirán de un destino específico.*

Por desgracia, el 1822 nunca fue un estándar de la industria. Como muy pocos fabricantes vendieron tarjetas de interfaz 1822 es muy difícil conectar máquinas nuevas a ARPANET. Para resolver el problema, ARPA desarrolló una nueva interfaz PSN que utiliza un estándar de comunicaciones de datos internacional conocido como *CCITT X.25* (el nombre le fue asignado por el comité de estándares que lo desarrolló). La primera versión de una implementación X.25 de PSN utilizaba sólo la parte de transferencia de datos del estándar X.25 (conocida como HDLC/LAPB), pero en versiones posteriores se hizo posible utilizar todo el X.25 cuando se conectaba a una PNS (por ejemplo, ARPANET parecía ser una red X.25). Muchos puertos MILNET utilizan ahora el X.25.

Internamente, por supuesto, ARPANET utiliza su propio conjunto de protocolos que son invisibles para el usuario. Por ejemplo, había un protocolo especial que permitía a una PSN solicitar el estatus a otra, otro protocolo utilizaba una PSN para enviar paquetes entre ellas mismas, y otro permitía a las PSN intercambiar información acerca del estado de enlace y de optimización de rutas.

Dado que ARPANET fue en sus orígenes construida como una sola red independiente para ser utilizada en la investigación, la estructura de sus protocolos y direccionamientos fue diseñada sin pensar mucho en la expansión. A mediados de la década de los setenta comenzó a ser evidente que una sola red no resolvería todos los problemas de comunicación y ARPA comenzó a investigar tecnologías de red que transmitían paquetes por radio y vía satélite. Esta experiencia con una diversidad de tecnologías de red llevó al concepto de un enlace entre redes.

En la actualidad, ARPANET está desapareciendo en forma silenciosa y está siendo reemplazada por nuevas tecnologías. MILNET continúa como el lado militar de la conexión a Internet.

2.7.1 Direccionamiento ARPANET

Los detalles del direccionamiento en ARPANET no son importantes, sin embargo este direccionamiento ilustra una forma alternativa en la que las redes de área amplia forman las direcciones físicas. A diferencia de las redes de área local, como Ethernet, las redes de área amplia por lo general incorporan información en la dirección que ayuda a la red a dirigir los paquetes hacia su destino con eficiencia. En la tecnología ARPANET, cada conmutador de paquetes es asignado a un único entero, *P*, y cada puerto anfitrión en el conmutador es numerado de *0* a *N-1*. Conceptualmente, una dirección de destino consiste en un pequeño par de enteros, *(P, N)*. En la práctica, el hardware utiliza un sola dirección en forma de número entero grande, valiéndose de algunos bits de la dirección para representar *N* y otros para representar *P*.

2.8 Red de la Fundación Nacional de Ciencias

Llevar a cabo la comunicación de datos fue algo crucial para la investigación científica, en 1987 la Fundación Nacional de Ciencias estableció la *División of Network and Communications Research and Infrastructure* para ayudar a que los requisitos de las comunicaciones por red aseguraran que éstas estuvieran disponibles para los científicos y los ingenieros de los Estados Unidos. A pesar de las diferencias encontradas en la investigación básica de redes, se hizo énfasis al concentrar los esfuerzos para crear las bases en la construcción de extensiones hacia Internet.

Las extensiones NSF de Internet desde una jerarquía de tercer nivel consistían en una red de columna vertebral en Estados Unidos, un conjunto de redes de "nivel medio" y "regionales" donde cada una abarcaba una pequeña área geográfica y un conjunto de "campus" o redes de "acceso". En el modelo NSF, las redes de nivel medio se conectaban a la red de columna vertebral y las redes de campus a las redes de nivel medio. Los investigadores tenían una conexión de su computadora a la red local del campus. Podían utilizar esta conexión para comunicarse con las computadoras de los investigadores locales a través de la red local del campus, también podían hacerlo con investigadores del exterior debido a que sus máquinas podían dirigir información hacia la red local y a través de la red de nivel medio hasta la red de columna vertebral, conforme fuera necesario.

2.8.1 La red de columna vertebral original NSFNET

De todas las redes NSF fundadas, la red de columna vertebral NSFNET contó con la historia y utilizó la tecnología más interesante. Hasta la fecha, la red de columna vertebral se ha desarrollado en cuatro etapas mayores; se incrementó en tamaño y en capacidad, al mismo tiempo que ARPANET declinaba, hasta convertirse en la red de columna vertebral dominante en Internet. Una de las primeras justificaciones para la construcción de redes de columna vertebral fue que proporcionaban a los científicos acceso a las supercomputadoras NSF. Como resultado, la primera red de columna vertebral consistió en 6 microcomputadoras LSI-11 de la Digital Equipment Corporation, localizadas en el centro de supercómputo NSF. Geográficamente, la red de columna vertebral abarcaba el territorio continental de Estados Unidos desde Princeton, NJ hasta San Diego, CA, y utilizaba líneas arrendadas de 56 Kbps como se muestra en la figura 2.13.

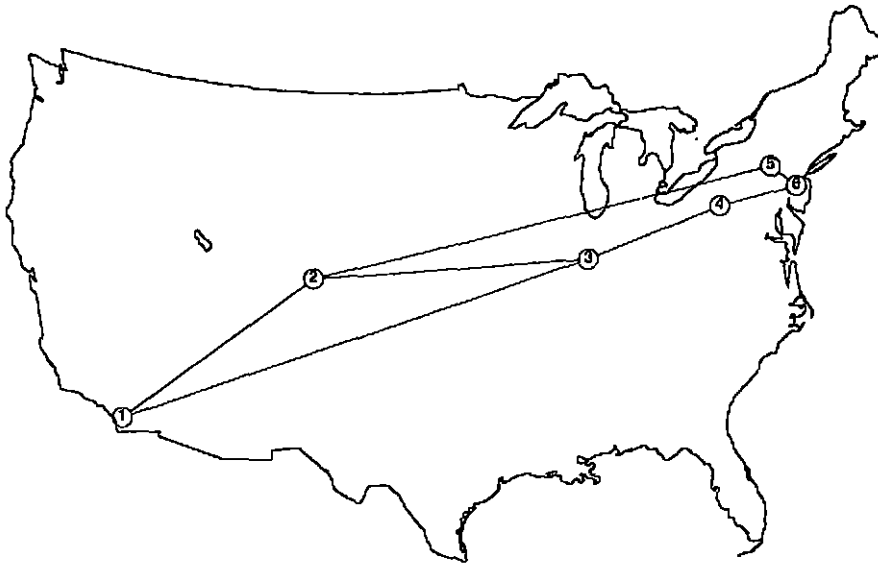


Figura 2.13 Circuitos en la red de columna vertebral NSFNET original con localidades en (1) San Diego CA, (2) Boulder CO, (3) Champaign IL, (4) Pittsburg PA, (5) Ithaca NY y (6) Princeton NJ.

En cada localidad, la microcomputadora *LSI-11* corría un software conocido como código *fuzzball*.[12] Desarrollado por Dave Mills, cada fuzzball accesaba computadoras en el centro de supercómputo local utilizando una interfaz Ethernet convencional. El acceso hacia otras líneas rentadas se dirigía primero al fuzzball en otros centros de supercómputo por medio de protocolos de nivel de enlace convencionales sobre las líneas seriales arrendadas. Los fuzzball contenían tablas con las direcciones de destinos posibles y las utilizaban para dirigir cada paquete en trámite hacia su destino.

La conexión primaria entre la red de columna vertebral original NSFNET y el resto de Internet estaba localizado en Carnegie Mellon, la cual tenía tanto un nodo de la red de columna vertebral NSFNET como una ARPANET PSN. Cuando un usuario, conectado con la NSFNET, enviaba tráfico de información hacia una localidad en ARPANET, los paquetes debían viajar a través de NSFNET hacia CMU donde el fuzzball lo ruteaba hacia ARPANET vía el Ethernet local. De la misma forma, el fuzzball entendía qué paquetes destinados a las localidades NSFNET debían ser aceptados desde la red Ethernet y cuáles enviados a través de la red de columna vertebral NSF hacia la localidad apropiada.

2.8.2 La segunda red de columna vertebral NSFNET 1988-1989

Aun cuando los usuarios estaban entusiasmados con las posibilidades de la comunicación de computadoras, la capacidad de transmisión y conmutación de la red de columna vertebral original era

[12] El origen exacto del término "fuzzball" (textualmente "bola de peluza"), no es claro.

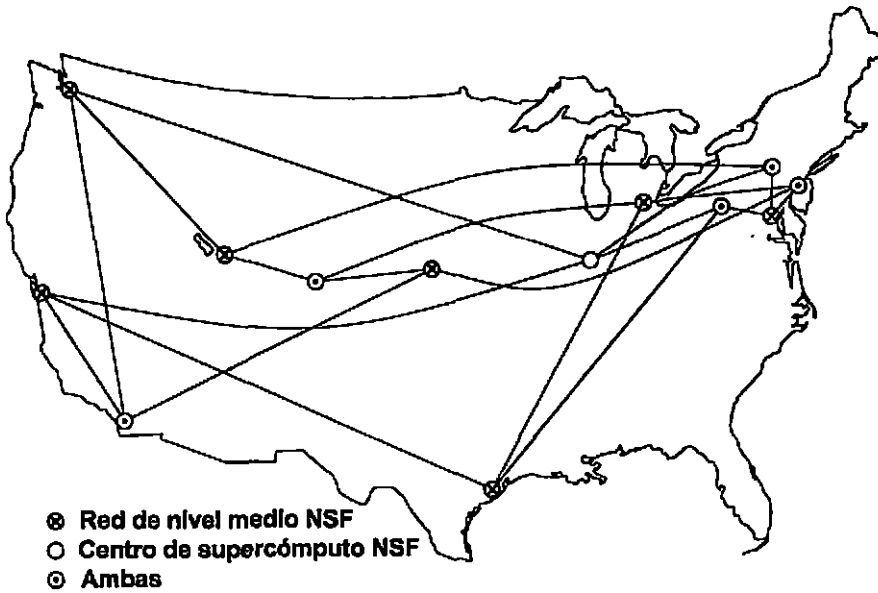


Figura 2.14 Circuitos lógicos en la segunda red de columna vertebral NSFNET del verano de 1988 al de 1989.

demasiado pequeña para proporcionar un servicio adecuado. Pocos meses después de su instalación, la red de columna vertebral fue sobrepasada y sus creadores trabajaron en implementar rápidamente alguna solución para los problemas más apremiantes mientras que la NSF comenzaba el largo proceso de planificar una segunda red de columna vertebral.

En 1987, la NSF publicó una solicitud de propuestas de parte de grupos que estuvieran interesados en establecer y operar una nueva red de columna vertebral de alta velocidad. Las propuestas fueron presentadas en agosto de 1987, y evaluadas hacia finales de ese mismo año. El 24 de noviembre de 1987 la NSF anunció que había seleccionado una propuesta presentada por una sociedad, ésta estaba formada por la MERIT Inc., la red de computadoras estatal que corría fuera de la Universidad de Michigan, en Ann Arbor, la IBM Corporation y la MCI Incorporated. La sociedad proponía la construcción de una segunda red de columna vertebral, establecer un centro de control y operación de red en Ann Arbor y tener el sistema ya operando para el próximo verano. Debido a que la NSF había fundado varias redes de nivel medio, la red de columna vertebral propuesta estaba planeada para servir a más localidades que la original. Cada localidad adicional proporcionaría una conexión entre la red de columna vertebral y una de las redes de nivel medio de la NSF.

La forma más fácil de imaginar la división de trabajo entre los tres grupos es asumiendo que MERIT estaba a cargo de la planeación, establecimiento y la operación del centro de la red. La IBM contribuiría con máquinas y mano de obra calificada de sus laboratorios de investigación para auxiliar a MERIT en el desarrollo, la configuración y las pruebas necesarias para el hardware y el software. La MCI, una compañía de comunicaciones de larga distancia, proporcionaría el ancho de banda de

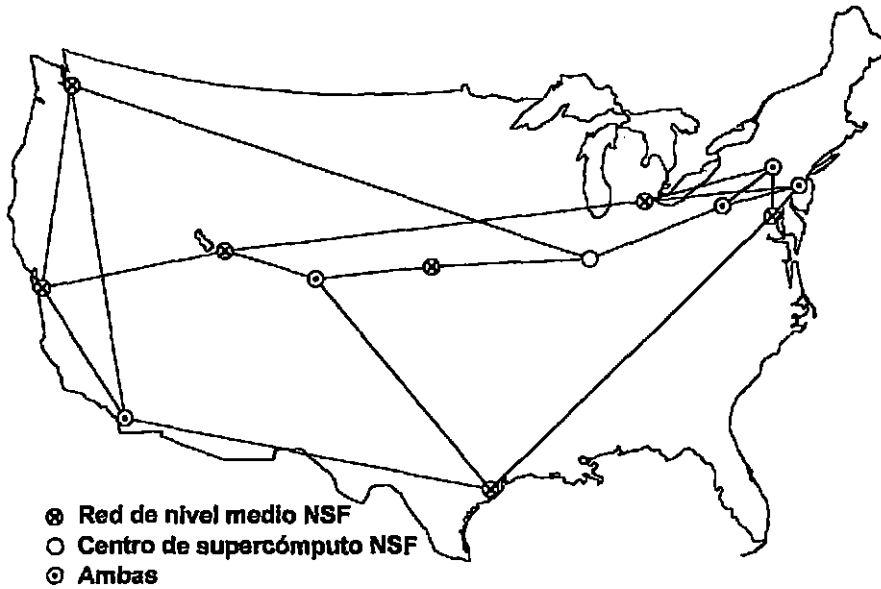


Figura 2.15 Circuitos en la segunda red de columna vertebral NSFNET del verano de 1989 al de 1990.

comunicación mediante el uso de fibra óptica ya colocada para sus redes de voz. Por supuesto, en la práctica había una colaboración más cercana entre todos los grupos, incluyendo proyectos de estudio conjuntos y representaciones de IBM y MCI en la dirección del proyecto.

Hacia mediados del verano de 1988, el hardware había ocupado su lugar y la NSFNET comenzó a utilizar la segunda red de columna vertebral. Poco tiempo después, la red de columna vertebral original fue apagada y desconectada. La figura 2.14 muestra la topología lógica de la segunda red de columna vertebral luego de que fue instalada en 1988.

La tecnología seleccionada para la segunda red de columna vertebral NSFNET es interesante. En esencia, la red de columna vertebral era una red de área amplia compuesta por ruteadores de paquetes interconectados por líneas de comunicación. Como con la red de columna vertebral original, el conmutador de paquetes en cada localidad se conectaba con la red Ethernet local, así como con las líneas de comunicación principales que se dirigían a otras localidades.

2.8.3 Red de columna vertebral NSFNET 1989-1990

Luego de aliviar el tráfico en la segunda red de columna vertebral NSFNET por un año, el centro de operaciones reconfiguró la red al añadir unos circuitos y suprimir otros. Además se incrementó la velocidad de los circuitos de ES-1 (1.544 Mbps). La figura 2.15 muestra la topología de conexión revisada, la cual proporcionaba conexiones redundantes para todas las localidades.

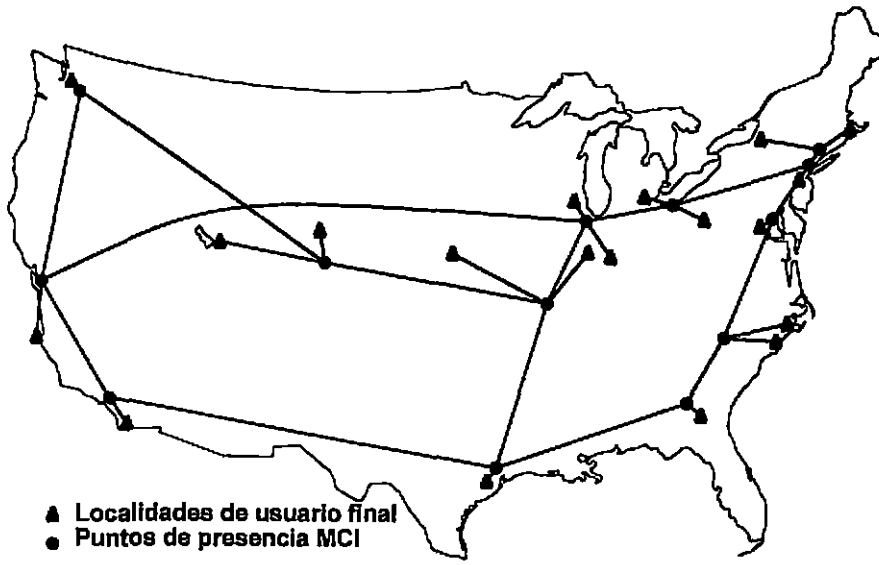


Figura 2.16 Circuitos en ANSNET, la red de columna vertebral de Internet en los Estados Unidos que se inició en 1993. Cada circuito opera a 45 Mbps.

2.9 ANSNET

Hacia 1991, la NSF y otras dependencias gubernamentales comenzaron la ampliación de Internet más allá del dominio académico y científico original. Muchas compañías alrededor del mundo comenzaban a conectarse con Internet y el número de usuarios que no se dedicaba a la investigación se incrementó de manera súbita. El tráfico en NSFNET había crecido a cerca de un billón de paquetes por día y la capacidad de 1.5 Mbps comenzaba a ser insuficiente en el caso de varios de los circuitos. Se hizo necesaria una capacidad mayor para la red de columna vertebral. Como resultado de ello el gobierno de Estados Unidos implementó una política de comercialización y privatización. La NSF decidió transferir la red de columna vertebral a una compañía privada y comenzar a cobrar a las instituciones por la conexión.

En respuesta a la nueva política gubernamental, en diciembre de 1991, IBM, MERIT y MCI formaron una compañía sin fines de lucro llamada *Advanced Networks and Services (ANS)*. ANS propuso la construcción de una nueva red de columna vertebral de Internet de alta velocidad. A diferencia de las anteriores redes de área amplia utilizadas en Internet, las cuales habían sido propiedad del gobierno de los Estados Unidos, ANS sería propietaria de la nueva red de columna vertebral. Hacia 1993, ANS había instalado una nueva red que reemplazaba a la NSFNET. Llamada *ANSNET* la nueva red de columna vertebral opera a 45 Mbps[13], y alcanza una capacidad de unas 30 veces la

[13] Las compañías de telecomunicaciones utilizan el término *DS3* para referirse a un circuito que opera a 45 Mbps; el término se confunde con *T3* frecuentemente, porque éste denota una codificación específica utilizada en un circuito que opera a velocidad DS3.

capacidad de la red de columna vertebral anterior NSFNET. La figura 2.16 muestra los circuitos mayores en ANSNET y algunas de las localidades conectadas en 1994. Cada punto presenta una ubicación a la que se han conectado muchas localidades.

2.10 Una red de columna vertebral de área amplia planeada

La NSF ha concesionado a MCI un contrato para construir una red de columna vertebral de 155 Mbps para reemplazar a ANSNET. Llamada *very high speed Backbone Network Service (vBNS)*, la nueva red de columna vertebral ofrecerá un incremento substancial en la capacidad y requerirá de procesadores de alta velocidad para el ruteo de paquetes.

2.11 Otras tecnologías en las que se ha utilizado el TCP/IP

Una de las mayores cualidades del TCP/IP radica en la variedad de tecnologías de red física sobre las que se puede utilizar. Hemos analizado varias tecnologías ampliamente utilizadas, incluyendo redes de área local y de área amplia. En esta sección se revisa de manera breve otros puntos que ayudan a entender un principio importante:

> *Muchos de los éxitos del protocolo TCP/IP radican en su capacidad para adaptarse a casi cualquier tecnología de comunicación subyacente.*

2.11.1 X25NET

En 1980, la NSF formó la organización CSNET para ayudar a proporcionar servicios de Internet a la industria y a pequeñas escuelas. La CSNET utiliza varias tecnologías para conectar a los suscriptores con Internet, incluyendo una llamada *X25NET*. Originalmente desarrollada en la universidad Purdue, la X25NET corre protocolos TCP/IP en *Public Data Networks (PDN)*. La motivación para construirla como una red se originaba en la economía de las telecomunicaciones: a pesar de que las líneas seriales arrendadas eran caras, las compañías de telecomunicaciones habían comenzado a ofrecer al público servicios de conmutación de paquetes. La X25NET fue diseñada para permitir a una localidad el uso de sus conexiones para un servicio de conmutación de paquetes y enviar y recibir tráfico de Internet.

Los lectores que conozcan acerca de las redes de conmutación de paquetes pueden encontrar extraña la X25NET debido a que los servicios públicos utilizan el protocolo X.25 de CCITT[14] exclusivamente mientras que Internet emplea protocolos TCP/IP. Cuando se usa el TCP/IP para transportar tráfico, sin embargo, la red X.25 subyacente sólo proporciona una ruta sobre la cual el tráfico de Internet puede transferirse. Hemos indicado que muchas tecnologías subyacentes pueden

[14] El grupo responsable de las redes de datos en el Consultative Committee for International Telephone and Telegraph ha dado origen a la Telecommunication Section de la International Telecommunication Union (ITU-TS).

ser empleadas para acarrear tráfico de Internet. Esta técnica, a veces llamada *tunneling*, tan sólo significa que el TCP/IP trata a un sistema de red complejo con sus propios protocolos como cualquier otro hardware de sistema de transmisión. Para enviar tráficos de información del TCP/IP a través de un *túnel* X.25, se hace una conexión con X.25 y entonces se envían paquetes TCP/IP como si éstos fueran datos. El sistema X.25 transportará paquetes a lo largo de la conexión y los entregará al otro extremo X.25, donde éstos serán seleccionados y enviados hacia su destino final. Debido a que el proceso mediante túneles trata a los paquetes como datos, no proporciona tramas autoidentificables. Así, sólo trabaja cuando ambos extremos de la conexión X.25 acuerdan de antemano que intercambiarán paquetes TCP/IP.

Lo que hace peculiar el uso de X.25 es su interfaz. A diferencia del hardware de muchas otras redes, el protocolo X.25 proporciona una transmisión de flujo confiable, a veces llamada *circuito virtual*, entre el emisor y el receptor, mientras que los protocolos de Internet han sido diseñados para sistemas de transferencia de paquetes. Esto podría hacer que los dos (aparentemente) sean incompatibles.

La consideración de la conexión X.25 simplemente como una ruta de transferencia produce un giro inesperado. El resultado es que las redes X.25 muestran una mejoría sustancial en su desempeño con conexiones múltiples simultáneas. Esto es, en lugar de abrir una sola conexión para un destino, un emisor X.25NET por lo general abre conexiones múltiples y distribuye paquetes entre éstas para mejorar su desempeño. El receptor acepta los paquetes de todas las conexiones X.25 y los reúne de nuevo.

El esquema de direccionamiento utilizado por las redes X.25 se describe en un estándar conocido como X.121. Cada dirección física X.121 consiste de un número de 14 dígitos, con 10 dígitos asignados por el vendedor que proporciona el servicio de red X.25. Recordando los números telefónicos, un vendedor muy popular estableció las asignaciones incluyendo un código de área basado en la zona geográfica. Este esquema de direccionamiento no sorprende porque proviene de una organización que determina los estándares telefónicos internacionales. Sin embargo, resulta desafortunado dado que dificulta el direccionamiento en Internet. Los suscriptores que utilizan X25NET deben mantener una tabla de transformaciones entre las direcciones de Internet y las direcciones X.25. En el capítulo 5, se trata en detalle el problema de la transformación de direcciones y se presenta una alternativa para el uso de tablas fijas. El capítulo 18 muestra cómo se presenta el mismo problema en las redes ATM, las cuales ya utilizan otra alternativa.

Debido a que la red pública X.25 opera de manera independiente a Internet, se debe proporcionar un punto de contacto entre las dos. Tanto en ARPA como en CSNET operan máquinas dedicadas que proporcionan la interconexión entre X.25 y ARPANET. La primera interconexión se conoció como *VAN gateway*. La VAN acepta conexiones X.25 y por medio de una conexión rutea cada datagrama que llega hacia su destino.

La X25NET es significativa debido a que ilustra la flexibilidad y adaptabilidad de los protocolos TCP/IP. En particular, muestra cómo al hacer túneles, es posible utilizar un rango extremadamente grande de complejas tecnologías en una red de redes.

2.11.2 Marcación IP

Otro uso interesante del TCP/IP iniciado por CSNET corre protocolos TCP/IP en una red de voz de marcación (esto es, el sistema telefónico). Las localidades miembros de CSNET que utilizan Internet con poca frecuencia pueden encontrar injustificable el costo de una línea de conexión arrenda-

da. Para este tipo de localidades, CSNET desarrolló un sistema de marcación que trabaja como sería de esperarse: cada vez que la conexión es necesaria, el software de una localidad miembro utiliza un módem para establecer una conexión hacia un concentrador CSNET a través de una red telefónica de voz. Una computadora en el concentrador responde a la llamada telefónica, luego de obtener una autorización válida, y comienza a enviar tráfico de información entre la localidad y las otras computadoras en Internet. La llamada introduce un retardo luego de que el primer paquete se ha enviado. Sin embargo, en servicios automatizados, como el correo electrónico, el retardo es imperceptible.

2.11.3 Otras tecnologías Token Ring

La FDDI no es la primera tecnología de red de tipo token ring, los productos token ring han existido por varias décadas. Por ejemplo, IBM produce una tecnología LAN token ring utilizada en localidades que cuentan con equipo IBM. El token ring de IBM opera a 16 Mbps; la versión original operaba a 4 Mbps. Como en el caso de otros sistemas token ring, una red token ring de IBM consiste en un ciclo cerrado que se comunica con todas las computadoras. Una estación debe esperar el token (prenda) antes de transmitir y enviar el token luego de haber transferido un paquete.

Una tecnología token ring, diseñada por la compañía Proteon, empleaba un novedoso esquema de direccionamiento al que nos referimos en capítulos posteriores para ilustrar uno de los tipos de direcciones de hardware que utilizan el TCP/IP. La tecnología se conocía como red *proNET* y permitía a los usuarios seleccionar una dirección de hardware para cada computadora. A diferencia de las redes Ethernet en las que cada tarjeta de interfaz contenía una sola dirección asignada por el fabricante, una tarjeta de interfaz proNET contiene ocho interruptores que pueden ser configurados antes de que la interfaz se instale en la computadora. Los interruptores forman un número en lenguaje binario del 0 al 255, inclusive. Una red proNET dada puede tener un máximo de 254 computadoras conectadas ya que la dirección 255 se reserva para difusión y la dirección 0, por lo general, no se utiliza. Cuando se instala por primera vez una red proNET, un administrador de red selecciona una dirección única para cada computadora. Por lo regular, las direcciones se asignan en forma secuencial comenzando por la *1*.

Una tecnología que permite que el usuario asigne las direcciones de hardware tiene ventajas y desventajas. La gran desventaja es el problema potencial que puede presentarse si el administrador de red accidentalmente asigna la misma dirección a dos máquinas. La gran ventaja radica en la facilidad de mantenimiento: si una tarjeta de interfaz falla, puede reemplazarse sin cambiar las direcciones de hardware de las computadoras.

2.11.4 Transmisión de paquetes por radio

Uno de los experimentos de ARPA más interesantes en conmutación de paquetes condujo a una tecnología que utiliza ondas de radio de difusión a transferir paquetes. Diseñada para un ambiente militar, en el cual las estaciones están en movimiento, la transmisión de paquetes por radio incluye un hardware y un software que permite a las localidades encontrar otras localidades, establecer una comunicación punto a punto y luego utilizar la comunicación punto a punto para transferir paquetes. Dado que las localidades cambian de ubicación geográfica y pueden salir del rango de comunicación, el sistema debe monitorear constantemente la conectividad y recomputar las rutas para que éstas reflejen los cambios en la topología. Un sistema de transmisión de paquetes por radio se cons-

truyó y utilizó para demostrar la comunicación TCP/IP entre localidades de transmisión de paquetes por radio y otras localidades en Internet.

Hace poco, algunos vendedores comenzaron a distribuir equipo de red inalámbrico que emplea una técnica de espectro extendido, la cual, como una secuencia directa o un salto de frecuencias, proporciona la conexión en una red inalámbrica. El equipo de comunicaciones inalámbrico es pequeño y ligero. Puede conectarse con facilidad a una notebook portátil, lo que permite continuar la comunicación alrededor de un área como, por ejemplo, un edificio de oficinas.

Con frecuencia el equipo de red inalámbrico simula una red convencional de conmutación de paquetes. Por ejemplo, un vendedor de equipo inalámbrico envía y recibe tramas utilizando el mismo formato que una red Ethernet. De hecho, el hardware ha sido construido para emular exactamente una interfaz de las redes Ethernet. De esta forma, se puede utilizar un protocolo estándar para comunicar redes inalámbricas como si fueran redes Ethernet.

2.12 Resumen y conclusión

Hemos revisado varias tecnologías de hardware de red utilizadas por los protocolos TCP/IP, abarcando desde redes de área local de alta velocidad, como las redes Ethernet, hasta redes de gran alcance y baja velocidad como ARPANET y ANSNET. También hemos visto que es posible correr el TCP/IP en otros protocolos de red de propósito general mediante una técnica llamada tunneling (procedimiento que consiste en hacer "túneles"). Aunque los detalles de las tecnologías de red específicas no son importantes, debemos considerar la siguiente idea general:

> *Los protocolos TCP/IP son muy flexibles por el hecho de que casi cualquier tecnología subyacente puede usarse para transferir tráfico de información TCP/IP.*

PARA CONOCER MÁS

Los primeros sistemas de comunicación de computadoras empleaban interconexiones punto a punto que utilizaba el hardware de líneas seriales de propósito general que describe McNamara (1982). Metcalf y Boggs (1976) introducen la red Ethernet con una versión prototipo de 3 Mbps. Digital *et. al.* (1980) especifica el estándar de 10 Mbps adoptado por muchos vendedores con el estándar 802.3 reportado en Nelson (1983). Shoch, Dalal y Redell (1982) proporcionan una perspectiva histórica de la evolución de Ethernet. En Abramson (1970) se presenta un informe del trabajo en las redes ALOHA, con una revisión de la tecnología aportada por Cotton (1979).

La tecnología de anillo (ring) con paso de prenda (token) es propuesta en Farmer Newhall (1969). Miller y Thompson (1982), así como Andrews y Shultz (1982) aportan resúmenes recientes. Otra alternativa, la red de anillo ranurado, es propuesta por Pierce (1972). Para una comparación de tecnologías, consultar Roshental (1982).

Detalles de la propuesta para la segunda red de columna vertebral NSFNET se pueden encontrar en MERIT (noviembre de 1987). Para más información sobre ARPANET, consultar Cerf (1989) y BBN (1981). Las ideas que iniciaron el X25NET están resumidas en Comer y Korb (1983); Lanzillo

y Partridge (enero de 1989) describen la marcación IP. De Prycker (1993) describe el Modo de *Transferencia Asíncrono y su uso en los servicios de área amplia*. Partridge (1994) aporta muchas tecnologías gigabit, incluyendo ATM, y describe la estructura interna de los conmutadores de alta velocidad.

Quarterman (1990) proporciona un resumen de las mayores redes de computadoras de área amplia. LaQuey (1990) aporta un directorio de redes de computadoras.

EJERCICIOS

2.1 Determine qué tecnologías de red se utilizan en su localidad.

2.2 ¿Cuál es el tamaño máximo de un paquete que puede enviarse en una red de alta velocidad como el sistema de red de Corporation's Hyperchannel?

2.3 Si su localidad utiliza tecnología de concentrador Ethernet, determine cuántas conexiones se pueden hacer hacia un solo concentrador. Si su localidad tiene varios concentradores (por ejemplo, uno en cada piso de un edificio), determine cómo están comunicados los concentradores.

2.4 ¿Cuáles son las ventajas y las desventajas del tunneling (creación de túneles)?

2.5 Lea el estándar Ethernet para encontrar detalles exactos del intervalo entre paquetes y del tamaño del preámbulo. ¿Cuál es el máximo estado permanente (steady-state) en el que Ethernet puede transportar datos?

2.6 *¿Qué características de un canal de comunicación de satélites es más deseable?* ¿La menos deseable?

2.7 Encuentre el límite inferior de tiempo que toma transferir un archivo de 5 megaoctetos a través de una red que opera a: 9600 bps, 56 Kbps, 10 Mbps, 100 Mbps y 2.4 Gbps.

2.8 ¿El procesador, el disco y el bus interno de su computadora lo operan suficientemente rápido como para enviar datos desde un archivo de disco a razón de 2 giga bits por segundo?

3

Concepto del enlace de redes y modelo arquitectónico

3.1 Introducción

Hasta ahora hemos visto los detalles de bajo nivel de transmisión a través de redes individuales, fundamento sobre el que se lleva a cabo toda la comunicación por computadora. En este capítulo, se da un gran salto conceptual al describir un esquema que nos permite reunir las distintas tecnologías de red dentro de un todo coordinado. El objetivo primordial es obtener un esquema que esconde los detalles del hardware subyacente de red a la vez que proporciona servicios universales de comunicación. El resultado principal es una abstracción de alto nivel que proporciona la estructura para todas las decisiones en cuanto a diseño. En los capítulos subsecuentes, se muestra cómo utilizamos esta abstracción para construir las capas necesarias del software para comunicación en red de redes y cómo dicho software oculta los mecanismos físicos de transporte subyacentes. En los siguientes capítulos, también se muestra cómo utilizan las aplicaciones el sistema resultante de comunicación.

3.2 Interconexión de nivel de aplicación

Los diseñadores han tomado dos enfoques diferentes para ocultar los detalles de las redes, utilizando programas de aplicación para manejar la heterogeneidad, o bien, ocultando los detalles en el sistema operativo. Las primeras interconexiones heterogéneas de red proporcionaban la uniformidad por medio de programas de nivel de aplicación. En tales sistemas, un programa de nivel de aplica-

ción que corre en cada máquina de la red "entiende" los detalles sobre las conexiones de red para esa máquina e interactúa con los programas de aplicación a través de dichas conexiones. Por ejemplo, algunos sistemas de correo electrónico consisten en programas gestores de correo que direccionan un memorando hacia una máquina a la vez. El camino desde el origen hasta el destino puede comprender muchas redes diferentes, pero esto no importa en tanto los sistemas de correo de todas las máquinas cooperen en el direccionamiento de cada mensaje.

Utilizar programas de aplicación para ocultar los detalles de la red puede parecer natural al principio, pero tal enfoque da como resultado una comunicación limitada e incómoda. Agregar funcionalidad al sistema implicaría diseñar un nuevo programa de aplicación para cada máquina. Agregar nuevo hardware de red implicaría modificar o crear nuevos programas para cada posible aplicación. En una máquina, cada programa de aplicación debe "entender" las conexiones de red para esa máquina, dando como resultado la duplicación del código.

Los usuarios que tienen experiencia con el trabajo con redes entienden que una vez que la interconexión crezca a cientos o miles de redes, nadie podría diseñar todos los programas necesarios de aplicación. Además, el éxito del esquema de comunicación un-paso-a-la-vez requiere que todos los programas de aplicación que se ejecutan a lo largo del camino funcionen correctamente. Cuando falla un programa intermedio, tanto el origen como el destino se encuentran imposibilitados para detectar o resolver el problema. Por lo tanto, los sistemas que utilizan programas intermedios no pueden garantizar una comunicación confiable.

3.3 Interconexión de nivel de red

La alternativa para proporcionar la interconexión con programas de nivel de aplicación es un sistema basado en la interconexión a nivel de red. Una interconexión a nivel de red proporciona un mecanismo que entrega en tiempo real los paquetes, desde su fuente original hasta su destino final. Conmutar pequeñas unidades de datos en vez de archivos o grandes mensajes tiene muchas ventajas. Primero, el esquema se proyecta directamente hacia el hardware subyacente de red, haciéndolo extremadamente eficiente. Segundo, la interconexión a nivel de red separa de los programas de aplicación las actividades de comunicación de datos, permitiendo que computadoras intermedias manejen el tráfico de red sin "entender" las aplicaciones que lo utilizan. Tercero, utilizar conexiones de red mantiene flexible a todo el sistema, haciendo posible la construcción de instalaciones de comunicación con propósitos generales. Cuarto, el esquema permite que los administradores de red agreguen nuevas tecnologías de red al modificar o agregar una pieza sencilla de software nuevo a nivel de red, mientras los programas de aplicación permanecen sin cambios.

La clave para diseñar una interconexión universal a nivel de red se encuentra en un concepto abstracto sobre sistemas de comunicación conocido como *enlace de redes (internetworking)*. El concepto de red de redes o *internet* es muy poderoso. Elimina la noción sobre comunicaciones de los detalles de las tecnologías de red y oculta los detalles de bajo nivel al usuario. De manera más importante, controla todas las decisiones sobre diseño de software y explica cómo manejar las direcciones físicas y las rutas. Después de revisar la motivación básica para el enlace de redes, consideraremos con mayor detalle las propiedades de una red de redes.

Comenzaremos con dos observaciones fundamentales sobre el diseño de sistemas de comunicación:

- Ningún hardware de red por sí mismo puede satisfacer todos los requerimientos.
- Los usuarios buscan la interconexión universal.

La primera observación es técnica. Las redes de área local, que proporcionan la mayor velocidad de comunicación, están limitadas en cuanto a su alcance geográfico; las redes de área amplia abarcan grandes distancias pero no pueden proporcionar conexiones de alta velocidad. Ninguna tecnología de red por sí misma satisface todas las necesidades, así que nos vemos forzados a considerar muchas tecnologías subyacentes de hardware.

La segunda observación es evidente. Por último, nos gustaría poder comunicarnos entre dos puntos cualquiera que éstos sean. En particular, queremos un sistema de comunicación que no esté limitado por las fronteras de las redes físicas.

La meta es construir una interconexión de redes, unificada y cooperativa, que incorpore un servicio universal de comunicación. Dentro de cada red, las computadoras utilizarán funciones subyacentes de comunicación sin importar la tecnología, como las que se describieron en el capítulo 2. El nuevo software, incorporado entre los mecanismos de comunicación de tecnología independiente y los programas de aplicación, ocultará los detalles de bajo nivel y hará que el grupo de redes parezca ser una sola y gran red. Un esquema de interconexión como el que se describe se conoce como *red de redes* o *internet*.

La idea de construir una red de redes sigue un patrón estándar de diseño de sistemas; los investigadores se imaginan un equipo de computación de alto nivel y trabajan con la tecnología computacional disponible, agregando capas de software hasta que logran un sistema que implante de manera eficaz el equipo de alto nivel deseado. En la siguiente sección, se muestra el primer paso del proceso de diseño al definir de manera más precisa el objetivo.

3.4 Propiedades de Internet

Tener una noción del servicio universal es importante, pero, por sí misma ésta no reúne todas las ideas que tenemos en mente sobre una red de redes unificada, ya que puede haber muchas ejecuciones de servicios universales. En nuestro diseño, queremos ocultar al usuario la arquitectura subyacente de la red de redes. Esto es, no queremos obligar a que los usuarios o los programas de aplicación entiendan los detalles de las interconexiones del hardware para utilizar la red de redes. Tampoco queremos imponer una topología de interconexión de red. En particular, agregar una nueva red a la red de redes no debe implicar la conexión a un punto centralizado de conmutación, ni tampoco implicar la añadidura de conexiones físicas directas entre la nueva red y las redes ya existentes. Queremos ser capaces de enviar datos a través de redes intermedias aunque no estén conectadas en forma directa a las máquinas de origen o destino. Queremos que todas las máquinas en la red de redes compartan un juego universal de identificadores de máquina (en los que se puede pensar como *nombres* o *direcciones*).

Nuestra idea sobre una red de redes unificada también incluye la idea de la independencia de red en la interfaz del usuario. Esto es, queremos que el grupo de operaciones utilizadas para establecer comunicación o para transferir datos se mantenga independiente tanto de las tecnologías subyacentes de red como de la máquina destino. Claro está, un usuario no tiene que entender la topología de la interconexión de redes cuando cree programas de aplicación que se comuniquen.

3.5 Arquitectura de Internet

Como hemos visto cómo se conectan máquinas a redes individuales, surge la pregunta: "¿cómo se interconectan las redes para formar una red de redes?" La respuesta tiene dos partes. Físicamente, dos redes sólo se pueden conectar por medio de una computadora en medio de las dos. Sin embargo, una conexión física no proporciona la interconexión que tenemos en mente, debido a que dicha conexión no garantiza que la computadora cooperará con otras máquinas que se desean comunicar. Para obtener una red de redes viable, necesitamos computadoras que estén dispuestas a intercambiar paquetes de una red a otra. Las computadoras que interconectan dos redes y transfieren paquetes de una a otra se conocen como *pasarelas* o *compuertas de red de redes* o *ruteadores de red de redes*.[1]

Consideremos un ejemplo consistente en dos redes físicas que se muestran en la figura 3.1. En la figura, el ruteador *R* conecta las redes *1* y *2*. Para que *R* actúe como ruteador, debe capturar y transferir los paquetes de la red *1* que estén dirigidos a las máquinas de la red *2*. De manera similar, *R* debe capturar y transferir los paquetes de la red *2* que estén dirigidos a las máquinas de la red *1*.

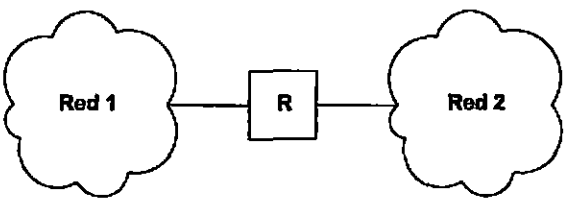


Figura 3.1 Dos redes físicas interconectadas por *R*, un ruteador (puerta IP).

En la figura, las formas que representan nubes se utilizan para denotar redes físicas, ya que el hardware específico no es importante. Cada red puede ser una LAN o una WAN, y cada una puede tener pocos o muchos anfitriones conectados.

3.6 Interconexión a través de ruteadores IP

Cuando la conexión de red de redes se vuelve más compleja, los ruteadores necesitan conocer la topología de la red de redes más allá de las redes que interconectan. Por ejemplo, en la figura 3.2 se muestran tres redes interconectadas por medio de dos ruteadores.

En este ejemplo, el ruteador R_1 debe transferir, de la red *1* a la red *2*, todos los paquetes destinados a las máquinas de la red *2* o de la red *3*. Para una gran red de redes, la tarea de los ruteadores de tomar decisiones sobre dónde enviar paquetes se vuelve más compleja.

---

[1] La literatura original utilizaba en término *pasarela IP*. Sin embargo, los fabricantes han adoptado el término *ruteador IP*, los dos términos se emplean de manera alterna en este texto.

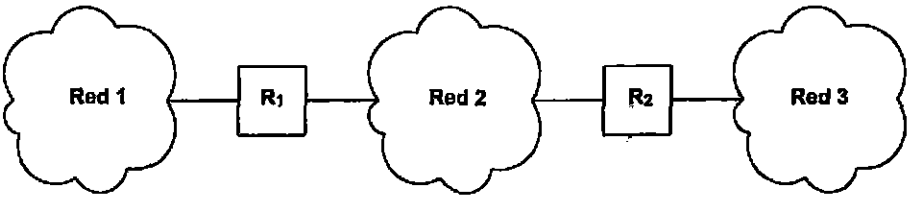


Figura 3.2 Tres redes interconectadas por dos ruteadores.

La idea de un ruteador parece sencilla, pero es importante debido a que proporciona una forma para interconectar redes, no sólo máquinas. De hecho, ya hemos descubierto el principio de interconexión utilizado a través de una red de redes:

> *En una red de redes TCP/IP, las computadoras llamadas* ruteadores *o* pasarelas *proporcionan todas las interconexiones entre las redes físicas.*

Se puede pensar que los ruteadores, que deben saber cómo rutear paquetes hacia su destino, son grandes máquinas con suficiente memoria primaria o secundaria para guardar información sobre cada máquina dentro de la red de redes a la que se conectan. Sin embargo, los ruteadores utilizados en las redes de redes TCP/IP son por lo general computadoras pequeñas. A menudo tienen muy poco o nada de almacenamiento en disco y memorias principales limitadas. El truco para construir un ruteador pequeño para red de redes reside en el siguiente concepto:

> *Los ruteadores utilizan la red de destino, no el anfitrión de destino, cuando rutean un paquete.*

Si el ruteo está basado en redes, la cantidad de información que necesita guardar un ruteador es proporcional al número de redes dentro de otra red, no al número de computadoras.

Debido a que los ruteadores juegan un papel clave en la comunicación de una red de redes, nos referiremos a ellos en los siguientes capítulos y trataremos los detalles de cómo operan y cómo aprenden las rutas. Por ahora, asumiremos que es posible y práctico tener rutas correctas para todas las redes en cada ruteador dentro de la red de redes. También, asumiremos que sólo los ruteadores proporcionan conexiones entre las redes físicas dentro de una red de redes.

3.7 El punto de vista del usuario

Recuerde que el TCP/IP está diseñado para proporcionar interconexión universal entre máquinas, independientemente de las redes en particular a las que están conectadas. Por lo tanto, queremos que un usuario vea una red de redes como una sola red virtual a la cual todas las máquinas se conectan sin importar sus conexiones físicas. En la figura 3.3a, se muestra cómo pensar en una red de redes, en vez de pensar en redes constitutivas, simplifica los detalles y ayuda al usuario a concep-

tualizar la comunicación. Además de los ruteadores que interconectan redes físicas, se necesita software en cada anfitrión para permitir que los programas de aplicación utilicen la red de redes como si ésta fuera una sola red física real.

La ventaja de proporcionar una interconexión a nivel de red ahora se vuelve clara. Debido a que los programas de aplicación que se comunican a través de la red de redes no conocen los detalles de las conexiones subyacentes, se pueden correr sin cambios en cualquier máquina. Como los detalles de la conexión física entre cada máquina y la red física están ocultos en el software para red, sólo éste necesita cambiar cuando aparecen nuevas conexiones físicas o cuando desaparecen conexiones antiguas. De hecho, es posible optimizar la estructura interna de la red de redes alterando las conexiones físicas sin compilar de nuevo los programas de aplicación.

Una segunda ventaja de tener la comunicación a nivel de red es menos visible: los usuarios no tienen que entender o recordar cómo se conectan las redes o qué tipo de tráfico llevan. Se pueden crear programas de aplicación que se comuniquen independientemente de la conectividad física subyacente. De hecho, los gerentes de red están en libertad de cambiar partes interiores de la arquitectura subyacente sin tener que cambiar software de aplicación en la mayoría de las computadoras conectadas (claro está, el software de red se necesita reconfigurar cuando se mueve una computadora hacia una nueva red).

Como se muestra en la figura 3.3b, los ruteadores no proporcionan conexiones directas entre cada par de redes. Puede ser necesario que el tráfico que viaje de una máquina a otra pase a través de muchas redes intermedias. Por lo tanto, las redes que participan en una red de redes son análogas al sistema de carreteras interestatales de cualquier país: cada red accede a manejar el tráfico que llegue, a cambio del derecho de enviar tráfico a través de la red de redes. Los usuarios comunes no se ven afectados ni tienen conocimiento del tráfico adicional que pasa por su red local.

3.8 Todas las redes son iguales

En el capítulo 2, se vieron ejemplos del hardware de red utilizado para formar redes de redes TCP/IP y se ilustró la gran diversidad de tecnologías. Hemos descrito una red de redes como un conjunto de redes cooperativas interconectadas. Ahora, es importante entender un concepto fundamental: desde el punto de vista de una red de redes, cualquier sistema de comunicación capaz de transferir paquetes se cuenta como una sola red, independientemente de sus características de retraso y generación de salida, tamaño máximo de paquete o escala geográfica. En particular, en la figura 3.3b se utilizaba la misma representación de nubes para referirse a todas las redes físicas, debido a que el TCP/IP las trata de igual manera sin importar sus diferencias. El punto es:

> *Los protocolos TCP/IP para red de redes tratan de manera igual a todas las redes. Una red de área local como Ethernet, una red de área amplia como la columna vertebral ANSNET o un enlace punto-a-punto entre dos máquinas se cuentan como redes individuales.*

Los lectores que no estén acostumbrados a la arquitectura de una red de redes pueden encontrar difícil aceptar una vista tan simple de las redes. En esencia, el TCP/IP define una abstracción

de "red" que oculta los detalles de las redes físicas; aprenderemos cómo dicha abstracción ayuda a que el TCP/IP sea tan poderoso.

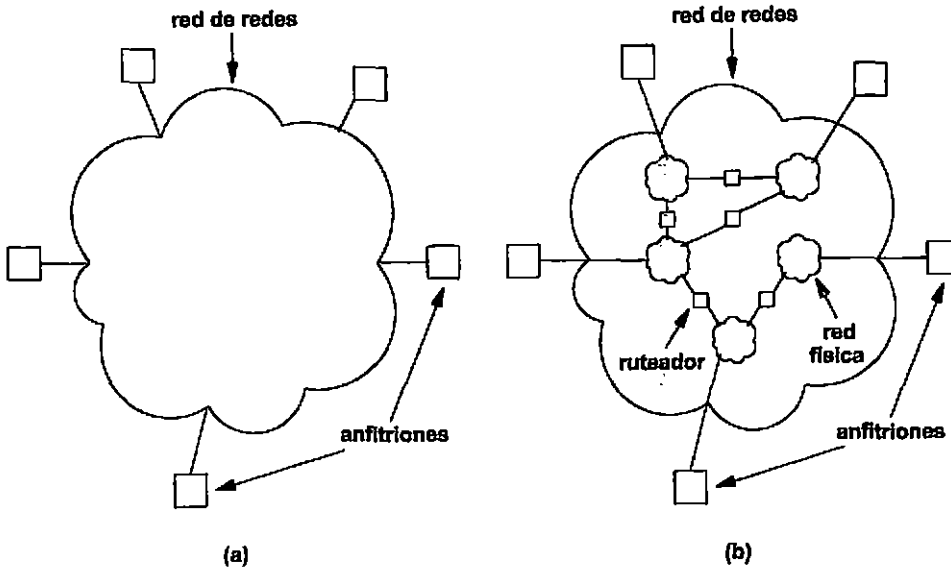


Figura 3.3 (a) Punto de vista del usuario de una red de redes TCP/IP, en la que cada computadora parece conectarse a una sola y gran red, y (b) estructura de las redes físicas y los ruteadores que proporcionan la interconexión.

3.9 Las preguntas sin respuesta

Nuestro bosquejo de una red de redes deja sin respuesta muchas preguntas. Por ejemplo, usted se puede preguntar sobre la forma exacta de direcciones de máquina de una red de redes o cómo dicha dirección se relaciona con las direcciones de hardware físico de Ethernet, FDDI o ATM que se describieron en el capítulo 2. En los siguientes tres capítulos, se tratan estas preguntas. En ellos se describe el formato de las direcciones IP y se ilustra cómo los anfitriones distinguen entre las direcciones de red de redes y las direcciones físicas. Quizá también quiera saber cómo se ve exactamente un paquete cuando viaja a través de una red de redes o qué pasa cuando los paquetes llegan demasiado rápido para ser manejados por un anfitrión o un ruteador. En el capítulo 7, se responde a estas preguntas. Por último, se puede preguntar cómo muchos programas de aplicación, al correr de manera concurrente en una sola máquina, pueden enviar y recibir paquetes desde y hacia muchos destinos sin confundirse con las transmisiones de otros, o cómo los ruteadores de una red de redes aprenden las rutas. También se dará respuesta a todas estas preguntas.

Aunque por ahora parezca indefinida, la dirección que estamos siguiendo nos permitirá aprender tanto la estructura como la utilización del software protocolo de red de redes. Examinaremos cada parte, incluyendo tanto los conceptos y principios como los detalles técnicos. Comenza-

mos por describir la capa física de comunicación sobre la que se construye una red de redes. Cada uno de los siguientes capítulos explorará una parte del software de red de redes, hasta que comprendamos cómo funcionan todas las piezas.

3.10 Resumen

Una red de redes es mucho más que un conjunto de redes interconectadas por computadoras. En enlace de redes implica que todos los sistemas interconectados estén de acuerdo en reglas que permitan que cada computadora se comunique con cualquier otra. En particular, una red de redes permitirá que dos máquinas se comuniquen, inclusive si el camino de comunicación entre ellas pasa a través de una red a la que ninguna de las dos se conecta de manera directa. Tal cooperación sólo es posible cuando las computadoras se ponen de acuerdo para utilizar un grupo de identificadores universales y un conjunto de procedimientos para transferir los datos a su destino final.

En una red de redes, las interconexiones entre redes se forman por computadoras llamadas ruteadores IP, o pasarelas IP, que se conectan a dos o más redes. Un ruteador encamina paquetes entre redes al recibirlos de una red y enviarlos a otra.

PARA CONOCER MÁS

Nuestro modelo de una red de redes viene de Cerf and Cain (1983) y Cerf y Kahn (1974), quienes describen una red de redes como un grupo de redes interconectadas por ruteadores y bosquejan un protocolo para red de redes similar al que finalmente se desarrolló para el grupo de protocolos TCP/IP. Se puede encontrar mayor información sobre la arquitectura Internet en Postel (1980); Postel, Sunshine, and Chen (1981); y en Hinden, Haverty, and Sheltzer (1983). Shoch (1978), presenta aspectos sobre nomenclatura y direccionamiento en una red de redes. Boggs *et. al.* (1980), describe la red de redes desarrollada en Xerox PARC, alternativa a la red de redes TCP/IP que también examinaremos. Cheriton (1983) describe el enlace de redes, relacionado con el V-system.

EJERCICIOS

3.1 Cambiar la información en un ruteador puede ser contraproducente ya que es imposible cambiar de manera simultánea todos los ruteadores. Investigue algoritmos que garanticen instalar un cambio en un grupo de computadoras o no instalar cambios en ninguna.

3.2 En una red de redes, los ruteadores intercambian de manera periódica información sobre sus tablas de ruteo, lo que hace posible que aparezca un nuevo ruteador y comience a rutear paquetes. Investigue los algoritmos utilizados para intercambiar información de ruteo.

3.3 Compare la organización de una red de redes TCP/IP con el tipo de red de redes diseñado por Xerox Corporation.

3.4 ¿Qué procesadores se utilizaron como ruteadores en Internet? ¿Le sorprende la velocidad y tamaño del hardware antiguo de ruteador? ¿Por qué?

3.5 ¿Aproximadamente cuántas redes comprenden la red de redes en su sitio de trabajo? ¿Aproximadamente, cuántos ruteadores?

3.6 Considere la estructura interna del ejemplo de una red de redes presentado en la figura 3.3b. ¿Qué ruteadores son cruciales? ¿Por qué?

4

Direcciones Internet

4.1 Introducción

En el capítulo anterior se definió una red de redes TCP/IP como una red virtual formada al interconectar redes físicas a ruteadores. En este capítulo se analizan las direcciones, un ingrediente esencial que le ayuda al software TCP/IP a ocultar los detalles de las redes físicas y hace que la red de redes parezca una sola entidad uniforme.

4.2 Identificadores universales

Se dice que un sistema de comunicaciones proporciona *servicio universal de comunicaciones* si permite que cualquier computadora anfitrión se comunique con cualquier otro anfitrión. Para que nuestro sistema de comunicaciones sea universal, necesita un método aceptado de manera global para identificar cada computadora que se conecta a él.

A menudo, los identificadores de anfitrión se clasifican como *nombres, direcciones* o *rutas*. Shock (1978) sugiere que un nombre identifica *lo que* un objeto es una dirección identifica *donde* está y una ruta indica *cómo* llegar hasta ahí. Aunque estas definiciones son intuitivas, pueden ser confusas. Los nombres, direcciones y rutas se refieren a representaciones sucesivas de bajo nivel de *identificadores de anfitrión*. En general, las personas prefieren nombres pronunciables para identificar máquinas, mientras que el software trabaja de manera más eficiente con representaciones compactas de los identificadores que nosotros conocemos como direcciones. Este término también se pudo haber llamado identificadores universales de anfitrión TCP/IP. Se tomó la decisión de llamarlos así para estandarizarlos en direcciones compactas y binarias que hacen que

cómputos tales como la selección de una ruta sean eficientes. Por ahora, sólo trataremos las direcciones binarias y pospondremos las preguntas de cómo llevar a cabo la transformación entre direcciones binarias y nombres pronunciables, y de cómo utilizar direcciones para el ruteo.

4.3 Tres tipos primarios de direcciones IP

Piense en una red de redes como en una gran red igual a cualquier otra red física. La diferencia, claro está, es que la red de redes tiene una estructura virtual, imaginada por sus diseñadores, e implantada totalmente en software. Por lo tanto, los diseñadores son libres de elegir el formato y tamaño de los paquetes, las direcciones, las técnicas de entrega, y así en adelante; nada es dictado por el hardware. Para las direcciones, los diseñadores del TCP/IP eligen un esquema análogo al direccionamiento en las redes físicas, en el que cada anfitrión en la red de redes tiene asignada una dirección de número entero de 32 bits, llamada su *dirección de red de redes* o *dirección IP*. La parte inteligente del direccionamiento en una red de redes es que los números enteros son seleccionados con cuidado para hacer eficiente el ruteo. De manera específica, una dirección IP codifica la identificación de la red a la que se conecta el anfitrión, así como la identificación de un anfitrión único en esa red. Podemos resumir que:

> *Cada anfitrión en una red de redes TCP/IP tiene asignada una dirección de número entero de 32 bits que se utiliza en todas las comunicaciones con dicho anfitrión.*

Los detalles de una dirección IP nos ayudan a entender mejor las ideas abstractas. Por ahora, damos una visión simplificada, pero la ampliaremos más adelante. En el caso más sencillo, cada anfitrión conectado a la red de redes tiene asignado un identificador universal de 32 bits como su dirección dentro de la red. Los bits de dirección IP de todos los anfitriones en una red comparten un prefijo común.

Conceptualmente, cada dirección es un par (*netid, hostid*), en donde *netid* identifica una red y *hostid* un anfitrión dentro de la red. En la práctica, cada dirección IP debe tener una de las primeras tres formas mostradas en la figura 4.1.[1]

Definida una dirección IP, se puede determinar su tipo según los tres bits de orden, de los que son necesarios sólo dos bits para distinguir entre los tres tipos primarios. Las direcciones tipo *A*, que se utilizan para las pocas redes que tienen más de 2^{16} de anfitriones (por ejemplo, 65,536), asignan 7 bits al campo netid y 24 bits al campo hostid. Las direcciones tipo B, que se utilizan para redes de tamaño mediano que tienen entre 2^8 (por ejemplo, 256) y 2^{16} anfitriones, asignan 14 bits al campo netid y 16 bits al hostid. Por último, las direcciones tipo *C*, que tienen menos de 2^8 anfitriones, asignan 21 bits al campo netid y sólo 8 bits al hostid. Nótese que las direcciones IP se han definido de tal forma que es posible extraer rápidamente los campos hostid o netid. Los ruteadores, que utilizan el campo netid de una dirección para decidir a dónde enviar un paquete, dependen de una extracción eficiente para lograr una velocidad alta.

---

[1] La cuarta forma, reservada para la multidifusión en la red de redes, será descrita en un capítulo posterior; por ahora, limitaremos nuestro comentarios a las formas que especifican direcciones de objetos individuales.

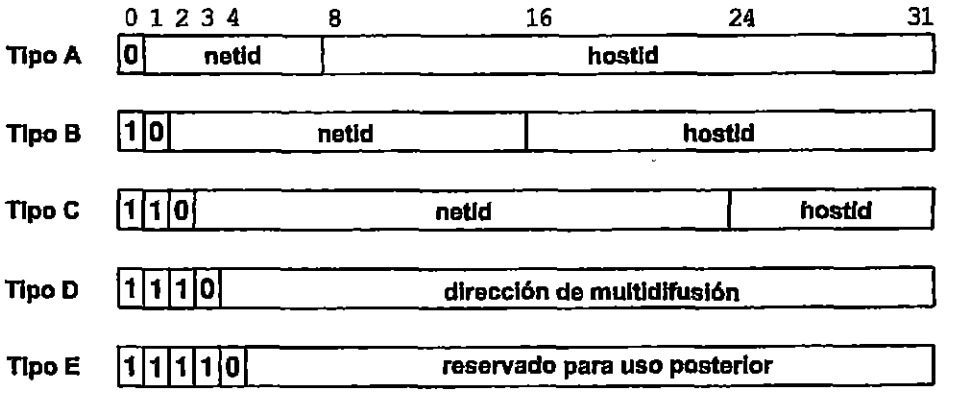


Figura 4.1 Las cinco formas de direcciones de Internet (IP). Las tres formas primarias, tipos *A*, *B* y *C*, se pueden distinguir por medio de los tres primeros bits.

4.4 Las direcciones especifican conexiones de red

Para simplificar el análisis, dijimos que una dirección de red de redes identifica un anfitrión, pero esto no es del todo preciso. Considere un ruteador que conecta dos redes físicas. ¿Cómo podemos asignar una sola dirección IP si dicha dirección codifica un identificador de red así como un identificador de anfitrión? De hecho, no podemos. Cuando computadoras convencionales tienen dos o más conexiones físicas se las llama *anfitriones multi-homed*. Los anfitriones multi-homed y los ruteadores requieren de muchas direcciones IP. Cada dirección corresponde a una de las conexiones de red de las máquinas. Referirnos a los anfitriones multi-homed nos lleva a la siguiente consideración:

> *Debido a que las direcciones IP codifican tanto una red y un anfitrión en dicha red, no especifican una computadora individual, sino una conexión a la red.*

Por lo tanto, un ruteador que conecta cierto número de redes tiene cierto número de direcciones IP distintas, una para cada conexión de red.

4.5 Direcciones de red y de difusión

Ya hemos mencionado la mayor ventaja de la codificación de información de red en las direcciones de red de redes: hacer posible que exista un ruteo eficiente. Otra ventaja es que las direcciones de red de redes se pueden referir tanto a redes como a anfitriones. Por regla, nunca se asigna un campo hostid igual a *0* a un anfitrión individual. En vez de eso, una dirección IP con campo hostid de *0* se utiliza para referirse a la red en sí misma. En resumen:

Las direcciones de red de redes se pueden utilizar para referirse a redes así como a anfitriones individuales. Por regla, una dirección que tiene todos los bits del campo hostid igual a 0, se reserva para referirse a la red en sí misma.

Otra ventaja significativa del esquema de direccionamiento en una red de redes es que éste incluye una *dirección de difusión* que se refiere a todos los anfitriones en la red. De acuerdo con el estándar, cualquier campo hostid consistente en solamente *1s*, está reservado para la difusión.[2] En muchas tecnologías de red (por ejemplo, Ethernet), la difusión puede ser tan eficiente como la transmisión normal; en otras, la difusión encuentra apoyo en el software de red, pero requiere sustancialmente mayor retraso que la transmisión simple. Algunas redes inclusive no cuentan con difusión. Por lo tanto, tener una dirección IP de difusión no garantiza la disponibilidad o eficiencia de la entrega por difusión. En resumen:

Las direcciones IP se pueden utilizar para especificar la difusión; estas direcciones se transforman en difusión por hardware, si ésta se encuentra disponible. Por regla, una dirección de difusión tiene todos los bits del campo hostid asignados como 1.

4.6 Difusión limitada

Técnicamente, la dirección de difusión que describimos se conoce como *dirección de difusión dirigida*, debido a que contiene tanto una identificación válida de red como el campo hostid de difusión. Una dirección de difusión dirigida se puede interpretar sin ambigüedades en cualquier punto de una red de redes ya que identifica en forma única la red objetivo, además de especificar la difusión en dicha red. Las direcciones de difusión dirigida proporcionan un mecanismo poderoso (y a veces algo peligroso) que permite que un sistema remoto envíe un solo paquete que será publidifundido en la red especificada.

Desde el punto de vista del direccionamiento, la mayor desventaja de la difusión dirigida es que requiere un conocimiento de la dirección de red. Otra forma de dirección de difusión, llamada *dirección de difusión limitada* o *dirección de difusión en red local*, proporciona una dirección de difusión para la red local, independientemente de la dirección IP asignada. La dirección de difusión local consiste en treinta y dos *1s* (por esto, a veces se le llama la dirección de difusión "todos *1s*"). Un anfitrión puede utilizar la dirección de difusión limitada como parte de un procedimiento de arranque antes de conocer su dirección IP o la dirección IP de la red local. Sin embargo, una vez que el anfitrión conoce la dirección IP correcta para la red local, tiene que utilizar la difusión dirigida.

Como regla general, los protocolos TCP/IP restringen la difusión al menor número posible de máquinas. En el capítulo de direccionamiento de subred, veremos cómo afecta esta regla a muchas redes que comparten direcciones.

---

[2] Desafortunadamente, una versión antigua de código TCP/IP que acompañaba al UNIX de Berkeley utilizó de forma incorrecta todos los ceros para la difusión. Como el error aún existe, el software TCP/IP a menudo incluye una opción que permite que un sitio utilice todos los ceros para difusión.

4.7 Interpretación de cero como "esto"

Hemos visto que un campo consistente en *1s* puede interpretarse como "todos", como en "todos los anfitriones" de una red. En general, el software de red de redes interpreta los campos que consisten en ceros *(0)* como si fuera "esto". La interpretación aparece a lo largo de la literatura. Por lo tanto, una dirección IP con campo hostid *0* se refiere a "este" anfitrión, y una dirección de red de redes con el ID de red de *0* se refiere a "esta" red. Claro está, sólo es significativo utilizar una dirección en esa forma dentro de un contexto en el que se pueda interpretar de una manera no ambigua. Por ejemplo, si una máquina recibe un paquete en el que el campo netid de la dirección de destino es *0* y el campo hostid de la dirección de destino corresponde a su dirección, el receptor interpreta el campo netid como "esta" red (por ejemplo, la red sobre la cual llegó el paquete).

La utilización de netid *0* es de especial importancia en casos en los que un anfitrión se quiere comunicar hacia una red pero todavía no conoce su dirección IP. El anfitrión utiliza de manera temporal la ID *0* de red, y otros anfitriones de la red interpretan la dirección como si fuera "esta red". En la mayor parte de los casos, las respuestas tendrán la dirección de red totalmente especificada, permitiendo que el transmisor original la registre para utilizarla después. En el capítulo 9 se discutirá en detalle cómo un anfitrión determina el campo netid de la red local.

4.7.1. Direccionamiento de subred y multidifusión

El esquema de direccionamiento descrito hasta aquí requiere un prefijo único de red para cada red física. En el capítulo 10 se consideran dos extensiones importantes al esquema de direccionamiento, diseñadas para conservar las direcciones de red: direccionamiento de subred y direccionamiento sin tipo. Además de la difusión, el esquema de direcciones IP incorpora una forma especial de entrega a muchos puntos conocida como *multidifusión*. La multidifusión es de gran utilidad para las redes en las que la tecnología de hardware incorpora la entrega por multidifusión. En el capítulo 17 se analiza en detalle el direccionamiento y la entrega mediante multidifusión.

4.8 Debilidades del direccionamiento de Internet

La codificación de información de red en una dirección de red de redes tiene algunas desventajas. La desventaja más obvia es que las direcciones se refieren a las conexiones de red, no a la computadora anfitrión:

> *Si una computadora anfitrión se mueve de una red a otra, su dirección IP debe cambiar.*

Para entender las consecuencias, considere a los viajeros que quieren desconectar su computadora personal, llevarla con ellos durante el viaje y reconectarla a la red de redes al llegar a su destino. La computadora personal no puede tener asignada una dirección IP permanente ya que ésta identifica la red a la que está conectada la computadora.

Otra debilidad del esquema de direccionamiento en una red de redes es que cuando una red tipo *C* crece hasta tener más de 255 anfitriones, tiene que cambiar su dirección a una tipo *B*. Aunque esto puede parecer un problema menor, el cambio de direcciones de red puede tomar demasiado tiempo y ser muy difícil de depurar. Debido a que la mayor parte del software no está diseñado para manejar muchas direcciones para la misma red física, los administradores no pueden planificar una transición suave en la que introduzcan lentamente la nueva dirección. En vez de eso, tienen que dejar de utilizar abruptamente la dirección de red, cambiar las direcciones de todas las máquinas y reiniciar la comunicación utilizando la nueva dirección de red.

La imperfección más importante del esquema de direccionamiento en una red de redes no se volverá evidente hasta que examinemos el ruteo. Sin embargo, su importancia requiere una breve introducción. Hemos sugerido que el ruteo se basará en direcciones de red de redes, con el campo netid de la dirección utilizado para tomar decisiones de ruteo. Considere un anfitrión con dos conexiones hacia la red de redes. Sabemos que un anfitrión así debe tener más de una dirección IP. Lo siguiente es cierto:

> *Como el ruteo utiliza la parte de red de la dirección IP, el camino tomado por los paquetes que viajan hacia un anfitrión con muchas direcciones IP depende de la dirección utilizada.*

Las implicaciones son sorprendentes. Los humanos piensan en cada anfitrión como en una sola entidad y quieren utilizar un solo nombre. A veces se sorprenden al encontrar que deben aprender más de un nombre, y se sorprende aún más cuando encuentran que los paquetes enviados en los que utilizan muchos nombres pueden comportarse de manera diferente.

Otra consecuencia sorprendente del esquema de direccionamiento en una red de redes es que no es suficiente conocer una dirección IP para un destino; puede ser imposible llegar al destino utilizando esa dirección. Considere la red de redes mostrada en la figura 4.2. En la figura, dos anfitriones, *A* y *B*, se conectan a la red *1*, y por lo general se comunican utilizando en forma directa dicha red. Por lo tanto, los usuarios en el anfitrión *A* se deben referir normalmente al anfitrión *B* utilizando la dirección IP I_3. Existe un camino alternativo de *A* a *B* a través del ruteador *R*, y se utiliza

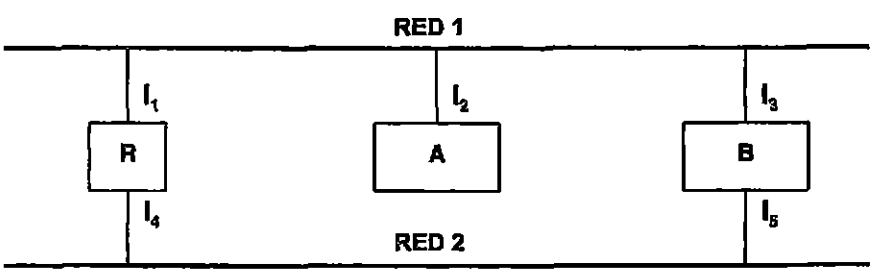


Figura 4.2 Red de redes de ejemplo con un anfitrión multi-homed, *B*, que muestra un problema del esquema IP de direccionamiento. Su interfaz I_3 se desconecta, *A* debe utilizar la dirección I_5 para llegar a *B*, enviando paquetes a través del ruteador *R*.

siempre que *A* envíe paquetes a la dirección IP I_5 (dirección de B en la red 2). Ahora suponga que falla la conexión de *B* con la red *1*, pero en sí la máquina sigue funcionando (por ejemplo, se rompe un cable entre *B* y la red *1*). Los usuarios en *A* que especifican la dirección IP I_3 no pueden llegar a *B*, pero los usuarios que especifican la dirección I_5 sí lo pueden hacer. Estos problemas con respecto a la nomenclatura y al direccionamiento surgirán de nuevo en capítulos posteriores, cuando consideremos el ruteo y el enlace de nombres.

4.9 Notación decimal con puntos

Cuando se comunican a los usuarios, ya sea en documentos técnicos o a través de programas de aplicación, las direcciones IP se escriben como cuatro enteros decimales separados por puntos, en donde cada entero proporciona el valor de un octeto de la dirección IP.[3] Por lo tanto, la dirección de 32 bits de una red de redes

10000000 00001010 00000010 00011110

se escribe

128.10.2.30

Utilizaremos la notación decimal con puntos cuando expresemos direcciones IP a través del resto del texto. De hecho, la mayor parte del software TCP/IP que muestra una dirección IP o que requiere que una persona la introduzca, utiliza notación decimal con puntos. Por ejemplo, el comando *netstat* de UNIX que muestra el ruteo actual, y los programas de aplicación como *telnet* y *ftp* utilizan la notación decimal con puntos cuando aceptan o muestran direcciones IP. Por lo tanto, sería útil entender la relación entre los tipos de direcciones IP y los números decimales con puntos. En la tabla de la figura 4.3 se resumen el rango de valores para cada tipo.

Tipo	Dirección más baja	Dirección más alta
A	0.1.0.0	126.0.0.0
B	128.0.0.0	191.255.0.0
C	192.0.1.0	223.255.255.0
D	224.0.0.0	239.255.255.255
E	240.0.0.0	247.255.255.255

Figura 4.3 Rango de valores decimales con punto que corresponde a cada tipo de dirección IP. Algunos valores están reservados para propósitos especiales.

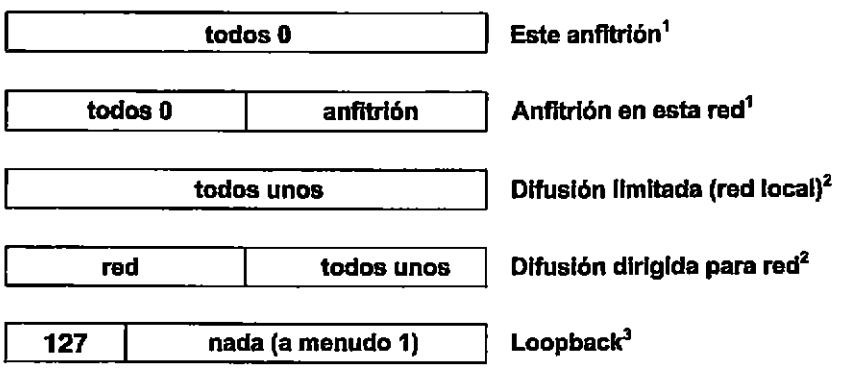
[3] La notación decimal con puntos a veces se conoce como *notación cuadrangular con puntos*.

4.10 Dirección loopback

En la tabla de la figura 4.3 se muestra que no todas las posibles direcciones se asignaron a algún tipo. Por ejemplo, la dirección 127.0.0.0, valor del rango tipo A, se reserva para *loopback*; y está diseñada para utilizarse en las pruebas del TCP/IP y para la comunicación de los procesos internos en la máquina local. Cuando algún programa utiliza la dirección loopback como destino, el software de protocolo en una computadora regresa los datos sin generar tráfico a través de alguna red. En el texto se declara de manera explícita que un paquete enviado a la dirección de una red 127 nunca debe aparecer en ninguna red. Además, un anfitrión o un ruteador nunca deben difundir información de ruteo ni de accesabilidad para el número de red *127*, pues no es una dirección de red.

4.11 Resumen de reglas especiales de direccionamiento

En la práctica, el IP utiliza sólo unas cuantas combinaciones de ceros (*0*) ("ésta") o unos (1) ("toda"). En la figura 4.4 se listan las posibilidades.



Notas: [1] Es permitido solamente en el arranque de sistema pero nunca es una dirección válida de destino.
[2] Nunca es una dirección válida de origen.
[3] Nunca debe aparecer en una red.

Figura 4.4 Formas especiales de direcciones IP, incluyendo combinaciones de ceros (0, esto), unos (1, todo). La longitud del campo de red de una difusión dirigida depende del tipo de dirección de red.

Como se menciona en las notas de la figura, la utilización de todos los ceros (*0*) para la red sólo está permitida durante el procedimiento de iniciación. Permite que una máquina se comunique temporalmente. Una vez que la máquina "aprende" su red y su dirección IP correctas, no debe utilizar la red *0*.

4.12 Autoridad de direccionamiento Internet

Para garantizar que el campo de red dentro de una dirección de Internet es único, todas las direcciones de Internet son asignadas por una autoridad central. La *Autoridad Internet de Números Asignados (IANA)* establece los procedimientos y tiene el control sobre los números asignados. Sin embargo, cuando una organización se une a Internet, puede obtener direcciones de red desde el *Centro de Información de la Red Internet (INTERNIC).*

Sólo se necesita una autoridad central para asignar el campo de red de una dirección; una vez que una organización obtiene un prefijo de red, puede escoger cómo asignar un sufijo único a cada anfitrión de su red sin tener que contactar a la autoridad central. La Autoridad Internet asigna un número tipo C a una red con pocas computadoras conectadas a ella (menos de 255); se reserva los números tipo B para una organización que tiene una red más grande. Por último, una organización debe tener una red con más de 65535 anfitriones conectados antes de que pueda obtener un número tipo A. El espacio del nombre está diferido debido a que la mayoría de las redes son pequeñas, un menor número de ellas son medianas y sólo muy pocas son gigantes.

Solamente es esencial para la autoridad central asignar direcciones IP para redes que están (o estarán) conectadas a la red global Internet. Una corporación individual puede tener la responsabilidad de asignar direcciones únicas de red dentro de su red de redes TCP/IP, siempre y cuando nunca conecte esa red de redes al mundo exterior. De hecho, muchos grupos corporativos que utilizan protocolos TCP/IP se autoasignan direcciones de red de redes. Por ejemplo, la dirección de red 9.0.0.0 se asignó a IBM Corporation y la dirección 12.0.0.0 se asignó a AT&T. Si una organización decide utilizar protocolos TCP/IP en dos de sus redes, sin conexión al Internet global, la organización puede asignar las direcciones 9.0.0.0 y 12.0.0.0 a sus redes locales. Sin embargo, la experiencia ha demostrado que no es acertado crear una red de redes privada utilizando las mismas direcciones que la red global Internet, debido a que imposibilita la interoperabilidad en el futuro y puede causar problemas cuando se trate de intercambiar software con otros sitios. Por lo tanto, se alienta fuertemente a todos los usuarios de TCP/IP a que se tomen el tiempo para obtener las direcciones oficiales de Internet por medio de INTERNIC.

4.13 Un ejemplo

Para hacer más claro el esquema de direccionamiento IP, considere el ejemplo de dos redes en el Departamento de Ciencias de la Computación de la Universidad de Purdue, que fueron conectadas a Internet a mediados de los años ochenta. En la figura 4.5 se muestran las direcciones de red y se ilustra cómo los ruteadores interconectan las redes.

En el ejemplo se muestran tres redes y los números de red asignados: la red ARPANET (10.0.0.0), una red Ethernet (128.10.0.0), y una red token ring (192.5.48.0). De acuerdo con la tabla de la figura 4.3, las direcciones son, respectivamente, tipo *A*, *B*, y *C*.

La figura 4.6 muestra las mismas redes con computadoras anfitrionas conectadas y direcciones de Internet asignadas a cada conexión de red.

En la figura, se conectan a la red cuatro anfitriones llamados *Arthur, Merlin, Guenevere y Lancelot. Taliesyn* es un ruteador que conecta las redes ARPANET y token ring, y *Glatisant* es un ruteador que conecta las redes token ring y Ethernet. El anfitrión *Merlin* tiene conexiones con las

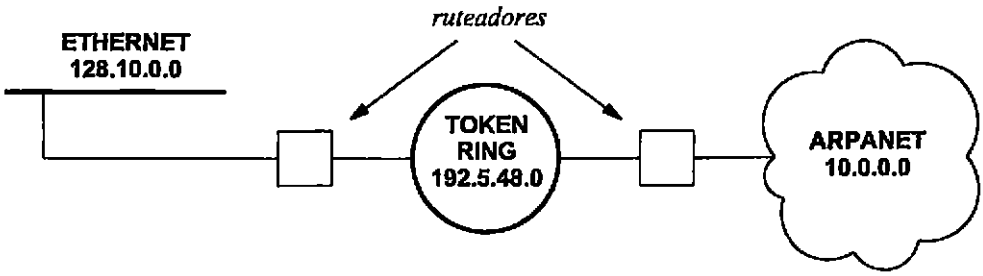


Figura 4.5 Conexión lógica de dos redes a la columna vertebral Internet. Cada red tiene asignada una dirección IP.

redes Ethernet y token ring, así que puede alcanzar directamente destinos en cualquiera de ellas. Aunque un anfitrión multi-homed como *Merlin* se puede configurar para rutear paquetes entre dos redes, la mayor parte de los sitios utilizan computadoras dedicadas como ruteadores para evitar so-

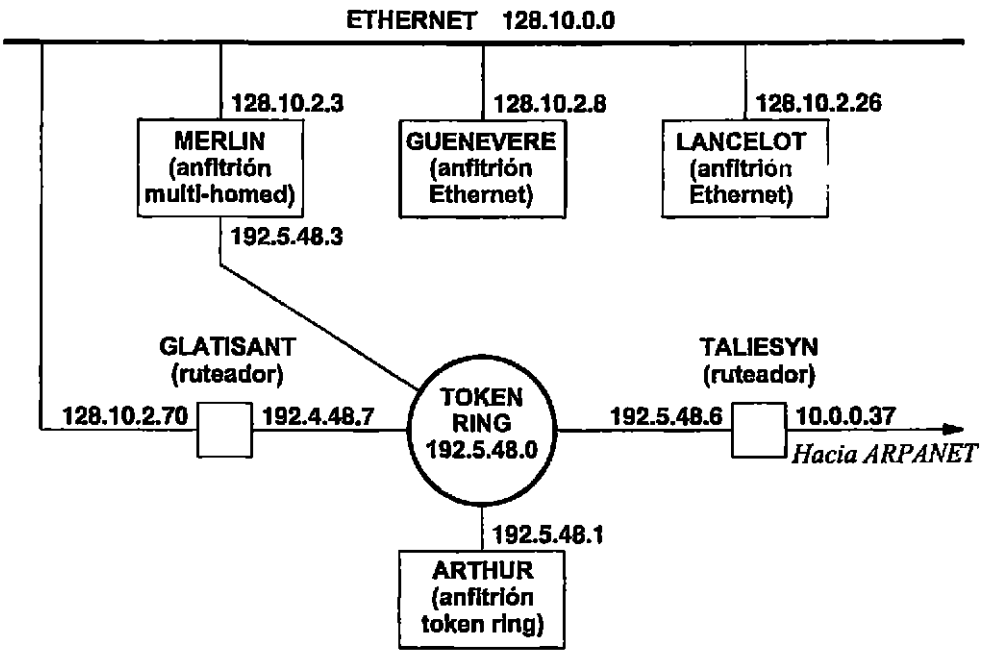


Figura 4.6 Ejemplo de asignación de direcciones IP para ruteadores y anfitriones conectados a las tres redes mostradas en la figura anterior.

brecargar los sistemas convencionales de computadoras con el procesamiento requerido para el ruteo. En la figura, un ruteador dedicado, *Glatisant*, realiza la tarea de rutear el tráfico entre las redes Ethernet y token ring. (Nota: el tráfico real entre estas dos redes es mayor que el que se muestra con esta configuración, debido a que en la figura sólo aparecen unas cuantas computadoras conectadas a las redes.)

Como se muestra en la figura 4.5, se debe asignar una dirección IP a cada conexión de red. *Lancelot*, que sólo se conecta a Ethernet, tiene asignada como su única dirección IP el número 128.10.2.26. *Merlin* tiene la dirección 128.10.2.3 para su conexión con Ethernet y 192.5.48.3 para su conexión con la red token ring. La persona que hizo la asignación de direcciones escogió el mismo valor para el octeto de orden bajo de cada dirección. Las direcciones asignadas a los ruteadores *Glatisant* y *Taliesyn* no siguen esta regla. Por ejemplo, las direcciones de *Taliesyn* 10.0.0.37 y 192.5.48.6, son dos cadenas de dígitos sin ninguna relación. Al IP no le importa si cualquier octeto en la forma decimal con puntos de una dirección de computadora es igual o diferente. Sin embargo, los técnicos, gerentes y administradores de red quizá necesiten utilizar las direcciones para mantenimiento, prueba y depuración. Hacer que todas las direcciones de las computadoras terminen con el mismo octeto facilita que las personas recuerden o adivinen la dirección de una interfaz en particular.

4.14 Orden de octetos de red

Para crear una red de redes que sea independiente de cualquier arquitectura de máquina de una marca en especial o de cualquier hardware de red, el software debe definir una representación de datos estándar. Por ejemplo, considere lo que sucede cuando el software de una computadora envía un entero binario de 32 bits a otra computadora. El hardware físico de transporte mueve la secuencia de bits desde la primera máquina hasta la segunda sin cambiar el orden. Sin embargo, no todas las máquinas almacenan de la misma forma los enteros de 32 bits. En algunas (llamadas *Little Endians*), la dirección más baja de memoria contiene el octeto de orden bajo del entero. En otras (llamadas *Big Endian*), la dirección más baja de memoria guarda el octeto de orden alto del entero. Otras almacenan los enteros en grupos de palabras de 16 bits, con las direcciones más bajas guardando la palabra de orden inferior, pero con los octetos desordenados. Por lo tanto, la copia directa de octetos de una máquina a otra puede cambiar el valor del número.

La estandarización del orden de octetos para los enteros es muy importante en una red de redes ya que los paquetes llevan números binarios que especifican información como las direcciones de destino y la longitud de los paquetes. Tales cantidades deben entenderlas tanto el receptor como el transmisor. Los protocolos TCP/IP resuelven el problema del orden de octetos al definir un *orden de octetos estándar de red* que todas las máquinas utilizan para los campos binarios en los paquetes de red de redes. Cada anfitrión o ruteador convierte los artículos binarios de la representación local al orden de octetos estándar de red antes de enviar un paquete y los convierte del orden de octeto estándar de red al orden específico del anfitrión cuando llega un paquete. Desde luego, el campo de datos del usuario de un paquete está exento de este estándar —los usuarios pueden dar formato a sus datos de la manera que quieran. Por supuesto, la mayoría de los usuarios confía en programas estándar de aplicación y no tiene que manejar directamente el problema del orden de octetos.

El estándar de red de redes para el orden de octetos especifica que la parte de los enteros que se envía primero es el octeto más significativo (por ejemplo, el tipo *Big Endian*). Si se consideran los octetos en un paquete mientras viajan de una máquina a otra, un entero binario en dicho paquete tiene su octeto más significativo cerca del comienzo del paquete y su octeto menos significativo cerca del final. Existen muchos argumentos sobre qué representación de datos se debe utilizar y el estándar de red de redes algunas veces se ve atacado. Sin embargo, todo el mundo está de acuerdo en que es crucial tener un estándar y la forma exacta del estándar es mucho menos importante.

4.15 Resumen

El TCP/IP utiliza direcciones binarias de 32 bits como identificadores universales de máquinas. Llamados direcciones IP o direcciones de red de redes, los identificadores se dividen en tres tipos principales. Debido a que los bits guía definen el tipo de dirección, dicho tipo no tiene el mismo tamaño. El esquema IP de direccionamiento permite que existan unos cuantos cientos de redes con más de un millón de anfitriones cada una, miles de redes con miles de anfitriones cada una, y más de un millón de redes con hasta 254 anfitriones cada una. Para hacer que sea más fácil para una persona entender dichas direcciones, están escritas en notación decimal con puntos, con los valores de los cuatro octetos escritos en números decimales y separados por puntos.

Ya que la dirección IP codifica la identificación de red, así como la identificación de un anfitrión específico en dicha red, el ruteo es eficiente. Una característica importante de una dirección IP es que se refiere a las conexiones de red. Los anfitriones con muchas conexiones tienen muchas direcciones. Una ventaja del esquema de direccionamiento en red de redes es que el formato incluye una dirección para un anfitrión específico, una red o todos los anfitriones dentro de una red (difusión). La mayor desventaja del esquema de direccionamiento IP es que si una máquina tiene muchas direcciones, saber una dirección no será suficiente para alcanzarla cuando no exista un camino hacia la interfaz especificada (por ejemplo, si una red en particular no está disponible).

Para permitir el intercambio de datos binarios entre máquinas, los protocolos TCP/IP requieren del ordenamiento estándar de octetos para los enteros dentro de los campos del protocolo. Un anfitrión debe convertir todos los datos binarios de su forma interna a un orden estándar de octetos de red antes de enviar un paquete y debe hacer la conversión de orden de octeto de red al orden interno cuando reciba paquetes.

PARA CONOCER MÁS

El esquema de direccionamiento en red de redes que aquí presentamos se puede encontrar en Reynolds and Postel (RFC 1700): se puede obtener mayor información en Stahl, Romano, and Recker (RFC 1117).

Se han realizado muchas adiciones importantes al esquema de direccionamiento de Internet a lo largo de los años; en los siguientes capítulos se examinan con mayor detalle. En el capítulo 10 se analiza una idea en evolución llamada *direccionamiento sin tipo*, un esquema intermedio de direccionamiento, diseñado para utilizarse durante los próximos años. Además, en el capítulo 10 se exa-

mina una parte esencial del estándar existente de direcciones Internet, llamado *direccionamiento de subred*. El direccionamiento de subred permite que una sola dirección de red se utilice con muchas redes físicas. En el capítulo 17 se continúa la exploración de las direcciones IP al describir cómo las direcciones tipo D se asignan para la *multidifusión* en la red de redes.

INTERNIC puede proporcionar información sobre cómo obtener direcciones (ver el Apéndice 1 para obtener la dirección y número telefónico de INTERNIC). Cohen (1981) explica el ordenamiento de octetos y bits, y presenta los términos "Big Endian" y "Little Endian".

EJERCICIOS

4.1 ¿Exactamente cuántas redes tipos *A*, *B* y *C* pueden existir? ¿Cuántos anfitriones puede tener una red de cada tipo? Asegúrese de permitir la difusión, así como direcciones tipo *D* y *E*.

4.2 A una lista de direcciones asignadas a veces se le conoce como *tabla de anfitriones* de red de redes. Si su sitio cuenta con una tabla de anfitriones, encuentre cuántos números de redes tipo *A*, *B* y *C* se han asignado.

4.3 ¿Cuántos anfitriones están conectados a cada una de las redes de área local en su sitio? ¿Su sitio tiene alguna red de área local para la que una dirección tipo *C* sea insuficiente?

4.4 ¿Cuál es la principal diferencia entre el esquema IP de direccionamiento y el esquema de asignación de números telefónicos de Estados Unidos?

4.5 Una sola autoridad central no se las puede arreglar para asignar direcciones Internet lo suficientemente rápido como para satisfacer toda la demanda. ¿Puede inventar un esquema que permita que la autoridad central divida sus tareas entre muchos grupos, pero que aún así asegure que cada dirección asignada es única?

4.6 ¿Hay diferencia entre el orden estándar de octetos y el orden de octetos en su máquina local?

4.7 ¿Cuántas direcciones IP se necesitarían para asignar un número único de red a cada hogar en su país? ¿Es suficiente el espacio de la dirección IP?

5

Transformación de direcciones Internet en direcciones físicas (ARP)

5.1 Introducción

Hemos descrito el esquema de direcciones TCP/IP, en el que cada anfitrión tiene asignada una dirección de 32 bits; asimismo, hemos dicho que una red de redes se comporta como una red virtual que utiliza sólo direcciones asignadas cuando envía y recibe paquetes. También hemos revisado muchas tecnologías de redes físicas y hemos notado que dos máquinas, en una red física, se pueden comunicar *solamente si conocen sus direcciones físicas de red*. Lo que no hemos mencionado es cómo un anfitrión o un ruteador transforman una dirección IP en la dirección física correcta cuando necesitan enviar un paquete a través de una red física. En este capítulo, se considera dicha transformación y se muestra de qué manera se implementa para los dos esquemas más comunes de direccionamiento de red física.

5.2 El problema de la asociación de direcciones

Considere que dos máquinas, A y B, comparten una red física. Cada una tiene asignada una dirección IP, I_A e I_B, así como una dirección física, P_A y P_B. El objetivo es diseñar un software de bajo nivel que oculte las direcciones físicas y permita que programas de un nivel más alto trabajen sólo

con direcciones de la red de redes. Sin embargo, la comunicación debe llevarse a cabo por medio de redes físicas, utilizando cualquier esquema de direcciones físicas proporcionado por el hardware. Suponga que la máquina *A* quiere enviar un paquete a la máquina *B* a través de una red física a la que ambas se conectan, pero *A* sólo tiene la dirección de red de redes I_B de *B*. Surge, pues, la siguiente pregunta: ¿cómo transforma *A* dicha dirección en la dirección física P_B de *B*?

La transformación de direcciones se tiene que realizar en cada fase a lo largo del camino, desde la fuente original hasta el destino final. En particular, surgen dos casos. Primero, en la última fase de entrega de un paquete, éste se debe enviar a través de una red física hacia su destino final. La computadora que envía el paquete tiene que transformar la dirección Internet de destino final en su dirección física. Segundo, en cualquier punto del camino, de la fuente al destino, que no sea la fase final, el paquete se debe enviar hacia un ruteador intermedio. Por lo tanto, el transmisor tiene que transformar la dirección Internet del ruteador en una dirección física.

El problema de transformar direcciones de alto nivel en direcciones físicas se conoce como *problema de asociación de direcciones* y se ha resuelto de muchas maneras. Algunos grupos de protocolos cuentan con tablas en cada máquina que contienen pares de direcciones, de alto nivel y físicas. Otros solucionan el problema al codificar direcciones de hardware en direcciones de alto nivel. Basarse en cualquiera de estos enfoques sólo hace que el direccionamiento de alto nivel sea muy delicado. En este capítulo, se tratan dos técnicas para la definición de direcciones utilizadas por los protocolos TCP/IP y se muestra cuándo es apropiada cada una de ellas.

5.3 Dos tipos de direcciones físicas

Existen dos tipos básicos de direcciones físicas, ejemplificados por Ethernet que tiene direcciones físicas grandes y fijas, así como por proNET que tiene direcciones físicas cortas y de fácil configuración. La asociación de direcciones es difícil para las redes de tipo Ethernet, pero resulta sencilla para redes como proNET. Consideraremos, primero, el caso más fácil.

5.4 Asociación mediante transformación directa

Considere una red token ring tipo proNET. Recuerde que, en el capítulo 2, vimos que proNET utiliza números enteros pequeños para sus direcciones físicas y permite que el usuario elija una dirección de hardware cuando instala una tarjeta de interfaz en una computadora. La clave para facilitar la definición de direcciones con dicho hardware de red radica en observar que, mientras se tenga la libertad de escoger tanto la dirección IP como la física, se puede hacer que ambas posean las mismas partes. Normalmente, una persona asigna direcciones IP con el campo hostid igual a 1, 2, 3, etcétera, y luego, cuando instala hardware de interfaz de red, selecciona una dirección física que corresponda a la dirección IP. Por ejemplo, el administrador de sistema podría seleccionar la dirección física 3 para una computadora que tenga la dirección IP 192.5.48.3, debido a que la dirección anterior es tipo *C* y tiene el campo de anfitrión igual a 3.

Para las redes como proNET, computar una dirección física basándose en una dirección IP es trivial. El cómputo consiste en extraer el campo de anfitrión de la dirección IP. La extracción es

computacionalmente eficiente pues sólo necesita unas cuantas instrucciones de máquina. La transformación es fácil de mantener porque se puede realizar sin consultar datos externos. Por último, es posible agregar nuevas máquinas a la red sin cambiar las asignaciones ya existentes ni recopilar los códigos.

Conceptualmente, escoger un esquema de numeración que facilite la asociación de direcciones significa seleccionar una función f que transforme direcciones IP en direcciones físicas. El diseñador también puede ser capaz de seleccionar un esquema de numeración para direcciones físicas, dependiendo del hardware. Definir la dirección IP, I_A, implica computar:

$$P_A = f(I_A)$$

Queremos que el cómputo de f sea eficiente. Si se constriñe el juego de direcciones físicas, puede ser posible realizar transformaciones eficientes, diferentes a la que se ejemplifica arriba. Por ejemplo, cuando se utiliza el IP en una red orientada a la conexión como ATM, no se pueden escoger las direcciones físicas. En redes como esa, una o más computadoras almacenan pares de direcciones, en donde cada par contiene una dirección Internet y su dirección física correspondiente. Por ejemplo, los valores se pueden almacenar dentro de una tabla en memoria, que se tiene que buscar. Para lograr que, en esos casos, la definición de direcciones sea eficaz, el software podría valerse de una función convencional de comprobación aleatoria para buscar dentro de la tabla. En el ejercicio 5.1, se sugiere una alternativa relacionada.

5.5 Definición mediante enlace dinámico

Para entender por qué la definición de direcciones es difícil para algunas redes, consideremos la tecnología Ethernet. Recuerde que, en el capítulo 2, vimos que cada interfaz Ethernet tiene asignada una dirección física de 48 bits desde la fabricación del producto. En consecuencia, cuando el hardware falla y se necesita reemplazar una interfaz Ethernet, la dirección física de la máquina cambia. Además, como la dirección Ethernet es de 48 bits, no hay posibilidad de codificarla en una dirección IP de 32 bits.[1]

Los diseñadores de los protocolos TCP/IP encontraron una solución creativa para el problema de la asociación de direcciones en redes como Ethernet, que tienen capacidad de difusión. La solución permite agregar nuevas máquinas a la red, sin tener que recopilar el código y no requiere tener una base de datos centralizada. Para evitar la definición de una tabla de conversiones, los diseñadores utilizan un protocolo de bajo nivel para asignar direcciones en forma dinámica. Conocido como *Protocolo de Asociación de Direcciones (ARP)*, éste proporciona un mecanismo razonablemente eficaz y fácil de mantener.

Como se muestra en la figura 5.1, la idea detrás de la asociación dinámica con ARP es muy sencilla: cuando el anfitrión A quiere definir la dirección IP, I_B, transmite por difusión un paquete especial que pide al anfitrión que posee la dirección IP I_B, que responda con su dirección física, P_B. Todos los anfitriones, incluyendo a B, reciben la solicitud, pero sólo el anfitrión B reconoce su propia dirección IP y envía una respuesta que contiene su dirección física. Cuando A recibe la respues-

[1] Debido a que la transformación directa es más conveniente y eficiente que la asignación dinámica, la próxima generación de IP se está diseñando para permitir que las direcciones de 48 bits se puedan codificar en direcciones IP.

ta, utiliza la dirección física para enviar el paquete de red de redes directamente a *B*. Se puede resumir que:

El Protocolo de Asociación de Direcciones ARP permite que un anfitrión encuentre la dirección física de otro anfitrión dentro de la misma red física con sólo proporcionar la dirección IP de su objetivo.

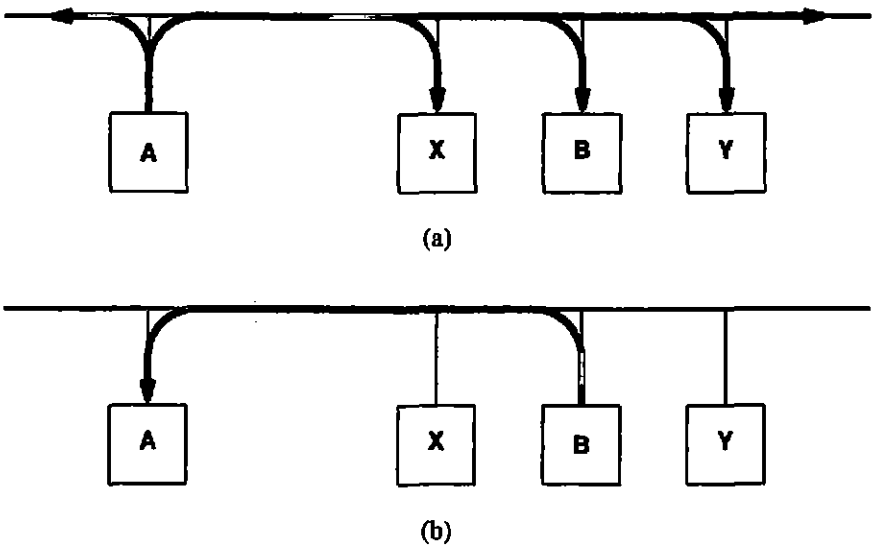


Figura 5.1 Protocolo ARP. Para determinar la dirección física P_B de *B*, desde su dirección IP, I_B, (a) el anfitrión *A* transmite por difusión una solicitud ARP que contiene I_B a todas las máquinas en la red, y (b) el anfitrión *B* envía una respuesta ARP que contiene el par (I_B, P_B).

5.6 Memoria intermedia para asociación de direcciones

Puede parecer extraño que para que *A* envíe un paquete a *B*, primero, tenga que transmitir una difusión que llegue a *B*. Podría parecer aún más extraño que *A* transmita por difusión la pregunta ¿cómo puedo llegar hasta ti?, en lugar de sólo transmitir por difusión el paquete que quiere entregar. Pero existe una razón importante para este intercambio. La difusión es demasiado cara para utilizarse cada vez que una máquina necesita transmitir un paquete a otra, debido a que requiere que cada máquina en la red procese dicho paquete. Para reducir los costos de comunicación, las computadoras, que utilizan ARP, mantienen una memoria intermedia de las asignaciones de dirección IP a dirección física recientemente adquiridas, para que no tengan que utilizar ARP varias veces. Siempre que una computadora recibe una respuesta ARP, ésta guarda la dirección IP del transmisor, así como la dirección de hardware correspondiente, en su memoria intermedia, para utilizarla en búsquedas posteriores. Cuando transmite un paquete, una computadora siempre busca, en su memoria intermedia, una asignación antes de enviar una solicitud ARP. Si una computadora en-

cuentra la asignación deseada en su memoria intermedia ARP, no necesitan transmitir una difusión a la red. La experiencia nos indica que, como la mayor parte de la comunicación en red comprende más que la sola transferencia de un paquete, hasta una memoria intermedia pequeña es muy valiosa.

5.7 Refinamientos ARP

Se pueden lograr muchos refinamientos de ARP. Primero, observe que si el anfitrión *A* va a utilizar ARP porque necesita enviar algo a *B*, existe una alta posibilidad de que *B* necesite enviar algo a *A* en un futuro cercano. Para anticipar la necesidad de *B* y evitar tráfico de red adicional, *A* incluye su asignación de dirección IP como dirección física cuando envía una solicitud a *B*. *B* extrae la asignación de *A* de la solicitud, la graba en su memoria intermedia ARP y envía la respuesta hacia *A*. Segundo, nótese que, debido a que *A* transmite por difusión su solicitud inicial, todas las máquinas en la red la reciben y pueden extraer, así como grabar, en su memoria intermedia, la asignación de dirección IP como dirección física de *A*. Tercero, cuando a una máquina se le reemplaza la interfaz de anfitrión (por ejemplo, a causa de una falla en el hardware), su dirección física cambia. Las otras computadoras en la red, que tienen almacenada una asignación en su memoria intermedia ARP, necesitan ser informadas para que puedan cambiar el registro. Un sistema puede notificar a otros sobre una nueva dirección al enviar una difusión ARP cuando se inicia.

La siguiente regla resume los refinamientos:

> *El transmisor incluye, en cada difusión ARP, su asignación de dirección IP como dirección física; los receptores actualizan su información en memoria intermedia antes de procesar un paquete ARP.*

5.8 Relación de ARP con otros protocolos

ARP proporciona un mecanismo para transformar direcciones IP en direcciones físicas; ya hemos visto que algunas tecnologías de red no lo necesitan. El punto es que ARP sería totalmente innecesario si pudiéramos hacer que todo el hardware de red reconociera direcciones IP. Por lo tanto, ARP sólo impone un nuevo esquema de direccionamiento sobre cualquier mecanismo de direccionamiento de bajo nivel que el hardware utilice. La idea se puede resumir de la siguiente manera:

> *ARP es un protocolo de bajo nivel que oculta el direccionamiento físico subyacente de red, al permitir que se asigne una dirección IP arbitraria a cada máquina. Pensamos en ARP como parte del sistema físico de red, no como parte de los protocolos de red de redes.*

5.9 Implantación de ARP

De manera funcional, ARP está dividido en dos partes. La primera parte transforma una dirección IP en una dirección física cuando se envía un paquete y la segunda responde solicitudes de otras

máquinas. La definición de direcciones para los paquetes salientes parece muy clara, pero los pequeños detalles complican la implantación. Al tener una dirección IP de destino, el software consulta su memoria intermedia ARP para encontrar la transformación de la dirección IP a la dirección física. Si la conoce, el software extrae la dirección física, pone los datos en una trama utilizando esa dirección y envía la trama. Si no conoce la transformación, el software debe transmitir una difusión que contenga la solicitud ARP y esperar una respuesta.

La difusión de una solicitud ARP para encontrar una transformación de direcciones se puede volver compleja. La máquina de destino puede estar apagada o tan sólo muy ocupada para aceptar la solicitud. Si es así, el transmisor quizá no reciba la respuesta o la reciba con retraso. Debido a que Ethernet es un sistema de entrega con el mejor esfuerzo, también se puede perder la solicitud de difusión inicial ARP (en cuyo caso, el que la envía debe retransmitirla por lo menos una vez). Mientras tanto, el anfitrión tiene que almacenar el paquete original para que se pueda enviar ya que se haya asociado la dirección IP a la dirección de red.[2] De hecho, el anfitrión debe decidir si permite que otros programas de aplicación funcionen mientras realiza una solicitud ARP (la mayor parte de ellos lo permite). Si así es, el software debe manejar el hecho de que una aplicación genere solicitudes ARP adicionales para la misma dirección sin transmitir por difusión muchas solicitudes para un mismo objetivo.

Por último, considere el caso en el que la máquina *A* ya obtuvo una asignación para la máquina *B*, pero el hardware de *B* falla y es reemplazado. Aunque la dirección de *B* ha cambiado, las asignaciones en memoria temporal de *A* no lo han hecho, así que *A* utiliza una dirección de hardware que no existe, por lo que la recepción exitosa se vuelve imposible. En este caso se muestra por qué es importante tener software ARP que maneje de manera temporal la tabla de asignaciones y que remueva los registros después de un periodo establecido de tiempo. Claro está, el controlador de tiempo para un registro en la memoria temporal se debe reiniciar cada vez que llegue una difusión ARP que contenga la asignación (pero no se reinicia cuando el registro se utiliza para enviar un paquete).

La segunda parte del código ARP maneja paquetes que llegan por medio de la red. Cuando llega un paquete ARP, el software extrae la dirección IP del transmisor y la dirección del hardware, luego, examina la memoria temporal local para verificar si ya existe un registro para el transmisor. Si es así, el controlador actualiza el registro al sobreescribir la dirección física con la dirección obtenida del paquete. Después, el receptor procesa el resto del paquete ARP.

El receptor debe manejar dos tipos de paquetes ARP entrantes. Si llega una solicitud ARP, la máquina receptora debe verificar si es el objetivo de la solicitud (por ejemplo, si alguna otra máquina transmitió por difusión una solicitud de la dirección física del receptor). Si es así, el software ARP formula una respuesta al proporcionar su dirección física de hardware y la envía directamente al solicitante. El receptor también agrega el par de direcciones del transmisor a su memoria temporal si éstas no están presentes. Si la dirección IP mencionada en la solicitud ARP no corresponde a la dirección IP local, el paquete solicitará la transformación de alguna otra máquina en la red aunque podría ser ignorado.

El otro caso interesante sucede cuando llega una respuesta ARP. Dependiendo de la implantación, el controlador quizá necesite crear un registro en su memoria temporal o el registro se pueda crear cuando se genere la solicitud. En cualquiera de estos casos, una vez que se actualiza la memoria temporal, el receptor intenta encontrar una correspondencia entre la respuesta y una solicitud expedida con anterioridad. Por lo general, las respuestas llegan obedeciendo a una solicitud

---

[2] Si el retraso es significativo, el anfitrión puede descartar los paquetes salientes.

que se generó porque la máquina tiene que entregar un paquete. Entre el tiempo en que una máquina transmite por difusión su solicitud ARP y recibe la respuesta, los programas de aplicación o los protocolos de un nivel más alto pueden generar solicitudes adicionales para la misma dirección; el software debe recordar que ya envió una solicitud para no enviar más. Por lo común, el software ARP coloca los paquetes adicionales en una cola de espera. Una vez que llega la respuesta y se conoce la asignación de dirección, el software ARP remueve los paquetes de la cola de espera, pone cada paquete en una trama y utiliza la asignación de dirección para llenar la dirección física del destino. Si, con anterioridad, no expidió una solicitud de la dirección IP en la respuesta, la máquina actualizará el registro del transmisor en su memoria temporal y tan sólo dejará de procesar el paquete.

5.10 Encapsulación e identificación de ARP

Cuando los mensajes ARP viajan de una máquina a otra, se deben transportar en tramas físicas. En la figura 5.2, se muestra cómo se transporta el mensaje ARP en la porción de datos de una trama.

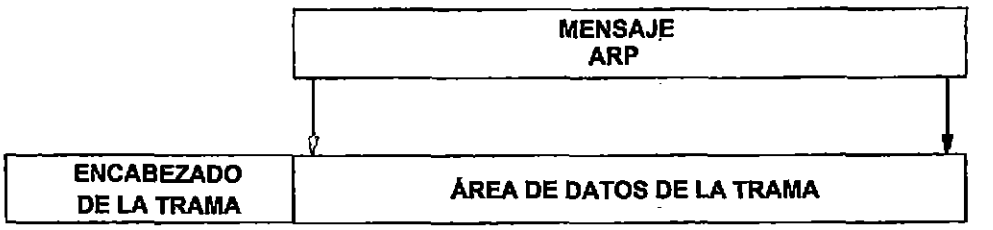


Figura 5.2 Mensaje ARP encapsulado en una trama de red física.

Para identificar que la trama transporta una mensaje ARP, el transmisor asigna un valor especial al campo de tipo en el encabezado de la trama y coloca el mensaje ARP en el campo de datos de la misma. Cuando llega una trama a una computadora, el software de red utiliza el campo de tipo de trama para determinar su contenido. En la mayor parte de las tecnologías, se utiliza un solo valor para el tipo de todas las tramas que transportan un mensaje ARP —el software de red en el receptor debe examinar el mensaje ARP para distinguir entre solicitudes y respuestas. Por ejemplo, en una Ethernet, las tramas que transportan mensajes ARP tienen un campo de tipo de 0806_{16}. Este es un valor estándar asignado por la autoridad para Ethernet; otras tecnologías de hardware de red emplean otros valores.

5.11 Formato del protocolo ARP

A diferencia de la mayor parte de los protocolos, los datos en los paquetes ARP no tienen un encabezado con formato fijo. Por el contrario, para hacer que ARP sea útil para varias tecnologías de

red, la longitud de los campos que contienen direcciones depende del tipo de red. Sin embargo, para hacer posible la interpretación de un mensaje ARP arbitrario, el encabezado incluye campos fijos cerca del comienzo, que especifican la longitud de las direcciones que se encuentran en los campos siguientes. De hecho, el formato de un mensaje ARP es lo suficientemente general como para permitir que sea utilizado con direcciones físicas arbitrarias y direcciones arbitrarias de protocolos. En el ejemplo de la figura 5.3 se muestra el formato de 28 octetos de un mensaje ARP que se utiliza en el hardware Ethernet (en el que las direcciones físicas tienen una longitud de 48 bits o de 6 octetos), cuando se asocian direcciones de protocolo IP (que tienen una longitud de 4 octetos).

En la figura 5.3, se muestra un mensaje ARP con 4 octetos por línea, formato estándar a través de todo este texto. Por desgracia, a diferencia de la mayor parte de los otros protocolos, los campos de longitud variable en los paquetes ARP no se alinean firmemente en fronteras de 32 bits, lo cual causa que el diagrama sea difícil de leer. Por ejemplo, la dirección de hardware del transmisor, etiquetada como *SENDER HA,* ocupa 6 octetos contiguos, por lo que abarca dos líneas en el diagrama.

0	8	16	24 31
TIPO DE HARDWARE		TIPO DE PROTOCOLO	
HLEN	PLEN	OPERACIÓN	
SENDER HA (octetos 0-3)			
SENDER HA (octetos 4-5)		SENDER IP (octetos 0-1)	
SENDER IP (octetos 2-3)		TARGET HA (octetos 0-1)	
TARGET HA (octetos 2-5)			
TARGET IP (octetos 0-3)			

Figura 5.3 Ejemplo del formato de mensaje ARP/RARP cuando se utiliza para la transformación de una dirección IP en una dirección Ethernet. La longitud de los campos depende del hardware y de la longitud de las direcciones de protocolos, que son de 6 octetos para una dirección Ethernet y de 4 octetos para una dirección IP.

El campo *HARDWARE TYPE* especifica un tipo de interfaz de hardware para el que el transmisor busca una respuesta; contiene el valor *1* para Ehternet. De forma similar, el campo *PROTOCOL TYPE* especifica el tipo de dirección de protocolo de alto nivel que proporcionó el transmisor: contiene 0800_{16} para la dirección IP. El campo *OPERATION* especifica una solicitud ARP (*1*), una respuesta ARP (*2*), una solicitud RARP[3] (*3*) o una respuesta RARP (*4*). Los campos *HLEN* y *PLEN* permiten que ARP se utilice con redes arbitrarias ya que éstas especifican la longitud de la dirección de hardware y la longitud de la dirección del protocolo de alto nivel. El transmisor proporciona sus direcciones IP y de hardware, si las conoce, en los campos *SENDER HA* y *SENDER IP*.

[3] En el siguiente capítulo se describe RARP, otro protocolo que utiliza el mismo formato de mensajes.

Cuando realiza una solicitud, el transmisor también proporciona la dirección IP del objetivo (ARP) o la dirección de hardware del objetivo (RARP), utilizando los campos *TARGET HA* y *TARGET IP*. Antes de que la máquina objetivo responda, completa las direcciones faltantes, voltea los pares de objetivo y transmisor, y cambia la operación a respuesta. Por lo tanto, una respuesta transporta las direcciones tanto de hardware como de IP del solicitante original, lo mismo que las direcciones de hardware e IP de la máquina para la que se realizó asignación.

5.12 Resumen

Las direcciones IP se asignan independientemente de la dirección física de hardware de una máquina. Para enviar un paquete de red de redes a través de una red física desde una máquina hacia otra, el software de red debe transformar la dirección IP en una dirección física de hardware y utilizar esta última para transmitir la trama. Si las direcciones de hardware son más pequeñas que las direcciones IP, se puede establecer una transformación directa al codificar la dirección física de una máquina dentro de su dirección IP. De otra forma, la transformación debe realizarse de manera dinámica. El Protocolo de Definición de Direcciones (ARP) realiza la definición dinámica de direcciones, utilizando sólo el sistema de comunicación de red de bajo nivel. ARP permite que las máquinas asocien direcciones sin tener un registro permanente de asignaciones.

Una máquina utiliza ARP para encontrar la dirección de hardware de otra máquina al transmitir por difusión una solicitud ARP. La solicitud contiene la dirección IP de la máquina de la que se necesita la dirección de hardware. Todas las máquinas en una red reciben la solicitud ARP. Si la solicitud corresponde a la dirección IP de una máquina, ésta responde al enviar una respuesta que contiene la dirección de hardware requerida. Las respuestas se dirigen a una sola máquina; no se transmiten por difusión.

Para lograr que ARP sea eficiente, cada máquina guarda en su memoria temporal las asignaciones de dirección IP a dirección física. Como el tráfico de una red de redes tiende a ser una secuencia de interacciones entre pares de máquinas, la memoria temporal elimina la mayor parte de las solicitudes ARP transmitidas por difusión.

PARA CONOCER MÁS

El protocolo de definición de direcciones que aquí se utiliza está proporcionado por Plummer (RFC 826) y se ha convertido en un estándar de protocolos TCP/IP para red de redes. Dalal y Printis (1981) describen la relación entre las direcciones IP y las direcciones Ethernet; asimismo, Clark (RFC 814) trata en general las direcciones y las asignaciones. Parr (RFC 1029) analiza la definición de direcciones tolerante de fallas. Kirkpatrick y Recker (RFC 1166) especifican valores utilizados para identificar tramas de red en el documento Internet Numbers. En el volumen II de esta obra, se presenta el ejemplo de una ejecución ARP y se analiza el procedimiento respecto a la memoria temporal.

EJERCICIOS

5.1 Teniendo un pequeño grupo de direcciones físicas (números enteros positivos), ¿puede encontrar una función *f* y una asignación de direcciones IP, de tal forma que *f* transforme las direcciones IP, una por una, en direcciones físicas y que el cómputo de *f* sea eficiente? Pista: consulte la documentación sobre dispersión perfecta, (*perfect hasshing*).

5.2 ¿En qué casos especiales un anfitrión conectado a una Ethernet no necesita utilizar ARP o una memoria temporal ARP antes de transmitir un datagrama IP?

5.3 Un algoritmo común para manejar la memoria temporal ARP reemplaza el registro menos utilizado cuando agrega uno nuevo. ¿Bajo qué circunstancias este algoritmo podría generar un tráfico de red innecesario?

5.4 Lea cuidadosamente el estándar. ¿ARP debe actualizar la memoria intermedia si ya existe un registro antiguo para cierta dirección IP? ¿Por qué?

5.5 ¿El software ARP debe modificar la memoria intermedia inclusive cuando recibe información sin solicitarla de manera específica? ¿Por qué?

5.6 Cualquier implantación ARP que utilice una memoria temporal de tamaño fijo puede fallar cuando se utiliza en una red que tiene muchos anfitriones y mucho tráfico ARP. Explique cómo.

5.7 A veces, se refieren a ARP como una debilidad de seguridad. Explique por qué.

5.8 Explique qué puede pasar si el campo de dirección de hardware en una respuesta ARP se corrompe durante la transmisión. Pista: algunas implantaciones ARP no remueven los registros en memoria temporal si se utilizan con frecuencia.

5.9 Suponga que la máquina *C* recibe una solicitud ARP de *A* buscando al objetivo *B*, y suponga que *C* tiene la asignación de I_B a P_B en su memoria temporal. ¿*C* debe contestar la solicitud? Explíquelo.

5.10 ¿Cómo puede utilizar ARP una estación de trabajo cuando se inicia para descubrir si alguna otra máquina en la red la está personificando? ¿Cuáles son las desventajas del esquema?

5.11 Explique de qué manera el envío de paquetes hacia direcciones no existentes en una Ethernet remota puede generar tráfico de difusión excesivo en esa red.

6

Determinación en el arranque de una dirección Internet (RARP)

6.1 Introducción

Hasta ahora sabemos que las direcciones físicas de red son de bajo nivel y dependientes del hardware. Asimismo, entendemos que cada máquina que utiliza el TCP/IP tiene asignada una o más direcciones IP de 32 bits, independientemente de su dirección de hardware. Los programas de aplicación siempre utilizan la dirección IP cuando especifican un destino. Los anfitriones y los ruteadores deben utilizar direcciones físicas para transmitir datagramas a través de las redes subyacentes; confían en los esquemas de asociación de direcciones como ARP para realizar los enlaces.

Por lo general, la dirección IP de una máquina se mantiene en el área secundaria de almacenaje, en donde el sistema operativo la encuentra en el momento del arranque. Ahora bien, surge la siguiente pregunta, ¿cómo puede una máquina que no cuenta con disco permanente determinar su dirección IP? El problema es crítico para las estaciones de trabajo que almacenan sus archivos en un servidor remoto, ya que dichas máquinas necesitan una dirección IP antes de poder utilizar protocolos TCP/IP estándar para transferencia de archivos a fin de obtener su imagen inicial de arranque. En este capítulo, se examina la cuestión de cómo obtener una dirección IP y se describe el protocolo que muchas máquinas utilizan antes del arranque desde un servidor remoto de archivos.

Debido a que una imagen de sistema operativo que tiene una dirección IP específica, limitada dentro del código, no se puede utilizar en muchas computadoras, los diseñadores por lo general tratan de evitar la compilación de una dirección IP en el código del sistema operativo o dentro del software de apoyo. En particular, el código de iniciación que se encuentra, a menudo, en la Memoria de Sólo Lectura (ROM), generalmente se construye para que la misma imagen pueda correr en

muchas máquinas. Cuando un código así inicia su ejecución, en una máquina sin disco, utiliza el hardware para contactar un servidor y, con ello, obtener su dirección IP.

El proceso de iniciación parece paradójico: una máquina se comunica con un servidor remoto a fin de obtener la dirección que necesita para la comunicación. Sin embargo, esta paradoja es sólo aparente, ya que la máquina *sabe* cómo comunicarse. Puede utilizar su dirección física para comunicarse a través de una sola red. Por tanto, la máquina debe recurrir de manera temporal al direccionamiento físico de red, de la misma forma en que el sistema operativo utiliza el direccionamiento físico de memoria para establecer tablas de página para el direccionamiento virtual. Una vez que la máquina conoce su dirección IP, se puede comunicar a través de una red de redes.

La idea detrás de encontrar una dirección IP es sencilla: una máquina que necesita conocer su dirección envía una solicitud a un *servidor*[1] en otra máquina y espera a que el servidor, a su vez, mande la respuesta. Asumimos que el servidor accesa un disco en el que guarda una base de datos de las direcciones internas. En la solicitud, sólo la máquina que necesita saber su dirección de red de redes se tiene que identificar, para que el servidor pueda buscar la dirección correcta y responder. Tanto la máquina que genera la solicitud como el servidor que responde utilizan direcciones físicas de red durante su breve comunicación. ¿Cómo sabe el solicitante la dirección física de un servidor? Por lo general, no lo sabe tan sólo transmite por difusión la solicitud a todas las máquinas de la red local. Y entonces, uno o más servidores responden.

Una máquina que transmite por difusión una solicitud de dirección sólo tiene que identificarse. ¿Qué información se puede incluir en esa solicitud que únicamente identificará a la máquina? Cualquier sufijo único para la identificación del hardware (por ejemplo, el número serial de la CPU). Sin embargo, la identificación tiene que basarse en algo que un programa en ejecución pueda obtener con facilidad. El objetivo es crear una sola imagen de software que pueda ejecutarse en un procesador cualquiera. Además, la longitud o el formato de la información específica de la CPU puede variar entre los diferentes modelos de procesadores, y nos gustaría imaginar un servidor que acepte solicitudes realizadas por todas las máquinas en la red por medio del uso de un solo formato.

6.2 Protocolo de asociación de direcciones por réplica (RARP)

Los diseñadores de los protocolos TCP/IP se dieron cuenta de que ya existe otra pieza disponible para la identificación exclusiva, a saber, la dirección física de red de la máquina. Utilizar la dirección física como identificación única tiene dos ventajas. Debido a que un anfitrión obtiene sus direcciones físicas del hardware de interfaz de red, dichas direcciones siempre están disponibles y no tienen que limitarse al código de iniciación. Como la información de identificación depende de la red y no del modelo o la marca de la CPU, todas las máquinas en una red proporcionarán identificadores únicos y uniformes. Por lo tanto, el problema se convierte en el inverso de la asociación de direcciones; una vez dada una dirección física de red, invente un esquema que permita que un servidor la transforme en una dirección de red de redes.

Una máquina sin disco utiliza un protocolo TCP/IP para red de redes llamado *RARP (Protocolo inverso de asociación de direcciones)* a fin de obtener su dirección IP desde un servidor. RARP es una adaptación al protocolo ARP que vimos en el capítulo anterior y utiliza el mismo formato de mensajes mostrado en la figura 5.3. En la práctica, el mensaje RARP enviado para solici-

[1] En el capítulo 19 se analizan los servidores más detalladamente.

tar una dirección de red de redes es un poco más general de lo que hemos subrayado arriba: permite que una máquina solicite la dirección IP de una tercera máquina tan fácilmente como si solicitara la suya. También lo permite cuando se trata de muchos tipos de redes físicas.

Al igual que un mensaje ARP, un mensaje RARP se envía de una máquina a otra, encapsulado en la porción de datos de una trama de red. Por ejemplo, una trama Ethernet que transporta una solicitud RARP tiene el preámbulo usual, las direcciones Ethernet tanto fuente como destino y campos de tipo de paquete al comienzo de la trama. El tipo de trama contiene el valor 8035_{16} para identificar que el contenido de la trama contiene un mensaje RARP. La porción de datos de la trama contiene el mensaje RARP de 28 octetos.

En la figura 6.1, se ilustra la manera en que un anfitrión utiliza RARP. El que envía transmite por difusión una solicitud RARP especificada como máquina transmisora y receptora, y proporciona su dirección física de red en el campo de dirección de hardware objetivo. Todas las máquinas en la red reciben la solicitud, pero sólo las autorizadas para proporcionar el servicio RARP la procesan y envían la respuesta; dichas máquinas se conocen de manera informal como *servidores RARP*. Para que RARP funcione correctamente, la red debe contener por lo menos un servidor RARP.

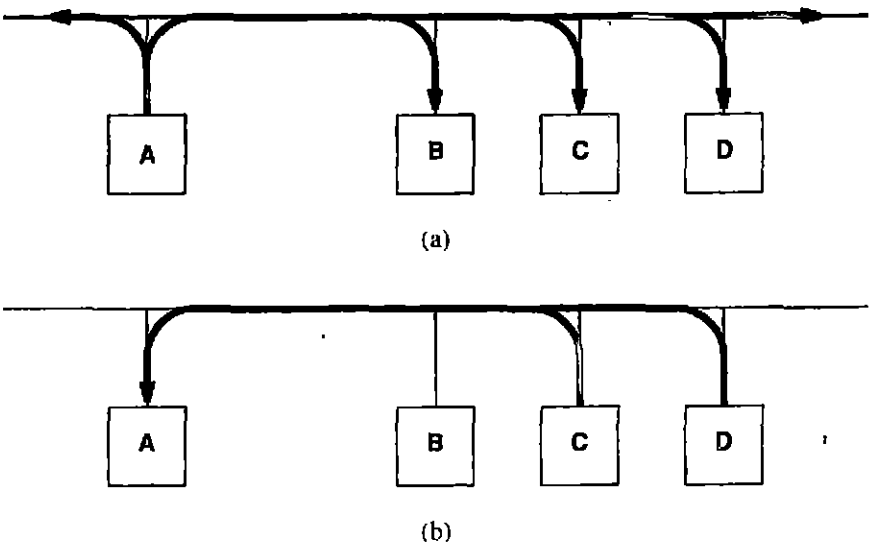


Figura 6.1 Ejemplo de un intercambio en el que se utiliza el protocolo RARP. (a) la máquina *A* transmite por difusión una solicitud RARP especificándose como destino y (b) las máquinas autorizadas para proporcionar el servicio RARP (*C* y *D*) responden directamente a *A*.

Una vez llenado el campo de dirección de protocolo objetivo, los servidores contestan las solicitudes, cambian el tipo de mensaje de *solicitud* a *respuesta* y envían ésta de vuelta directamente a la máquina que la solicitó. La máquina original recibe respuesta de todos los servidores RARP, aunque sólo se necesite una contestación.

Tenga en mente que toda la comunicación entre la máquina que busca su dirección IP y el servidor que la proporciona se debe llevar a cabo utilizando sólo una red física. Además, el protocolo permite que un anfitrión pregunte sobre un objetivo arbitrario. Por lo tanto, el transmisor proporciona su dirección de hardware separada de la dirección de hardware del objetivo y el servidor tiene cuidado de enviar la respuesta a la dirección de hardware del transmisor. En una Ethernet, tener un campo para la dirección de hardware del transmisor podría parecer redundante ya que la información también está contenida en el encabezado de la trama Ethernet. Sin embargo, no todo el hardware Ethernet proporciona al sistema operativo acceso al encabezado de la trama física.

6.3 Temporización de las transacciones RARP

Como cualquier comunicación en una red de entrega con el mejor esfuerzo, las solicitudes RARP son susceptibles de pérdida o corrupción. Ya que RARP utiliza directamente la red física, ningún otro software de protocolos cronometrará la respuesta ni retransmitirá la solicitud; es el software RARP el que debe manejar estas tareas. En general, RARP se utiliza sólo en redes de área local, como Ethernet, en las que la probabilidad de falla es muy baja. Sin embargo, si una red tiene sólo un servidor RARP, dicha máquina quizá no sea capaz de manejar la carga y, por tanto, los paquetes se pierdan.

Algunas estaciones de trabajo que dependen de RARP para realizar su proceso de iniciación reintentan éste una y otra vez hasta que reciben una respuesta. Otras implementaciones, al cabo de un par de intentos, lo suspenden indicando que hay fallas y evitan con ello inundar la red con tráfico innecesario de difusión (por ejemplo, en el caso de que el servidor no esté disponible). En una Ethernet, la falla de red no sucede solamente por la sobrecarga del servidor. Hacer que el software RARP retransmita rápidamente puede causar un efecto indeseable: inundar con más tráfico un servidor congestionado. Valerse de un retraso largo garantiza que los servidores tengan tiempo suficiente para satisfacer la solicitud y generar una respuesta.

6.4 Servidores RARP primarios y de respaldo

La principal ventaja de tener varias máquinas funcionando como servidores RARP es que se obtiene un sistema más confiable. Si un servidor falla, o está demasiado congestionado como para responder, otro servidor contestará la solicitud. Por tanto, es mucho más probable que el servicio siempre se encuentre disponible. La principal desventaja de utilizar varios servidores es que, cuando una máquina transmite por difusión una solicitud RARP, la red se sobrecarga en el momento en que todos los servidores intentan responder. Por ejemplo, en una Ethernet, emplear muchos servidores RARP ocasiona que la probabilidad de colisión sea muy alta.

¿Cómo se puede distribuir el servicio RARP para mantenerlo a disposición y confiable sin sufrir el costo por solicitudes excesivas y simultáneas? Existen, por lo menos, dos posibilidades y ambas implican el retraso de las respuestas. En la primera, a cada máquina que realiza solicitudes RARP se le asigna un *servidor primario*. Bajo circunstancias normales, sólo el servidor primario de la máquina responde a su solicitud RARP. Todos los servidores que no son primarios reciben la

solicitud, pero únicamente registran su tiempo de llegada. Si el servidor primario no está disponible, la máquina original cronometrará el tiempo de respuesta y, si ésta no aparece, transmitirá de nuevo por difusión la solicitud. Cuando un servidor no primario recibe una segunda copia de una solicitud RARP, poco tiempo después de la primera, éste responde.

En la segunda posibilidad, se emplea un esquema similar pero se intenta evitar que todos los servidores no primarios transmitan de manera simultánea las respuestas. Cada máquina no primaria que recibe una solicitud computa un retraso en forma aleatoria y, luego, envía la respuesta. Bajo circunstancias normales, el servidor primario responde de inmediato y las respuestas sucesivas se retrasan, así que existe una probabilidad muy baja de que lleguen al mismo tiempo. Cuando el servidor primario no está disponible, la máquina solicitante pasa por un corto retraso antes de recibir una respuesta. Al escoger con cuidado los tiempos de retraso, el diseñador puede asegurar que las máquinas solicitantes no hagan transmisiones por difusión antes de recibir una respuesta.

6.5 Resumen

En el arranque del sistema, una computadora que no tenga un disco permanente debe contactar a un servidor para encontrar su dirección IP antes de que se pueda comunicar por medio del TCP/IP. Examinamos el protocolo RARP, el cual utiliza el direccionamiento físico de red para obtener la dirección de red de redes de la máquina. El mecanismo RARP proporciona la dirección de hardware físico de la máquina de destino para identificar de manera única el procesador y transmite por difusión la solicitud RARP. Los servidores en la red reciben el mensaje, buscan la transformación en una tabla (de manera presumible en su almacenamiento secundario) y responden al transmisor. Una vez que una máquina obtiene su dirección IP, la guarda en la memoria y no vuelve a utilizar RARP hasta que se inicia de nuevo.

PARA CONOCER MÁS

Los detalles de RARP se encuentran en Finlayson, *et. al.* (RFC 903). Finlayson (RFC 906) describe el proceso de iniciación de una estación de trabajo utilizando el protocolo TFTP. Bradley y Brown (RFC 1293) especifican un protocolo relacionado, *ARP inverso*. El protocolo ARP inverso permite que una computadora busque en la máquina ubicada en el extremo opuesto de una conexión de hardware para determinar su dirección IP. ARP fue diseñado para las computadoras conectadas por medio de Frame Relay. En el volumen II de esta obra se describe un ejemplo de una implementación de RARP.

En el capítulo 21, se consideran alternativas a RARP, conocidas como BOOTP y DHCP, una extensión más reciente. A diferencia del esquema de determinación de direcciones de bajo nivel que proporciona RARP, tanto BOOTP como DHCP están construidos con protocolos de más alto nivel, como IP y UDP. En el capítulo 21, se comparan los dos enfoques y se analizan las ventajas y debilidades de cada uno.

EJERCICIOS

6.1 Un servidor RARP puede transmitir por difusión respuestas RARP a todas las máquinas o transmitir cada respuesta de manera directa a la máquina que lo solicite. Caracterice una tecnología de red en la que sea benéfica la transmisión de respuestas por difusión a todas las máquinas.

6.2 RARP es un protocolo enfocado de manera específica, en el sentido de que sus respuestas sólo contienen una pieza de información (por ejemplo, la dirección IP solicitada). Cuando una computadora se inicia, por lo general necesita conocer su nombre además de su dirección Internet. Amplíe el concepto de RARP para proporcionar la información adicional.

6.3 ¿Qué tanto más grandes serán las tramas Ethernet cuando se añada información a RARP como se describe en el ejercicio anterior?

6.4 Agregando un segundo servidor RARP a una red aumenta su confiabilidad. ¿Tiene sentido agregar un tercero? ¿Por qué sí o por qué no?

6.5 Las estaciones de trabajo sin disco utilizan RARP para obtener sus direcciones IP, pero siempre asumen que la respuesta proviene del servidor de archivos de la estación de trabajo. La máquina sin disco trata de obtener una imagen de iniciación de ese servidor. Si no recibe una respuesta, la estación de trabajo entra en un bucle infinito de transmisión de solicitudes. Explique cómo agregar un servidor RARP de respaldo a una configuración de este tipo puede causar que la red se congestione con difusiones. Pista: piense en las fallas de alimentación de corriente.

6.6 Monitorée una red local mientras reinicia varias computadoras. ¿Cuál utiliza RARP?

6.7 Los servidores RARP de respaldo mencionados en el texto se valen de la llegada de una segunda solicitud dentro de un corto periodo de tiempo para realizar una respuesta. Considere el esquema de servidor RARP, que hace que todos los servidores contesten la primera solicitud, pero evita el congestionamiento al hacer que cada servidor retrase por un periodo aleatorio la respuesta. ¿Bajo qué circunstancias dicho diseño daría mejores resultados que el diseño descrito en el texto?

7

Protocolo Internet: entrega de datagramas sin conexión

7.1 Introducción

En capítulos anteriores, revisamos algunas partes del hardware y del software de red que hacen posible la comunicación entre redes, explicamos la tecnología de red subyacente y la asignación de direcciones. En este capítulo, se explica el principio fundamental de la entrega sin conexión y analiza cómo se proporciona ésta por medio del *Protocolo Internet (IP)*, uno de los dos protocolos más importantes utilizados en el enlace de redes. Estudiaremos el formato de los datagramas IP y veremos como éstos forman la base para toda la comunicación en una red de redes. En los siguientes dos capítulos, continuaremos nuestro examen del Protocolo Internet analizando el ruteo de datagramas y el manejo de errores.

7.2 Una red virtual

En el capítulo 3, se analizó una arquitectura de red de redes en la que los ruteadores conectan múltiples redes físicas. Considerar esto exclusivamente como una arquitectura puede ser engañoso, debido a que el enfoque se daría hacia la interfaz que proporciona la red de redes al usuario, sin considerar la tecnología de interconexión.

Un usuario concibe una red de redes como una sola red virtual que interconecta a todos los anfitriones, y a través de la cual es posible la comunicación; la arquitectura subyacente permanece oculta y es irrelevante.

En cierto sentido, una red de redes es una abstracción de una red física porque, en los niveles inferiores, proporciona la misma funcionalidad: acepta paquetes y los entrega. En niveles superiores el software de la red de redes aporta la mayor parte de las funciones más elaboradas que percibe el usuario.

7.3 Arquitectura y filosofía de Internet

Conceptualmente, como se muestra en la figura 7.1, una red de redes TCP/IP proporciona tres conjuntos de servicios; su distribución en la figura sugiere una dependencia entre ellos. En el nivel inferior, un servicio de entrega sin conexión proporciona el fundamento sobre el cual se apoya el resto. En el siguiente nivel, un servicio de transporte confiable proporciona una plataforma de alto nivel de la cual dependen las aplicaciones. Exploraremos cada uno de estos servicios, entendiendo qué es lo que proporciona cada uno y considerando los protocolos asociados a ellos.

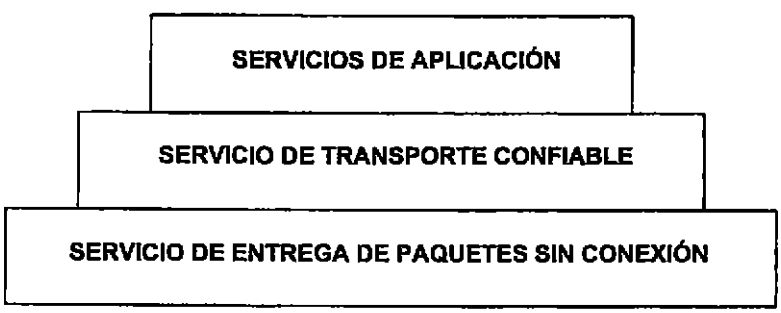


Figura 7.1 Las tres capas conceptuales de los servicios de Internet.

7.4 El concepto de entrega no confiable

Aun cuando podemos asociar un software de protocolo con cada uno de los servicios en la figura 7.1, la razón para identificarlos como partes conceptuales de la red de redes es que éstos constituyen claramente un aspecto esencial respecto a la filosofía del diseño. El punto a considerar es el siguiente:

El software de Internet está diseñado en torno a 3 conceptos de servicios de red arreglados jerárquicamente; muchos de los éxitos alcanzados se deben a esta arquitectura sorprendentemente robusta y adaptable.

Una de las ventajas más significativas de esta separación de conceptos es que es posible reemplazar un servicio sin afectar a los otros. Así, los investigadores y desarrolladores pueden proceder concurrentemente en los tres.

7.5 Sistema de entrega sin conexión

El servicio más importante de la red de redes consiste en un sistema de entrega de paquetes. Técnicamente, el servicio se define como un sistema de entrega de paquetes sin conexión y con el mejor esfuerzo, análogo al servicio proporcionado por el hardware de red que opera con un paradigma de entrega con el mejor esfuerzo. El servicio se conoce como *no confiable* porque la entrega no está garantizada. Los paquetes se pueden perder, duplicar, retrasar o entregar sin orden, pero el servicio no detectará estas condiciones ni informará al emisor o al receptor. El servicio es llamado *sin conexión* dado que cada paquete es tratado de manera independiente de todos los demás. Una secuencia de paquetes que se envían de una computadora a otra puede viajar por diferentes rutas, algunos de ellos pueden perderse mientras otros se entregan. Por último, se dice que el servicio trabaja con base en una *entrega con el mejor esfuerzo* porque el software de red de redes hace un serio intento por entregar los paquetes. Esto es, la red de redes no descarta paquetes caprichosamente; la no confiabilidad aparece sólo cuando los recursos están agotados o la red subyacente falla.

7.6 Propósito del Protocolo Internet

El protocolo que define el mecanismo de entrega sin conexión y no confiable es conocido como *Protocolo Internet* y, por lo general se le identifica por sus iniciales, *IP*.[1] El protocolo IP proporciona tres definiciones importantes. Primero, define la unidad básica para la transferencia de datos utilizada a través de una red de redes TCP/IP. Es decir, especifica el formato exacto de todos los datos que pasarán a través de una red de redes TCP/IP. Segundo, el software IP realiza la función de *ruteo*, seleccionando la ruta por la que los datos serán enviados. Tercero, además de aportar especificaciones formales para el formato de los datos y el ruteo, el IP incluye un conjunto de reglas que le dan forma a la idea de entrega de paquetes no confiable. Las reglas caracterizan la forma en que los anfitriones y ruteadores deben procesar los paquetes, cómo y cuándo se deben generar los mensajes de error y las condiciones bajo las cuales los paquetes pueden ser descartados. El IP es una parte fundamental del diseño de la red de redes TCP/IP, que a veces se conoce como *tecnología basada en el IP*.

Iniciaremos nuestra consideración acerca del IP en este capítulo revisando el formato de los paquetes y sus especificaciones. Dejaremos para capítulos posteriores los temas de ruteo y manejo de errores.

---

[1] De la abreviatura IP se genera la expresión "dirección IP"

7.7 El datagrama de Internet

La analogía entre una red física y una red de redes TCP/IP es muy fuerte. En una red física, la unidad de transferencia es una trama que contiene un encabezado y datos, donde el encabezado contiene información sobre la dirección de la fuente (física) y la del destino. La red de redes llama a esta unidad de transferencia básica *datagrama Internet*, a veces *datagrama IP* o simplemente *datagrama*. Como una trama común de red física, un datagrama se divide en áreas de encabezado y datos. También, como una trama, el encabezado del datagrama contiene la dirección de la fuente y del destino, contiene también un campo de tipo que identifica el contenido del datagrama. La diferencia, por supuesto, es que el encabezado del datagrama contiene direcciones IP en tanto que el encabezado de la trama contiene direcciones físicas. La figura 7.2 muestra la forma general de un datagrama:

ENCABEZADO DEL DATAGRAMA	ÁREA DE DATOS DEL DATAGRAMA

Figura 7.2 Forma general de un datagrama IP, la estructura análoga de TCP/IP con respecto a la trama de una red. El IP especifica el formato de un encabezado incluyendo las direcciones de fuente y destino. El IP no especifica el formato del área de datos; el datagrama se puede utilizar para transportar datos arbitrarios.

7.7.1 Formato de datagrama

Ahora que hemos descrito la disposición general de un datagrama IP, podemos observar su contenido con mayor detalle. La figura 7.3 muestra el arreglo de campos en un datagrama.

Debido a que el proceso de los datagramas se da en el software, el contenido y el formato no está condicionado por ningún tipo de hardware. Por ejemplo, el primer campo de 4 bits en un datagrama (*VERS*) contiene la versión del protocolo IP que se utilizó para crear el datagrama. Esto se utiliza para verificar que el emisor, el receptor y cualquier ruteador entre ellos proceda de acuerdo con el formato del datagrama. Todo software IP debe verificar el campo de versión antes de procesar un datagrama para asegurarse de que el formato corresponde al tipo de formato que espera el software. Si hay un cambio en el estándar, las máquinas rechazarán los datagramas con versiones de protocolo que difieren del estándar, evitando con ello que el contenido de los datagramas sea mal interpretado debido a un formato obsoleto. El protocolo IP actual trabaja con la versión número 4.

El campo de longitud encabezado *(HLEN)*, también de 4 bits, proporciona el encabezado del datagrama con una longitud medida en palabras de 32 bits. Como podremos ver, todos los campos del encabezado tienen longitudes fijas excepto para el campo *OPTIONS* de *IP* y su correspondiente campo *PADDING*. El encabezado más común, que no contiene opciones ni rellenos, mide 20 octetos y tiene un campo de longitud de encabezado igual a 5.

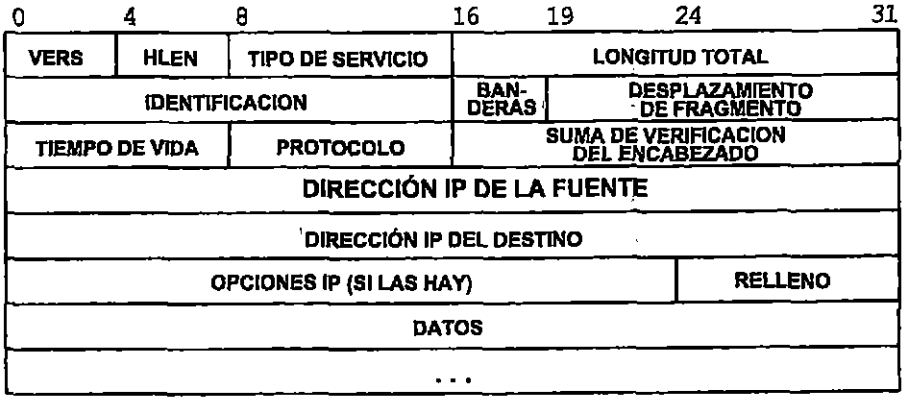


Figura 7.3 Formato de un datagrama Internet, la unidad básica de transferencia en una red de redes TCP/IP.

El campo *TOTAL LENGTH* proporciona la longitud del datagrama IP medido en octetos, incluyendo los octetos del encabezado y los datos. El tamaño del área de datos se puede calcular restando la longitud del encabezado (*HLEN*) de *TOTAL LENGTH*. Dado que el campo *TOTAL LENGTH* tiene una longitud de 16 bits, el tamaño máximo posible de un datagrama IP es de 2^{16} o 65,535 octetos. En la mayor parte de las aplicaciones, ésta no es una limitación severa, pero puede volverse una consideración importante en el futuro, si las redes de alta velocidad llegan a transportar paquetes de datos superiores a los 65,535 octetos.

7.7.2 Tipo de datagramas de servicios y prioridad de datagramas

Conocido informalmente como *Type Of Service (TOS)*, el campo de 8 bits *SERVICE TYPE* especifica cómo debe manejarse el datagrama; el campo está subdividido en 5 subcampos, como se muestra en la figura 7.4:

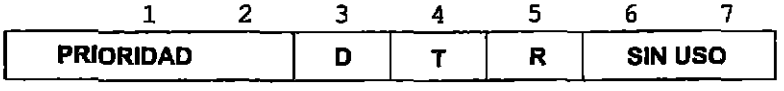


Figura 7.4 Los cinco subcampos que componen el campo *SERVICE-TYPE* de 8 bits.

Tres bits *PRECEDENCE* especifican la prioridad del datagrama, con valores que abarcan de 0 (prioridad normal) a 7 (control, de red), permitiendo con ello indicar al emisor la importancia de cada datagrama. Aun cuando la mayor parte del software de los anfitriones y los ruteadores ignora el tipo de servicio, éste es un concepto importante dado que proporciona un mecanismo que permi-

te controlar la información que tendrá prioridad en los datos. Por ejemplo, si todos los anfitriones y ruteadores responden a la prioridad, es posible implementar algoritmos de control de congestionamiento que no se vean afectados por el mismo congestionamiento que desean controlar.

Los bits *D*, *T* y *R* especifican el tipo de transporte deseado para el datagrama. Cuando está activado, el bit *D* solicita procesamiento con retardos cortos, el bit *T* solicita un alto desempeño y el bit *R* solicita alta confiabilidad. Por supuesto, no es posible para una red de redes garantizar siempre el tipo de transporte solicitado (por ejemplo, éste sería el caso si no se encuentra una ruta adecuada). De esta manera, debemos pensar en una solicitud de transporte como en una simple indicación para los algoritmos de ruteo, no como en un requerimiento obligatorio. Si un ruteador no conoce más que una posible ruta para alcanzar un destino determinado, puede utilizar el campo de tipo de transporte para seleccionar una con las características más cercanas a la petición deseada. Por ejemplo, supongamos que un ruteador puede seleccionar entre una línea arrendada de baja capacidad y una conexión vía satélite con un gran ancho de banda (pero con un retardo alto). Los datagramas que acarrean la información tecleada por un usuario hacia una computadora remota pueden tener el bit *D* activado, solicitando que la entrega sea lo más rápida posible, mientras que el transporte de datagramas en la transferencia de un archivo de datos grande podría tener activado el bit *T*, solicitando que el recorrido se haga a través de una ruta que incluya un satélite de alta capacidad.

También es importante para la realización del proceso que los algoritmos de ruteo seleccionen de entre las tecnologías de red física subyacente, las características de retardo, desempeño y confiabilidad. Con frecuencia, una tecnología dada intercambiará una característica por otra (por ejemplo, un alto desempeño implicará un mayor retardo). Así, la idea es proporcionar un algoritmo de ruteo como si se tratara de una indicación de qué es lo más importante; rara vez es necesario especificar los tres tipos de servicio juntos. En resumen:

> *Hemos visto la especificación del tipo de transporte como una indicación para el algoritmo de ruteo que ayuda en la selección de una ruta entre varias hacia un destino, con base en el conocimiento de las tecnologías de hardware disponibles en esas rutas. Una red de redes no garantiza la realización del tipo de transporte solicitado.*

7.7.3 Encapsulación de datagramas

Antes de que podamos entender los siguientes campos de un datagrama es importante considerar cómo los datagramas se relacionan con las tramas de las redes físicas. Comenzaremos con una pregunta: "¿qué tan grande puede ser un datagrama?"A diferencia de las tramas de las redes físicas que pueden ser reconocidas por el hardware, los datagramas son manejados por el software. Estos pueden tener cualquier longitud seleccionada por el diseño de protocolo. Hemos visto que el formato de los datagramas actuales asignan solamente 16 bits al campo de longitud total, limitando el datagrama a un máximo de 65,535 octetos. Sin embargo, este límite puede modificarse en versiones de protocolos recientes.

Las limitaciones más importantes en el tamaño de un datagrama se dan en la práctica misma. Sabemos que, como los datagramas se mueven de una máquina a otra, éstos deben transportarse siempre a través de una red física subyacente. Para hacer eficiente el transporte en la red de redes,

queremos garantizar que cada datagrama puede viajar en una trama física distinta. Esto es, queremos que nuestra abstracción de un paquete de red física se transforme directamente en un paquete real si es posible.

La idea de transportar un datagrama dentro de una trama de red es conocida como *encapsulación*. Para la red subyacente un datagrama es como cualquier otro mensaje que se envía de una máquina a otra. El hardware no reconoce el formato del datagrama ni entiende las direcciones de destino IP. Así, como se muestra en la figura 7.5, cuando una máquina envía un datagrama IP hacia otra, el datagrama completo viaja en la porción de datos de la trama de red.

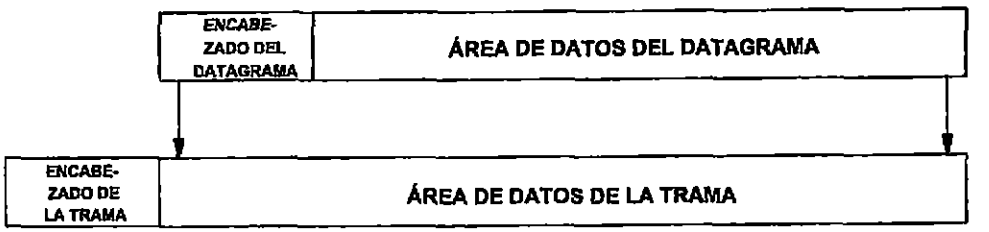


Figura 7.5 Encapsulación de un datagrama IP en una trama. La red física trata al datagrama entero, incluyendo el encabezado, como si se tratara de datos.

7.7.4 Tamaño de datagrama, MTU de red y fragmentación

En un caso ideal, el datagrama IP completo se ajusta dentro de una trama física haciendo que la transmisión a través de la red física sea eficiente.[2] Para alcanzar esta eficiencia, los diseñadores de IP tendrían que seleccionar un tamaño máximo de datagrama, de manera que el datagrama siempre se ajuste dentro de una trama. Pero ¿qué tamaño de trama deberían seleccionar? Después de todo, un datagrama debe viajar a través de muchos tipos de redes físicas conforme se mueve a través de una red de redes hacia su destino final.

Para entender el problema, necesitamos considerar un hecho a propósito del hardware de red: cada tecnología de conmutación de paquetes establece un límite superior fijo para la cantidad de datos que pueden transferirse en una trama física. Por ejemplo, Ethernet limita la transferencia de datos a 1,500[3] octetos, mientras que FDDI permite aproximadamente 4,470 octetos por trama. Nos referiremos a estos límites como la *unidad de transferencia máxima* de una red (*maximum transfer unit*, o *MTU* por sus siglas en inglés). El tamaño de MTU puede ser muy pequeño: algunas tecnologías de hardware limitan la transferencia a 128 octetos o menos. La limitación de los datagramas para que se ajusten a la MTU más pequeña posible en una red de redes hace que la transferencia sea ineficiente cuando estos datagramas pasan a través de una red que puede transportar tramas de tamaño mayor. Sin embargo, permitir que los datagramas sean más grandes que la MTU

[2] *Un campo en el encabezado de trama por lo general* identifica el tipo de datos que se están transportando; Ethernet utiliza el valor tipo 0800_{16} para especificar que el área de datos contiene un datagrama IP encapsulado.

[3] El límite de 1500 proviene de las especificaciones de Ethernet; cuando se utilizaba con un encabezado SNAP, el estándar 802.3 de IEEE limitaba los datos a 1492 octetos. Las plantaciones de algunos vendedores permiten transferencias ligeramente mayores.

mínima de una red, en una red de redes, puede significar que un datagrama no siempre se ajuste dentro de una sola trama de red.

La selección debería ser obvia: el punto a considerar en el diseño de una red de redes es ocultar la tecnología de red subyacente y hacer la comunicación conveniente para el usuario. Así, en lugar de diseñar datagramas que se ajusten a las restricciones de la red física, el software TCP/IP selecciona un tamaño de datagrama más conveniente desde el principio y establece una forma para dividir datagramas en pequeños fragmentos cuando el datagrama necesita viajar a través de una red que tiene una MTU pequeña. Las pequeñas piezas dentro de un datagrama dividido se conocen con el nombre de *fragmentos* y el proceso de división de un datagrama se conoce como *fragmentación*.

Como se ilustra en la figura 7.6, la fragmentación por lo general se da en un ruteador a lo largo del trayecto entre la fuente del datagrama y su destino final. El ruteador recibe un datagrama de una red con una MTU grande y debe enviarlo a una red en la que la MTU es más pequeña que el tamaño del datagrama.

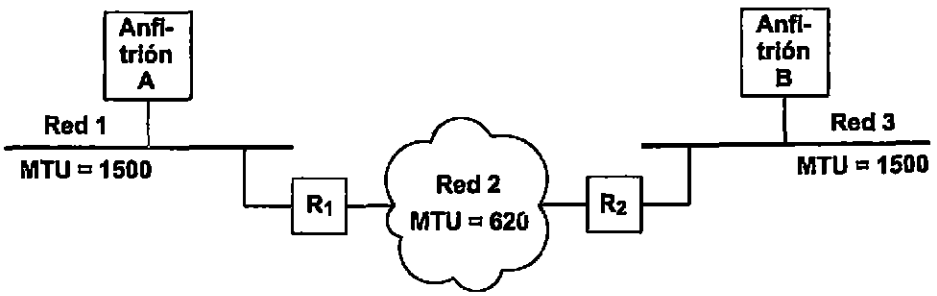


Figura 7.6 Una ilustración de los casos en que se presenta la fragmentación. El ruteador R_1 fragmenta un datagrama extenso para enviarlo desde A hacia B; R_2 fragmenta un datagrama extenso para enviarlo desde B hacia A.

En la figura, ambos anfitriones están conectados directamente a una red Ethernet, la cual tiene una MTU de 1,500 octetos. Así, ambos anfitriones pueden generar y enviar datagramas con un máximo de 1,500 octetos de largo. El trayecto entre ambos, sin embargo, incluye una red con una MTU de 620. Si el anfitrión A envía al anfitrión B un datagrama mayor a 620 octetos, el ruteador R_1 fragmentará el datagrama. De la misma forma, si B envía un datagrama grande hacia A, el ruteador R_2 fragmentará el datagrama.

El tamaño de cada fragmento se selecciona de manera que cada uno de éstos pueda transportarse a través de la red subyacente en una sola trama. Además, dado que el IP representa el desplazamiento de datos en múltiplos de 8 octetos, el tamaño del fragmento debe seleccionarse de modo que sea un múltiplo de 8. Por supuesto, al seleccionar el múltiplo de 8 octetos más cercano a la MTU de la red no es usual dividir el datagrama en fragmentos de tamaños iguales; los últimos fragmentos por lo general son más cortos que los otros. Los fragmentos se deben *reensamblar* para producir una copia completa del datagrama original, antes de que pueda procesarse en su lugar de destino.

El protocolo IP no limita los datagramas a un tamaño pequeño, ni garantiza que los datagramas grandes serán entregados sin fragmentación. La fuente puede seleccionar cualquier tamaño de datagrama que considere apropiado; la fragmentación y el reensamblado se dan automáticamente sin que la fuente deba realizar ninguna acción especial. Las especificaciones IP establecen que los ruteadores pueden aceptar datagramas con una longitud equivalente al valor máximo de la MTU de las redes a las que están conectados. Además, un ruteador siempre maneja datagramas de hasta 576 octetos. (Los anfitriones también son configurados para aceptar y reensamblar, si es necesario, datagramas de por lo menos 576 octetos.)

Fragmentar un datagrama significa dividirlo en varios segmentos. Podría ser sorprendente aprender que cada fragmento tiene el mismo formato que el datagrama original. La figura 7.7 muestra el resultado de la fragmentación.

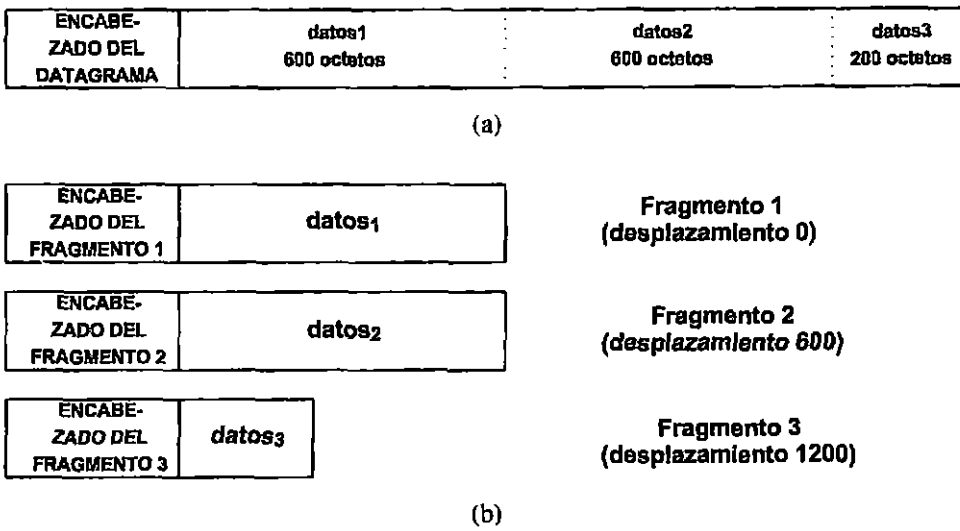


Figura 7.7 (a) Un datagrama original que transporta 1400 octetos de datos y (b) los tres fragmentos para la red con una MTU de 620. Los encabezados 1 y 2 tiene activado el bit *más fragmentos*. El desplazamiento se muestra como un número decimal de octetos; estos valores se deben dividir entre 8 para obtener el valor en el que se localiza el encabezado de los fragmentos.

Cada fragmento contiene un encabezado de datagrama que duplica la mayor parte del encabezado del datagrama original (excepto por un bit en el campo *FLAGS* que muestra que éste es un fragmento), seguido por tantos datos como puedan ser acarreados en el fragmento siempre y cuando la longitud total se mantenga en un valor menor a la MTU de la red en la que debe viajar.

7.7.5 Reensamblado de fragmentos

¿Un datagrama debe ser reensamblado luego de pasar a través de una red o los fragmentos deben transportarse hasta el anfitrión final antes de ser reensamblados? En una red de redes TCP/IP, una vez que un datagrama se ha fragmentado, los fragmentos viajan como datagramas separados hacia su destino final donde serán reensamblados. Preservar los fragmentos en todo el trayecto hasta su destino final tiene dos desventajas. Primero, dado que los datagramas no son reemsamblados inmediatamente después de pasar a través de una red con una MTU pequeña, los fragmentos pequeños deben transportarse en esa forma desde el punto de fragmentación hasta el destino final. Reensamblar los datagramas en el destino final puede implicar que el proceso se realice con cierta ineficiencia: aun cuando se encuentre en una red física con una capacidad de MTU grande después del punto de fragmentación, ésta será atravesada por fragmentos pequeños. Segundo, si se pierde cualquier fragmento, el datagrama no podrá reensamblarse. La máquina de recepción hace que arranque un *temporizador de reensamblado* cuando recibe un fragmento inicial. Si el temporizador termina antes de que todos los fragmentos lleguen, la máquina de recepción descartará los fragmentos sin procesar el datagrama. Así, la probabilidad de perder un datagrama se incrementa con la fragmentación ya que la pérdida de un solo fragmento provoca la pérdida del datagrama completo.

Aún considerando desventajas menores, la realización del reensamblado en el destino final trabaja bien. Esto permite que cada fragmento se pueda rutear de manera independiente sin necesidad de que ruteadores intermedios almacenen o reensamblen fragmentos.

7.7.6 Control de fragmentación

Tres campos en el encabezado del datagrama, *IDENTIFICATION, FLAGS* y *FRAGMENT OFFSET*, controlan la fragmentación y el reensamblado de los datagramas. El campo *IDENTIFICATION* contiene un entero único que identifica al datagrama. Recordemos que cuando un ruteador fragmenta un datagrama, éste copia la mayor parte de los campos del encabezado del datagrama dentro de cada fragmento. El campo *IDENTIFICATION* debe copiarse. Su propósito principal es permitir que el destino tenga información acerca de qué fragmentos pertenecen a qué datagramas. Conforme llega cada fragmento, el destino utiliza el campo *IDENTIFICATION* junto con la dirección de la fuente del datagrama para identificar el datagrama. Las computadoras que envían datagramas IP deben generar un valor único para el campo *IDENTIFICATION* por cada datagrama.[4] Hay una técnica utilizada por el software IP que establece un contador global en memoria, lo incrementa cada vez que se crea un datagrama nuevo y asigna el resultado al campo del datagrama *IDENTIFICATION*.

Recordemos que cada fragmento tiene exactamente el mismo formato que un datagrama completo. Para un fragmento, el campo *FRAGMENT OFFSET* especifica el desplazamiento en el datagrama original de los datos que se están acarreando en el fragmento, medido en unidades de 8 octetos,[5] comenzando con un desplazamiento igual a cero. Para reensamblar el datagrama, el destino debe obtener todos los fragmentos comenzando con el fragmento que tiene asignado un despla-

---

[4] En teoría, en las retransmisiones de un datagrama se puede acarrear el mismo campo *IDENTIFICATION* que en el original; en la práctica, los protocolos de alto nivel por lo general realizan la retransmisión dando como resultado un datagrama nuevo con su propio campo *IDENTIFICATION*.

[5] Para ahorrar espacio en el encabezado, los desplazamientos se especifican en múltiplos de 8 octetos.

zamiento igual a *0* hasta el fragmento con el desplazamiento de mayor valor. Los fragmentos no necesariamente llegarán en orden, además no hay comunicación entre el ruteador que fragmentó el datagrama y el destino que trata de reensamblarlo.

Los 2 bits de orden menor del campo de 3 bits *FLAGS* controlan la fragmentación. Por lo general, el software de aplicación que utiliza TCP/IP no se ocupa de la fragmentación debido a que tanto la fragmentación como el reensamblado son procedimientos automáticos que se dan a bajo nivel en el sistema operativo, invisible para el usuario final. Sin embargo, para probar el software de red de redes o depurar problemas operacionales, podría ser importante probar el tamaño de los datagramas en los que se presenta la fragmentación. El primer bit de control ayuda en esta prueba especificando en qué momento se debe fragmentar un datagrama. Se le conoce como bit de *no fragmentación* porque cuando está puesto a *1* especifica que el datagrama no debe fragmentarse. Una aplicación podría seleccionar no permitir la fragmentación cuando sólo el datagrama completo es útil. Por ejemplo, consideremos la secuencia de iniciación de una computadora, en la que una máquina comienza a ejecutar un pequeño programa en ROM y utiliza la red de redes para solicitar una primera iniciación, y otra máquina envía de regreso una imagen de memoria. Si el software ha sido diseñado así, necesitará la imagen completa, pues de otra forma no le será útil; por ello, el datagrama debe tener activado el bit de *no fragmentación*. Cada vez que un ruteador necesita fragmentar un datagrama que tiene activado el bit de *no fragmentación*, el ruteador descartará el datagrama y devolverá un mensaje de error a la fuente.

El bit de orden inferior en el campo *FLAGS* especifica si el fragmento contiene datos intermedios del datagrama original o de la parte final. Este bit es conocido como *more fragments* (*más fragmentos*). Para entender por qué este bit es necesario, consideremos el software IP en el destino final cuando trata de reensamblar un datagrama. Este recibirá los fragmentos (es posible que en desorden) y necesitará saber cuándo ha recibido todos los fragmentos del datagrama. Cuando un fragmento llega, el campo *TOTAL LENGTH* en el encabezado consulta el tamaño del fragmento y no el tamaño del datagrama original; de esta manera el destino no puede utilizar el campo *TOTAL LENGHT* para determinar si ha reunido todos los fragmentos. El bit *más fragmentos* resuelve este problema con facilidad: cada vez que, en el destino, se recibe un fragmento con el bit *más fragmentos* desactivado, se sabe que este fragmento acarrea datos del extremo final del datagrama original. De los campos *FRAGMENT OFFSET* y *TOTAL LENGHT* se puede calcular la longitud del datagrama original. Examinando *FRAGMENT OFFSET* y *TOTAL LENGHT* en el caso de todos los fragmentos entrantes, un receptor puede establecer en qué momento los fragmentos que ha reunido contienen toda la información necesaria para reensamblar el datagrama original completo.

7.7.7 Tiempo de vida (time to live o TTL)

El campo *TIME TO LIVE* especifica la duración, en segundos, del tiempo que el datagrama tiene permitido permanecer en el sistema de red de redes. La idea es sencilla e importante: cada vez que una máquina introduce un datagrama dentro de la red de redes, se establece un tiempo máximo durante el cual el datagrama puede permanecer ahí. Los ruteadores y los anfitriones que procesan los datagramas deben decrementar el campo *TIME TO LIVE* cada vez que pasa un datagrama y eliminarlo de la red de redes cuando su tiempo ha concluido.

Una estimación exacta de este tiempo es difícil dado que los ruteadores por lo general no conocen el tiempo de tránsito por las redes físicas. Unas pocas reglas simplifican el procedimiento y

hacen fácil el manejo de datagramas sin relojes sincronizados. En primer lugar, cada ruteador, a lo largo de un trayecto, desde una fuente hasta un destino, es configurado para decrementar por *1* el campo *TIME TO LIVE* cuando se procesa el encabezado del datagrama. Sin embargo, para manejar casos de ruteadores sobrecargados que introducen largos retardos, cada ruteador registra el tiempo local cuando llega un datagrama, y decrementa el *TIME TO LIVE* por el número de segundos que el datagrama permanece dentro del ruteador esperando que se le despache.

Cada vez que un campo *TIME TO LIVE* llega a cero, el ruteador descarta el datagrama y envía un mensaje de error a la fuente. La idea de establecer un temporizador para los datagramas es interesante ya que garantiza que los datagramas no viajarán a través de la red de redes indefinidamente, aun cuando si una tabla de ruteo se corrompa y los ruteadores direccionen datagramas en un ciclo.

7.7.8 Otros campos de encabezado de datagrama

El campo *PROTOCOL* es análogo al campo tipo en una trama de red. El valor en el campo *PROTOCOL* especifica qué protocolo de alto nivel se utilizó para crear el mensaje que se está transportando en el área *DATA* de un datagrama. En esencia, el valor de *PROTOCOL* especifica el formato del área *DATA*. La transformación entre un protocolo de alto nivel y el valor entero utilizado en el campo *PROTOCOL* debe administrarlo por una autoridad central para garantizar el acuerdo entre los enteros utilizados en Internet.

El campo *HEADER CHECKSUM* asegura la integridad de los valores del encabezado. La suma de verificación IP se forma considerando al encabezado como una secuencia de enteros de 16-bits (en el orden de los octetos de la red), sumándolos juntos mediante el complemento aritmético a uno, y después tomando el complemento a uno del resultado. Para propósitos de cálculo de la suma de verificación, el campo *HEADER CHECKSUM* se asume como igual a cero.

Es importante notar que la suma de verificación sólo se aplica para valores del encabezado IP y no para los datos. Separar la suma de verificación para el encabezado y los datos tiene ventajas y desventajas. Debido a que el encabezado por lo general ocupa menos octetos que los datos, tener una suma de verificación separada disminuye el tiempo de procesamiento y ruteo, los cuales sólo necesitan calcular la suma de verificación del encabezado. La separación también permite a los protocolos de alto nivel seleccionar su propio esquema de suma de verificación para los datos. La mayor desventaja es que los protocolos de alto nivel se ven forzados a añadir su propia suma de verificación o corren el riesgo de que las alteraciones de datos no sean detectadas.

Los campos *SOURCE IP ADDRESS* y *DESTINATION IP ADDRESS* contienen direcciones IP de 32 bits de los datagramas del emisor y del receptor involucrado. Aun cuando los datagramas sean dirigidos a través de muchos ruteadores inmediatos, los campos de fuente y destino nunca cambian; éstos especifican la dirección IP de la fuente original y del destino final.[6]

El campo marcado con el nombre *DATA* en la figura 7.3 muestra el comienzo del área de datos de un datagrama. Su longitud depende, por supuesto, de qué es lo que se está enviando en el datagrama. El campo *OPTIONS* de *IP* que se analiza a continuación tiene una longitud variable. El campo señalado como *PADDING* depende de las opciones seleccionadas. Este representa un grupo de bits puestos en cero que podrían ser necesarios para asegurar que la extensión del encabezado

---

[6] Se hace una excepción cuando el datagrama incluye una lista de opciones de la fuente.

sea un múltiplo exacto de 32 bits (recordemos que el campo de longitud del encabezado se especifica en unidades formadas por palabras de 32 bits).

7.8 Opciones para los datagramas Internet

El campo *OPTIONS* del *IP* aparece a continuación de la dirección de destino y no se requiere en todos los datagramas; las opciones se incluyen en principio para pruebas de red o depuración. Sin embargo, el procesamiento de las opciones es parte integral del protocolo IP, por lo tanto, todos los estándares de implementaciones se deben incluir.

La longitud del campo *OPTIONS* de *IP* varía dependiendo de qué opción sea seleccionada. Algunas opciones tienen una longitud de un octeto; éstas consisten en un solo octeto de *código de opción*. Otras tienen longitudes variables. Cuando las opciones están presentes en un datagrama, aparecen contiguas, sin separadores especiales entre ellas. Cada opción consiste en un solo octeto de código de opción que debe llevar a continuación un solo octeto y un conjunto de octetos de datos para cada opción. El octeto de código de opción se divide en tres campos como se muestra en la figura 7.8.

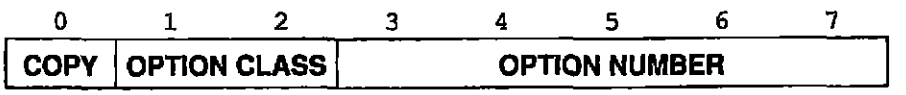


Figura 7.8 División del octeto de código de opción en tres campos de 1, 2 y 5 bits.

El campo consiste en una bandera de 1 bit, llamada *COPY*, un segmento de 2 bits, *OPTION CLASS*, y un segmento de 5 bits, *OPTION NUMBER*. La bandera *COPY* controla la forma en que los ruteadores tratan las opciones durante la fragmentación. Cuando el bit *COPY* está puesto a *1*, especifica que la opción se debe copiar en todos los fragmentos. Cuando está puesto a cero el bit

Option Class	Significado
0	Control de red o datagrama
1	Reservado para uso futuro
2	Depuración y medición
3	Reservado para uso futuro

Figura 7.9 Clases de opciones IP, como se codifican en los bits de *OPTION CLASS*, en un octeto de código de opción.

COPY significa que la opción sólo se debe copiar dentro del primer fragmento y no en todos los fragmentos.

Los bits *OPTION CLASS* y *OPTION NUMBER* especifican la clase general de opción y establecen una opción específica en esta clase. La tabla en la figura 7.9 muestra como se asignan las clases.

La tabla en la figura 7.10 lista las opciones posibles que pueden acompañar a un datagrama IP y muestra los valores para *OPTION CLASS* y *OPTION NUMBER*. Como se muestra en la lista, la mayor parte de las opciones se utiliza con propósito de control.

Option Class	Option Number	Longitud	Descripción
0	0	-	Fin de la lista de opciones. Se utiliza si las opciones no terminan al final del encabezado (ver también campo de relleno de encabezado).
0	1	-	No operación (se utiliza para alinear octetos en una lista de opciones).
0	2	11	Seguridad y restricciones de manejo (para aplicaciones militares).
0	3	var	Ruteo no estricto de fuente. Se utiliza para rutear un datagrama a través de una trayectoria específica.
0	7	var	Registro de ruta. Se utiliza para registrar el trayecto de una ruta.
0	8	4	Identificador de flujo. Se utiliza para transportar un identificador de flujo SATNET (Obsoleto).
0	9	var	Ruteo estricto de fuente. Se utiliza para establecer la ruta de un datagrama en un trayecto específico.
2	4	var	Sello de tiempo Internet. Se usa para registrar sellos de hora a lo largo de una ruta.

Figura 7.10 Las ocho opciones posibles IP con su clase en forma numérica y los códigos de número. El valor *var* en la columna de longitud significa *variable*.

7.8.1 Opción de registro de ruta

Las opciones de ruteo y sello de hora (*timestamp*) son las más interesantes porque proporcionan una manera de monitorear o controlar la forma en que la red de redes maneja las rutas de los datagramas. La opción *registro de ruta* permite a la fuente crear una lista de direcciones IP y arreglarla para que cada ruteador que maneje el datagrama añada su propia dirección IP a la lista. La figura 7.11 muestra el formato de la opción de registro de ruta.

Como se describe arriba, el campo *CODE* contiene la clase de opción y el número de opción (*0* y *7* para el registro de rutas). El campo *LENGTH* especifica la longitud total de la opción como aparece en el datagrama IP, incluyendo los 3 primeros octetos. El campo comienza con un *FIRST IP ADDRESS* que comprende el área reservada para registrar las direcciones de la red de redes. El

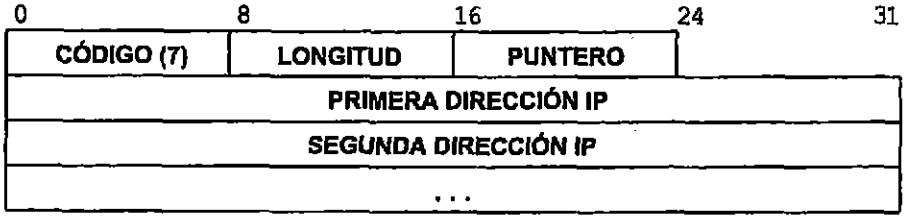


Figura 7.11 Formato de una opción de registro de ruta en un datagrama IP. La opción comienza con tres octetos seguidos inmediatamente por una lista de direcciones. Aun cuando el diagrama muestra direcciones en unidades de 32 bits, éstas no están alineadas con ningún octeto en la frontera de un datagrama.

campo *POINTER* especifica el desplazamiento dentro de la opción de la siguiente ranura disponible.

Cada vez que una máquina maneja un datagrama que tiene activada la opción de registro de ruta, la máquina añade su dirección a la lista del registro de ruta (se debe colocar suficiente espacio en la opción desde la fuente original para manejar todas las entradas que pudieran ser necesarias). Para añadirse a sí misma en la lista, una máquina primero compara el puntero y el campo de longitud. Si el puntero es mayor que la longitud, la lista estará llena y la máquina continuará con el envío del datagrama sin incluirse. Si la lista no está llena, la máquina insertará su dirección IP de 4 octetos en la posición especificada por el *POINTER* e incrementará en 4 el valor de *POINTER*.

Cuando un datagrama llega a su destino, la máquina puede extraer y procesar la lista de direcciones IP. Por lo general, una computadora que recibe un datagrama ignora la ruta registrada. Para usar la opción de registro de ruta se requiere de dos máquinas que estén de acuerdo para cooperar; una computadora no recibirá rutas registradas de los datagramas entrantes ni activará la opción de registro de ruta en los datagramas de salida de manera automática. La fuente debe aceptar la habilitación de la opción de registro de ruta y el destino debe aceptar el procesamiento de la lista resultante.

7.8.2 Opción de ruta fuente

Otra idea que los creadores de redes encontraron interesante es la opción de la *source route* (*ruta de fuente*). La idea de fondo del ruteo de fuente es que proporciona para el emisor una forma en la que éste puede determinar una ruta a través de la red de redes. Por ejemplo, para probar el desempeño de una red física en particular *N*, el administrador de sistemas puede utilizar la ruta de fuente para forzar a los datagramas IP a viajar a través de la red *N*, incluso si los ruteadores normalmente seleccionan una ruta que no está incluida en esa trayectoria. La capacidad para realizar esta prueba es especialmente importante en un ambiente de producción, debido a que permite a los administradores de red tener la libertad de enviar los datagramas a través de la red en una forma que ellos conocen y saben que opera correctamente mientras prueban al mismo tiempo otras redes. Por supues-

to, este tipo de ruteo es útil sólo para personas que entienden la topología de red; el usuario promedio no necesita conocerlo o utilizarlo.

El IP soporta dos formas de ruteo de fuente. Una forma, conocida como *ruteo estricto de fuente*, especifica una vía de ruteo incluyendo una secuencia de direcciones IP en la opción como se muestra en la figura 7.12.

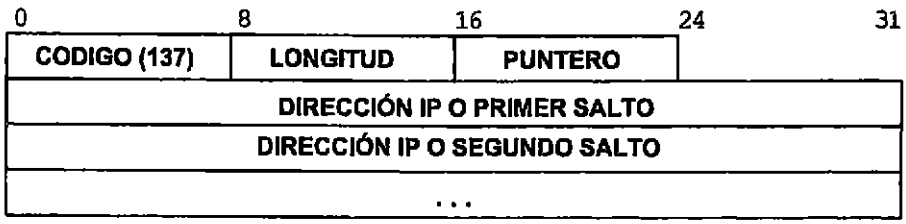


Figura 7.12 La opción de ruta estricta de fuente especifica una ruta precisa estableciendo una lista de direcciones IP que el datagrama debe seguir.

El ruteo estricto de fuente significa que las direcciones especifican la ruta exacta que los datagramas deben seguir para llegar a su destino. La ruta entre dos direcciones sucesivas de la lista debe consistir en una sola red física; se producirá un error si el ruteador no puede seguir una ruta estricta de fuente. La otra forma, conocida como *loose source routing (ruteo no estricto de fuente)*, también incluye una secuencia de direcciones IP. Ésta especifica que el datagrama debe seguir la secuencia de direcciones IP, pero permite múltiples saltos de redes entre direcciones sucesivas de la lista.

Ambas opciones de ruta de fuente requieren que los ruteadores, a lo largo de la trayectoria, anoten su propia dirección de red local en la lista de direcciones. Así, cuando un datagrama llega a su destino, contiene una lista con todas las direcciones recorridas, igual que la lista producida por la opción de registro de ruta.

El formato de una opción de ruta de fuente recuerda al de la opción de registro de ruta, mostrada arriba. Cada ruteador examina los campos *POINTER* y *LENGTH* para ver si la lista está completa. Si es así, el campo puntero es mayor que la longitud y el ruteador establece la ruta del datagrama hacia su destino como lo hace normalmente. Si la lista no está completa, el ruteador sigue al puntero, toma la dirección IP, la reemplaza con la dirección del ruteador[7] y establece la ruta para el datagrama utilizando la dirección que obtuvo de la lista.

7.8.3 Opción de sello de hora

La *opción de sello de hora* trabaja como la opción de registro de ruta. La opción de sello de hora contiene una lista inicial vacía y cada ruteador, a lo largo de la ruta, desde la fuente hasta el destino, escribe sus datos en la lista. Cada entrada a la lista contiene 2 datos de 32 bits: la dirección IP

[7] Un ruteador tiene una dirección por cada interfaz; éste registra la dirección que corresponde a la red en la que se está definiendo una ruta para el datagrama.

0	8	16	24	31
CÓDIGO (68)	LONGITUD	PUNTERO	OFLOW	FLAGS
PRIMERA DIRECCION IP				
PRIMER SELLO DE HORA				
...				

Figura 7.13 Formato de una opción de sello de hora. Los bits en el campo FLAGS (BANDERAS) controlan el formato exacto y las reglas de ruteo que se utilizan para procesar esta opción.

del ruteador que proporciona la entrada y un entero de sello de hora de 32 bits. La figura 7.13 muestra el formato de la opción de sello de hora.

En la figura, los campos *LENGTH* y *POINTER* se utilizan para especificar la longitud del espacio reservado para la opción y la localización de la siguiente ranura no utilizada (exactamente como en la opción de registro de ruta). El campo de 4 bits *OFLOW* contiene un contador entero de ruteadores que podría no proporcionar un sello de hora si la opción fue demasiado pequeña.

El valor en el campo *FLAGS* de 4 bits controla el formato exacto de la opción y establece cómo los ruteadores deben suministrar el sello de hora. Los valores son:

Valor de la bandera	Significado
0	Registro de sello de hora solamente; omite direcciones IP.
1	Anteponer a cada sello de hora una dirección IP (este es el formato que se muestra en la figura 7.13)
3	Las direcciones IP se especifican por el emisor; un ruteador sólo registra un sello de hora si la próxima dirección IP en la lista concuerda con la dirección IP del ruteador.

Figura 7.14 Interpretación de los valores en el campo FLAGS (BANDERAS) de la opción sello de hora

El sello de hora define la hora y la fecha en la que un ruteador manejó el datagrama, expresado en milisegundos desde la media noche Tiempo Universal.[8] Si la representación estándar para la hora no está disponible, el ruteador puede utilizar cualquier representación de tiempo local disponible activando el bit de orden superior en el campo de sello de hora. Por supuesto, el sello de hora para computadoras independientes no siempre será consistente si representan un tiempo universal. Cada máquina reportará una hora de acuerdo a su reloj local y los relojes pueden diferir. Así, el sello de hora deberá considerarse como una estimación, independientemente de la representación.

[8] El Tiempo Universal fue formalmente conocido como Hora del Meridiano de Greenwich; es la hora del día del primer meridiano.

Podría parecer extraño que la opción de sello de hora incluya un mecanismo para hacer que los ruteadores registren sus direcciones IP con sellos de hora dado que la opción de registro de ruta ya proporcionaba esta capacidad. Sin embargo, grabar las direcciones IP con sellos de hora elimina la ambigüedad. Tener un registro de ruta con sellos de hora es también útil pues permite que el receptor sepa con exactitud cuál fue la ruta seguida por el datagrama.

7.8.4 Fragmentación durante el procesamiento de las opciones

La idea en la que se apoya la implementación de bit *COPY* en el campo de opción *CODE* debe estar clara ahora. Cuando se fragmenta un datagrama, un ruteador reproduce algunas opciones IP en todos los fragmentos y, a la vez, coloca algunas otras sólo en parte de esos fragmentos. Por ejemplo, consideremos la opción utilizada para registrar la ruta del datagrama. Habíamos dicho que cada fragmento sería manejado como un datagrama independiente, la cual no garantiza que todos los fragmentos sigan la misma ruta hacia su destino. Si todos los fragmentos contienen la opción de registro de ruta, el destino podría recibir una lista diferente de rutas de cada fragmento. De esta manera, no podría producir una sola lista completa de las rutas al reensamblar los datagramas. Sin embargo, el estándar IP especifica que la opción de registro de ruta sólo se debe copiar dentro de uno de los fragmentos.

No todas las opciones IP se pueden restringir a un fragmento. Consideremos la opción de ruta de fuente, por ejemplo, que especifica de qué manera debe viajar un datagrama a través de la red de redes. La información de ruteo de fuente debe reproducirse en todos los encabezados de los fragmentos pues, de lo contrario, los fragmentos no seguirán la ruta especificada., Así pues, el campo de código para la ruta de fuente especifica que la opción se debe copiar en todos los fragmentos.

7.9 Resumen

El servicio fundamental proporcionado por el software TCP/IP de red de redes es un sistema de entrega de paquetes sin conexión, no confiable, y con el mejor esfuerzo. El Protocolo Internet (IP) especifica formalmente el formato de los paquetes en la red de redes, llamados *datagramas*, e informalmente le da cuerpo a la idea de entrega sin conexión. En este capítulo, nos concentramos en el formato de datagramas; en capítulos posteriores, analizaremos el ruteo IP y el manejo de errores.

De la misma forma que la trama física, el datagrama IP se divide en áreas de encabezado y áreas de datos. Además de información de otro tipo, el encabezado de datagrama contiene las direcciones de fuente y destino, control de fragmentación, prioridad y suma de verificación utilizada para identificar errores de transmisión. Además de los campos de longitud fija, cada encabezado de datagrama puede contener un campo de opciones. El campo de opciones tiene una longitud variable, dependiendo del número y tipo de opciones utilizadas así como del tamaño del área de datos para cada opción. Empleadas para ayudar a monitorear y controlar la red de redes, las opciones permiten especificar o registrar rutas o permiten reunir sellos de hora conforme viajan los datagramas por la red de redes.

PARA CONOCER MÁS

Postel (1980) trata las formas posibles de acercamiento hacia los protocolos, el direccionamiento y el ruteo en las redes de redes. En publicaciones posteriores, Postel (RFC 791) plantea el estándar del Protocolo Internet. Braden (RFC 1122) refina aún más el estándar. Horning (RFC 894) especifica el estándar para la transmisión de datagramas IP a través de una red Ethernet. Clark (RFC 815) describe el reensamblado eficiente de fragmentos. Además del formato para los paquetes, las autoridades de Internet también especifican muchas constantes necesarias en los protocolos de red. Estos valores se pueden encontrar en Reynolds y Postel (RFC 1700). Kent y Mogul (1987) tratan las desventajas de la fragmentación.

Un conjunto alternativo de protocolos de red de redes, conocido como *XNS*, se plantea en Xerox (1981). Boggs *et. al.* (1980) describe el protocolo Paquete Universal PARC (PUP), una abstracción de *XNS* relacionada estrechamente con los datagramas IP.

EJERCICIOS

7.1 ¿Cuál es la mayor ventaja que hay en el hecho de que la suma de verificación de IP cubra sólo el encabezado del datagrama y no los datos? ¿Cuál es la desventaja?

7.2 ¿Siempre es necesario utilizar una suma de verificación IP cuando se envían paquetes en una red Ethernet? ¿Por qué sí o por qué no?

7.3 ¿Cuál es el tamaño MTU para ANSNET? ¿Para Hyperchannel? ¿Para un conmutador ATM?

7.4 ¿Esperaría que una red de área local de alta velocidad tuviera un tamaño de MTU mayor o menor que una red de área amplia?

7.5 Analice qué fragmentos deberían tener un encabezado pequeño no estándar.

7.6 Determine cuándo se dio el último cambio de versión del protocolo IP. ¿Definir un número de versión de protocolo es realmente útil?

7.7 ¿Puede usted imaginar por qué la suma de verificación por complemento a uno fue seleccionada para IP, en lugar de la verificación por redundancia cíclica?

7.8 ¿Cuáles son las ventajas de reensamblar en el destino final, en lugar de hacerlo luego de que el datagrama ha atravesado una red?

7.9 ¿Cuál es la MTU mínima requerida para enviar un datagrama IP que contenga cuando menos un octeto de datos?

7.10 Supongamos que usted está interesado en implantación un procesamiento de datagramas IP en hardware. ¿Hay algún arreglo de campos del encabezado que pudiera hacer al hardware más eficiente? ¿Más fácil de construir?

7.11 Si tiene acceso a una implantación de IP, revísela y pruebe sus implantaciones locales disponibles de IP para comprobar si rechazan datagramas IP con un número de versión obsoleta.

7.12 Cuando un datagrama IP de tamaño mínimo viaja a través de una red Ethernet, ¿qué tan grande es la trama?

8

Protocolo Internet: ruteo de datagramas IP

8.1 Introducción

Hemos visto que todos los servicios de red de redes utilizan un sistema sin conexión de entrega de paquetes y también que la unidad básica de transferencia en una red de redes TCP/IP es el datagrama IP. En este capítulo, se proporciona mayor información sobre el servicio sin conexión, pues se describe cómo los ruteadores direccionan datagramas *IP* y cómo los entregan en su destino final. Pensamos en el formato de datagramas descrito en el capítulo 7 como los aspectos estáticos del Protocolo Internet. La descripción del ruteo en este capítulo presenta los aspectos operacionales. En el siguiente capítulo, concluirá nuestra presentación del IP con una descripción de cómo se manejan los errores: en los capítulos subsecuentes se mostrará cómo otros protocolos utilizan el IP para proporcionar servicios de un nivel más alto.

8.2 Ruteo en una red de redes

En un sistema de conmutación de paquetes, el *ruteo* es el proceso de selección de un camino sobre el que se mandarán paquetes y el *ruteador* es la computadora que hace la selección. El ruteo ocurre a muchos niveles. Por ejemplo, dentro de una red de área amplia que tiene muchas conexiones físicas entre conmutadores de datos, la red por sí misma es responsable de rutear paquetes desde que llegan hasta que salen. Dicho ruteo interno está completamente contenido dentro de la red de área

amplia. Las máquinas en el exterior no pueden participar en las decisiones; sólo ven la red como una entidad que entrega paquetes.

Recuerde que el objetivo del IP es proporcionar una red virtual que comprenda muchas redes físicas, así como ofrecer un servicio sin conexión de entrega de paquetes. Por lo tanto, nos enfocaremos en el *ruteo en red de redes* o *ruteo IP*.[1] De forma análoga al ruteo dentro de una red física, el ruteo IP selecciona un camino por el que se debe enviar un datagrama. El algoritmo de ruteo IP debe escoger cómo enviar un datagrama pasando por muchas redes físicas.

El ruteo en una red de redes puede ser difícil, en especial entre computadoras que tienen muchas conexiones físicas de red. De forma ideal, el software de ruteo examinaría aspectos como la carga de la red, la longitud del datagrama o el tipo de servicio que se especifica en el encabezado del datagrama, para seleccionar el mejor camino. Sin embargo, la mayor parte del software de ruteo en red de redes es mucho menos sofisticado y selecciona rutas basándose en suposiciones sobre los caminos más cortos.

Para entender completamente el ruteo IP, debemos regresar y recordar la arquitectura de una red de redes TCP/IP. Primero, recuerde que una red de redes se compone de muchas redes físicas, interconectadas por computadoras conocidas como *ruteadores*. Cada ruteador tiene conexiones directas hacia dos o más redes. En contraste, por lo general un anfitrión se conecta directamente a una red física. Sin embargo, sabemos que es posible tener un anfitrión multi-homed conectado directamente a muchas redes.

Tanto los anfitriones como los ruteadores participan en el ruteo de datagramas IP que viajan a su destino. Cuando un programa de aplicación en un anfitrión intenta comunicarse, los protocolos TCP/IP eventualmente generan uno o más datagramas IP. El anfitrión debe tomar una decisión de ruteo cuando elige a dónde enviar los datagramas. Como se muestra en la figura 8.1, los anfitriones deben tomar decisiones de ruteo, inclusive si sólo tienen una conexión de red.

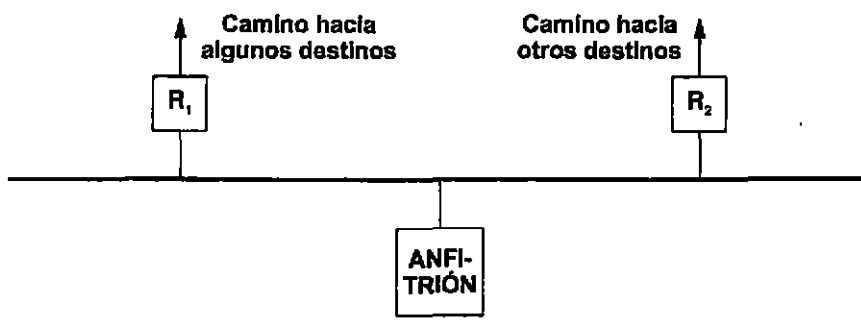


Figura 8.1 Ejemplo de un anfitrión singly-homed que debe rutear datagramas. El anfitrión debe enviar un datagrama al ruteador R_1 o al ruteador R_2, ya que cada uno proporciona el mejor camino hacia algunos destinos.

[1] Los fabricantes también utilizan los términos *direccionamiento IP* y *conmutación IP para describir el ruteo IP*. Asimismo, es interesante que la mayoría todavía se refiere a la información requerida como *información de ruteo IP*.

Por supuesto, los ruteadores también toman decisiones de ruteo IP (ese es su principal propósito y la razón de llamarlos *ruteadores*). ¿Qué hay sobre los anfitriones multi-homed? Cualquier computadora con muchas conexiones de red puede actuar como ruteador y, como veremos, los anfitriones multi-homed que ejecutan el TCP/IP tienen todo el software necesario para el ruteo. Además, los sitios que no pueden adquirir ruteadores por separado a veces utilizan máquinas de tiempo compartido y propósito general como anfitriones y ruteadores (esta práctica por lo general se ve limitada a los sitios en universidades). Sin embargo, los estándares TCP/IP hacen una gran diferenciación entre las funciones de un anfitrión y las de un ruteador, además los sitios que intentan mezclar funciones de anfitrión con funciones de ruteador en una sola máquina, a veces, encuentran que sus anfitriones multi-homed llevan a cabo interacciones inesperadas. Por ahora, distinguiremos los anfitriones de los ruteadores y asumiremos que los primeros no realizan la función, exclusiva de los ruteadores, de transferir paquetes de una red a otra.

8.3 Entrega directa e indirecta

Hablando sin formalismos, podemos dividir el ruteo en dos partes: *entrega directa* y *entrega indirecta*. La entrega directa, que es la transmisión de un datagrama desde una máquina a través de una sola red física hasta otra, es la base de toda la comunicación en una red de redes. Dos máquinas solamente pueden llevar a cabo la entrega directa si ambas se conectan directamente al mismo sistema subyacente de transmisión física (por ejemplo, una sola Ethernet). La *entrega indirecta* ocurre cuando el destino no es una red conectada directamente, lo que obliga al transmisor a pasar el datagrama a un ruteador para su entrega.

8.3.1 Entrega de datagramas sobre una sola red

Sabemos que una máquina en una red física puede enviar una trama física directamente a otra máquina en la misma red. Para transferir un datagrama IP, el transmisor encapsula el datagrama dentro de una trama física, transforma la dirección IP en una dirección física y utiliza la red para entregar el datagrama. En el capítulo 5 se describieron dos mecanismos posibles para la definición de direcciones, incluyendo la utilización del protocolo ARP para la asignación dinámica de direcciones en redes de tipo Ethernet. En el capítulo 7 se analizó la encapsulación de datagramas. Por lo tanto, ya hemos visto todas las piezas necesarias para entender la entrega directa. En resumen:

> *La transmisión de un datagrama IP entre dos máquinas dentro de una sola red física no involucra ruteadores. El transmisor encapsula el datagrama dentro de una trama física, transforma la dirección IP de destino en una dirección física de hardware y envía la trama resultante directamente a su destino.*

¿Cómo sabe el transmisor si el destino reside en una red directamente conectada? La respuesta es la siguiente: sabemos que las direcciones IP se dividen en un prefijo específico de red y un sufijo específico de anfitrión. Para averiguar si un destino reside en una de las redes directamente conectadas, el transmisor extrae la porción de red de la dirección IP de destino y la compara con

la porción de red de su propia dirección IP. Si corresponden, significa que el datagrama se puede enviar de manera directa. Aquí vemos una de las ventajas del esquema de direccionamiento de Internet, a saber:

> *Debido a que las direcciones de red de redes de todas las máquinas dentro de una sola red incluyen un prefijo en común y como la extracción de dicho prefijo se puede realizar mediante unas cuantas instrucciones de máquina, la comprobación de que una máquina se puede alcanzar directamente es muy eficiente.*

Desde la perspectiva de una red de redes, la forma más fácil de pensar en la entrega directa es como el paso final de cualquier transmisión de datagramas, aun si el datagrama atraviesa muchas redes y ruteadores intermedios. El último ruteador del camino entre la fuente del datagrama y su destino siempre se conectará directamente a la misma red física que la máquina de destino. Por lo tanto, el último ruteador entregará el datagrama utilizando la entrega directa. Podemos pensar en la entrega directa entre la fuente y el destino como un caso especial de ruteo de propósito general — en una ruta directa, el datagrama nunca pasa a través de ningún ruteador intermedio.

8.3.2 Entrega Indirecta

La entrega indirecta es más difícil que la directa porque el transmisor debe identificar un ruteador para enviar el datagrama. Luego, el ruteador debe encaminar el datagrama hacia la red de destino.

Para visualizar cómo trabaja el ruteo indirecto, imagínese una gran red con muchas redes interconectadas por medio de ruteadores, pero sólo con dos anfitriones en sus extremos más distantes. Cuando un anfitrión quiere enviar un datagrama a otro, lo encapsula y lo envía hacia el ruteador más cercano. Sabemos que se puede alcanzar un ruteador debido a que todas las redes físicas están interconectadas, así que debe existir un ruteador conectado a cada una. Por lo tanto, el anfitrión de origen puede alcanzar un ruteador utilizando una sola red física. Una vez que la trama llega al ruteador, el software extrae el datagrama encapsulado, y el software IP selecciona el siguiente ruteador a lo largo del camino hacia el destino. De nuevo, se coloca el datagrama en una trama y se envía a través de la siguiente red física hacia un segundo ruteador, y así sucesivamente, hasta que se pueda entregar de forma directa. Estas ideas pueden resumirse así:

> *Los ruteadores en una red de redes TCP/IP forman una estructura cooperativa e interconectada. Los datagramas pasan de un ruteador a otro hasta que llegan a uno que los pueda entregar en forma directa.*

¿Cómo sabe un ruteador a dónde enviar cada datagrama? ¿Cómo puede saber un anfitrión qué ruteador utilizar para llegar a un destino determinado? Las dos preguntas están relacionadas, ya que comprenden el ruteo IP. Las contestaremos en dos fases, considerando en este capítulo el algoritmo básico de ruteo controlado por tablas y posponiendo el análisis sobre cómo los ruteadores aprenden sus rutas.

8.4 Ruteo IP controlado por tabla

El algoritmo usual de ruteo IP emplea una *tabla de ruteo Internet* (a veces, conocida como *tabla de ruteo IP*) en cada máquina que almacena información sobre posibles destinos y sobre cómo alcanzarlos. Debido a que tanto los ruteadores como los anfitriones rutean datagramas, ambos tienen tablas de ruteo IP. Siempre que el software de ruteo IP en un anfitrión necesita transmitir un datagrama, consulta la tabla de ruteo para decidir a dónde enviarlo.

¿Qué información se debe guardar en las tablas de ruteo? Si cada tabla de ruteo contuviera información sobre cada posible dirección de destino, sería imposible mantener actualizadas las tablas. Además, como el número de destinos posibles es muy grande, las máquinas no tendrían suficiente espacio para almacenar la información.

De manera conceptual, nos gustaría utilizar el principio de ocultación de información y permitir a las máquinas tomar decisiones de ruteo con una información mínima. Por ejemplo, nos gustaría aislar la información sobre anfitriones específicos del ambiente local en el que existen y hacer que las máquinas que están lejos ruteen paquetes hacia ellos sin saber dichos detalles. Por fortuna, el esquema de direccionamiento IP nos ayuda a lograr este objetivo. Recuerde que las direcciones IP se asignan de tal manera que todas las máquinas conectadas a una red física compartan un prefijo en común (la porción de red de la dirección). Ya hemos visto que una asignación de este tipo hace que la comprobación para la entrega directa sea eficiente. También significa que las tablas de ruteo sólo necesitan contener prefijos de red y no direcciones IP completas.

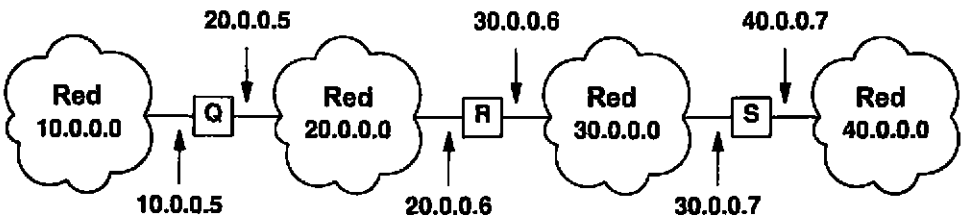
8.5 Ruteo con salto al siguiente

Utilizar la porción de red de una dirección de destino en vez de toda la dirección de anfitrión hace que el ruteo sea eficiente y mantiene reducidas las tablas de ruteo. También es importante, porque ayuda a ocultar información al mantener los detalles de los anfitriones específicos confinados al ambiente local en el que operan. Por lo común, una tabla de ruteo contiene pares (N, R), donde N es la dirección IP de una *red* de destino y R la dirección IP del "siguiente" ruteador en el camino hacia la red N. El ruteador R es conocido como el *salto siguiente* y la idea de utilizar una tabla de ruteo para almacenar un salto siguiente para cada destino es conocida como *ruteo con salto al siguiente*. Por lo tanto, la tabla de ruteo en el ruteador R sólo especifica un paso a lo largo del camino de R a su red de destino —el ruteador no conoce el camino completo hacia el destino.

Es importante entender que cada registro en una tabla de ruteo apunta hacia un ruteador que se puede alcanzar a través de una sola red. Esto es, que todos los ruteadores listados en la tabla de ruteo de la máquina M deben residir en las redes con las que M se conecta de manera directa. Cuando un datagrama está listo para dejar M, el software IP localiza la dirección IP de destino y extrae la porción de red. Luego, M utiliza la porción de red para tomar una decisión de ruteo, seleccionando un ruteador que se pueda alcanzar directamente.

En la práctica, también aplicamos el principio de ocultación de información a los anfitriones. Insistimos que, aunque los anfitriones tengan tablas de ruteo IP, deben guardar información mínima en ellas. La idea es obligar a los anfitriones a que deleguen la mayor parte de sus funciones de ruteo a los ruteadores.

En la figura 8.2 se muestra un ejemplo concreto que nos ayuda a explicar las tablas de ruteo. La red de redes ejemplificada consiste en cuatro redes conectadas por tres ruteadores. En la figura, la tabla de ruteo proporciona las rutas que utiliza el ruteador *R*. Ya que *R* se conecta de manera directa a las redes 20.0.0.0 y 30.0.0.0, puede utilizar la entrega directa para llevar a cabo un envío a un anfitrión en cualquiera de esas redes (posiblemente utilizando ARP para encontrar las direcciones físicas). Teniendo un datagrama destinado para un anfitrión en la red 40.0.0.0, *R* lo rutea a la dirección 30.0.0.7, que es la dirección del ruteador *S*. Luego, *S* entregará el datagrama en forma directa. *R* puede alcanzar la dirección 30.0.0.7 debido a que tanto *R* como *S* se conectan directamente con la red 30.0.0.0.



(a)

PARA ALCANZAR LOS ANFITRIONES EN LA RED	RUTEAR A ESTA DIRECCIÓN
20.0.0.0	ENTREGAR DIRECTAMENTE
30.0.0.0	ENTREGAR DIRECTAMENTE
10.0.0.0	20.0.0.5
40.0.0.0	30.0.0.7

(b)

Figura 8.2 (a) Ejemplo de una red con 4 redes y 3 ruteadores, y (b) la tabla de ruteo en *R*.

Como se demuestra en la figura 8.2, el tamaño de la tabla de ruteo depende del número de redes en la red; solamente crece cuando se agregan nuevas redes. Sin embargo, el tamaño y contenido de la tabla son independientes del número de anfitriones individuales conectados a las redes. Podemos resumir el principio subyacente:

Para ocultar información, mantener reducidas las tablas de ruteo y tomar las decisiones de ruteo de manera eficiente, el software de ruteo IP sólo puede guar-

dar información sobre las direcciones de las redes de destino, no sobre las direcciones de anfitriones individuales.

Escoger rutas basándose tan sólo en la identificación de la red de destino tiene muchas consecuencias. Primero, en la mayor parte de las implantaciones, significa que todo el tráfico destinado a una cierta red toma el mismo camino. Como resultado, aun cuando existen muchos caminos, quizá no se utilicen constantemente. De igual manera, todos los tipos de tráfico siguen el mismo camino sin importar el retraso o la generación de salida de las redes físicas. Segundo, debido a que sólo el último ruteador del camino intenta comunicarse con el anfitrión final, solamente el ruteador puede determinar si el anfitrión existe o está en operación. Por lo tanto, necesitamos encontrar una forma para que el ruteador envíe reportes sobre problemas de entrega, de vuelta a la fuente original. Tercero, debido a que cada ruteador rutea el tráfico de forma independiente, los datagramas que viajan del anfitrión *A* al *B* pueden seguir un camino totalmente distinto al que siguen los datagramas que viajan del anfitrión *B* al *A*. Necesitamos asegurarnos de que los ruteadores cooperen para garantizar que siempre sea posible la comunicación bidireccional.

8.6 Rutas asignadas por omisión

Otra técnica utilizada para ocultar información y mantener reducido el tamaño de las tablas de ruteo, es asociar muchos registros a un ruteador asignado por omisión. La idea es hacer que el software de ruteo IP busque primero la tabla de ruteo para encontrar la red de destino. Si no aparece una ruta en la tabla, las rutinas de ruteo envían el datagrama a un *ruteador asignado por omisión*.

El ruteo asignado por omisión es de gran ayuda cuando un sitio tiene pocas direcciones locales y sólo una conexión con el resto de la red de redes. Por ejemplo, las rutas asignadas por omisión trabajan bien en máquinas anfitriones que se conectan a una sola red física y alcanzan sólo un ruteador, que es la puerta hacia el resto de la red de redes. Toda la decisión de ruteo consiste en dos comprobaciones: una de la red local, y un valor asignado por omisión que apunta hacia el único ruteador posible. Inclusive si el sitio sólo contiene unas cuantas redes locales, el ruteo es sencillo ya que consiste en pocas comprobaciones de las redes locales, más un valor asignado por omisión para todos los demás destinos.

8.7 Rutas por anfitrión específico

Aunque hemos dicho que todo el ruteo está basado en redes y no en anfitriones individuales, la mayor parte del software de ruteo IP permite que se especifiquen rutas por anfitrión como caso especial. Tener rutas por anfitrión le da al administrador de red local un mayor control sobre el uso de la red, le permite hacer comprobaciones y también se puede utilizar para controlar el acceso por razones de seguridad. Cuando se depuran conexiones de red o tablas de ruteo, la capacidad para especificar una ruta especial hacia una máquina individual resulta ser especialmente útil.

8.8 El algoritmo de ruteo IP

Tomando en cuenta todo lo que hemos dicho, el algoritmo de ruteo IP es como sigue:

Algoritmo:

RutaDatagrama (Datagrama, Tabla de Ruteo)

Extraer la dirección IP de destino, D, del datagrama y computar el prefijo de red, N;
si N corresponde a cualquier dirección de red directamente conectada entregar el datagrama al destino D sobre dicha red. (Esto comprende la transformación de D en una dirección física, encapsulando el datagrama y enviando la trama.)
De otra forma, si la tabla contiene una ruta con anfitrión específico para D, enviar el datagrama al salto siguiente especificado en la tabla;
de otra forma, si la tabla contiene una ruta para la red N, enviar el datagrama al salto siguiente especificado en la tabla;
de otra forma, si la tabla contiene una ruta asignada por omisión, enviar el datagrama al ruteador asignado por omisión especificado en la tabla;
de otra forma, declarar un error de ruteo;

Figura 8.3 Algoritmo que utiliza IP para direccionar un datagrama. Por medio de un datagrama IP y una tabla de ruteo, este algoritmo selecciona el salto siguiente al que se debe enviar el datagrama. Todas las rutas deben especificar un salto siguiente que resida en una red conectada directamente.

8.9 Ruteo con direcciones IP

Es importante entender que, a excepción de la disminución del tiempo de vida y de volver a computar la suma de verificación, el ruteo IP no altera el datagrama original. En particular, las direcciones de origen y destino del datagrama permanecen sin alteración; éstas siempre especifican la dirección IP de la fuente original y la dirección IP del último destino.[2] Cuando el IP ejecuta el algoritmo de ruteo, selecciona una nueva dirección IP, que es la dirección IP de la máquina a la que a continuación se tendrá que enviar el datagrama. La nueva dirección es parecida a la dirección de

[2] La única excepción ocurre cuando el datagrama contiene una opción de ruta de origen.

un ruteador. Sin embargo, si el datagrama se puede entregar directamente, la nueva dirección será la misma que la del último destino.

Dijimos que la dirección IP seleccionada por el algoritmo de ruteo IP se conoce como la dirección de *salto al siguiente*, pues indica a dónde se tiene que enviar después el datagrama (aunque quizá no sea el último destino). ¿Dónde almacena el IP la dirección del salto siguiente? No en el datagrama; no existe un lugar reservado para ella. De hecho, el IP no "almacena" la dirección del salto siguiente. Después de ejecutar el algoritmo de ruteo, el IP pasa el datagrama y la dirección del salto siguiente al software de interfaz de red, responsable de la red física sobre la que el datagrama se debe enviar. El software de interfaz de red transforma la dirección de salto siguiente en una dirección física, crea una trama utilizando la dirección física, pone el datagrama en la porción de datos de la trama y envía el resultado. Después de utilizar la dirección de salto siguiente para encontrar una dirección física, el software de interfaz de red la descarta.

Puede parecer extraño que las tablas de ruteo almacenen la dirección IP del salto siguiente para cada red de destino cuando dichas direcciones se tienen que traducir a sus direcciones físicas correspondientes, antes de que se pueda enviar el datagrama. Si nos imaginamos un anfitrión que envía una secuencia de datagramas a la misma dirección de destino, la utilización de direcciones IP nos parecería increíblemente ineficiente. El IP obedientemente extrae la dirección de destino en cada datagrama y utiliza la tabla de ruteo para producir una nueva dirección de salto siguiente. Luego pasa el datagrama y la dirección de salto siguiente a la interfaz de red, que recomputa la asignación para obtener una dirección física. Si la tabla de ruteo utilizó direcciones físicas, la transformación entre la dirección IP de salto siguiente y la dirección física se pueden llevar a cabo sólo una vez, evitando así cómputos innecesarios.

¿Por qué el software IP evita la utilización de direcciones físicas cuando almacena y computa las rutas? Como se muestra en la figura 8.4, existen dos razones importantes.

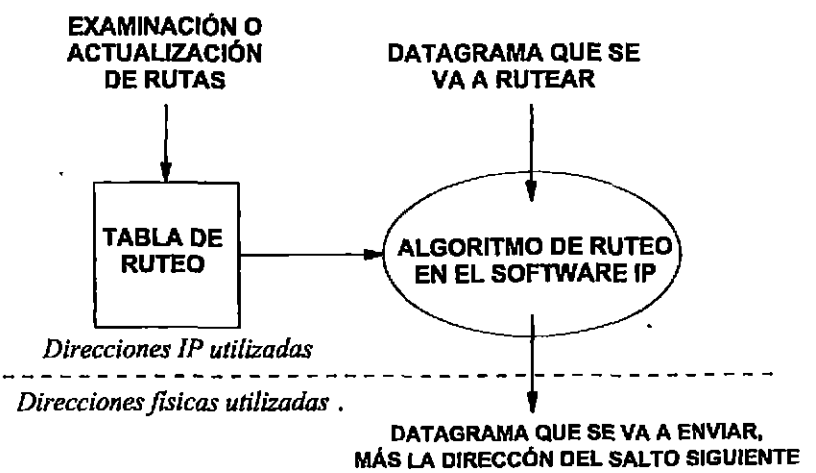


Figura 8.4 El software IP y la tabla de ruteo que utiliza, residen arriba de la frontera de dirección. Utilizar sólo direcciones IP facilita la examinación o cambios de las rutas y oculta los detalles de las direcciones físicas.

Primero, la tabla de ruteo proporciona una interfaz muy transparente entre el software IP que rutea datagramas y el software de alto nivel que manipula las rutas. Para depurar problemas de ruteo, los administradores de red a menudo necesitan examinar las tablas de ruteo. La utilización de direcciones IP solamente en la tabla de ruteo facilita que los administradores las entiendan, lo mismo que ver dónde el software actualizó correctamente las rutas. Segundo, todo el sentido del Protocolo Internet es construir una abstracción que oculte los detalles de las redes subyacentes.

En la figura 8.4 se muestra la *frontera de direcciones*, importante división conceptual entre el software de bajo nivel que entiende las direcciones físicas y el software interno que sólo utiliza direcciones de alto nivel. Arriba de esta frontera, se puede escribir todo el software para que se comunique utilizando direcciones de red de redes; el conocimiento de las direcciones físicas se relega a unas cuantas rutinas de bajo nivel. Veremos que, al respetar la frontera, también se facilita la comprensión, prueba y modificación de la implantación de los restantes protocolos TCP/IP.

8.10 Manejo de los datagramas entrantes

Hasta ahora, hemos analizado el ruteo IP al describir cómo se toman las decisiones sobre los paquetes salientes. Sin embargo, debe quedar claro que el software IP también tiene que procesar los datagramas entrantes.

Cuando un datagrama IP llega a un anfitrión, el software de interfaz de red lo entrega al software IP para su procesamiento. Si la dirección de destino del datagrama corresponde a la dirección IP del anfitrión, el software IP del anfitrión acepta el datagrama y lo pasa al software de protocolo de alto nivel apropiado, para su procesamiento posterior. Si la dirección IP de destino no corresponde, se requiere que el anfitrión descarte el datagrama (por ejemplo, está prohibido que los anfitriones intenten direccionar datagramas que accidentalmente se rutearon a la máquina equivocada).

A diferencia de los anfitriones, los ruteadores sí realizan el direccionamiento. Cuando llega un datagrama IP a un ruteador, éste lo entrega al software IP. De nuevo, surgen dos casos: que el datagrama haya podido llegar a su destino final o que, quizá, necesite viajar más. Como con los anfitriones, si la dirección de destino del datagrama corresponde a la dirección IP, el software IP pasa el datagrama a un software de protocolo de nivel más alto para su procesamiento.[3] Si el datagrama no ha llegado a su destino final, el IP lo rutea utilizando el algoritmo estándar así como la información en la tabla local de ruteo.

La determinación sobre si un datagrama IP alcanzó su destino final no es tan trivial como parece. Recuerde que hasta un anfitrión puede tener muchas conexiones físicas, cada una con su propia dirección IP. Cuando llega un datagrama IP, la máquina debe comparar la dirección de destino de red de redes con la dirección IP de cada una de sus conexiones de red. Si alguna corresponde, guarda el datagrama y lo procesa. Una máquina también debe aceptar datagramas que se transmitieron por difusión en la red física, si su dirección IP de destino es la dirección IP de difusión limitada, o es la dirección IP de difusión dirigida para esa red. Como veremos en los capítulos 10 y 17, las direcciones de subred y de multidifusión hacen que el reconocimiento de direcciones sea aún más complejo. En cualquier caso, si la dirección no corresponde a ninguna de las direcciones de la máquina local, el IP disminuye el campo de tiempo de vida en el encabezado del datagrama, des-

[3] Por lo general, los únicos datagramas destinados para un ruteador, son los utilizados para probar la conectividad o los que llevan comandos de manejo del ruteador.

cartándolo si el contador llega a cero, o computa una nueva suma de verificación y rutea el datagrama si la cuenta es positiva.

¿Todas las máquinas deben direccionar los datagramas IP que reciben? Obviamente, un ruteador debe direccionar datagramas entrantes ya que esa es su función principal. También hemos dicho que algunos anfitriones multi-homed actúan como ruteadores, aunque realmente son sistemas de computación multi-propósito. Aunque utilizar un anfitrión como ruteador por lo general no es una buena idea, si se elige utilizarlos de esa manera, el anfitrión debe configurarse para rutear datagramas igual que lo hace un ruteador. ¿Pero qué hay de los otros anfitriones, los que no están diseñados para ser ruteadores? La respuesta es que los anfitriones que no están diseñados para ello *no* deben rutear los datagramas que reciban, sino descartarlos.

Existen cuatro razones por las que un anfitrión que no esté diseñado para trabajar como ruteador debe abstenerse de realizar cualquier función de ruteo. Primero, cuando un anfitrión, de los antes mencionados, recibe un datagrama diseñado para alguna otra máquina, es que algo salió mal con el direccionamiento, ruteo o entrega en la red de redes. El problema puede no verse si el anfitrión toma una acción correctiva al rutear el datagrama. Segundo, el ruteo causará tráfico innecesario de red (y puede quitarle tiempo a la CPU para utilizar de forma legítima el anfitrión). Tercero, los errores simples pueden causar un caos. Suponga que cada anfitrión rutea tráfico e imagine lo que pasa si una máquina accidentalmente transmite por difusión un datagrama que está destinado al anfitrión *H*. Debido a que se llevó a cabo una difusión, cada anfitrión dentro de la red recibe una copia del datagrama. Cada anfitrión direcciona su copia hacia *H*, que se verá bombardeado con muchas copias. Cuarto, como se muestra en los siguientes capítulos, los ruteadores hacen mucho más que sólo rutear el tráfico. Como se mostrará en el siguiente capítulo, los ruteadores utilizan un protocolo especial para reportar errores y los anfitriones no (de nuevo, para evitar que muchos reportes de error saturen una fuente). Los ruteadores también propagan información de ruteo para asegurarse de que sus tablas están actualizadas. Si los anfitriones rutean datagramas sin participar por completo en todas las funciones de ruteo, se pueden presentar anormalidades inesperadas.

8.11 Establecimiento de tablas de ruteo

Hemos analizado cómo el IP rutea datagramas basándose en el contenido de las tablas de ruteo, sin indicar de qué manera inician o actualizan los sistemas sus tablas conforme cambia la red. En los capítulos posteriores, se tratarán estos temas y se analizarán los protocolos que permiten que los ruteadores mantengan sus tablas actualizadas. Por ahora, sólo es importante entender que el software IP utiliza la tabla de ruteo siempre que decide direccionar un datagrama, así que cambiar las tablas de ruteo cambiaría los caminos que siguen los datagramas.

8.12 Resumen

El ruteo IP consiste en decidir a dónde enviar un datagrama basándose en su dirección IP de destino. La entrega directa es posible si la máquina de destino reside en una red a la que se conecta la máquina transmisora; pensamos que ese es el paso final en la transmisión de datagramas. Si el

transmisor no puede alcanzar directamente al destino, debe direccionar el datagrama hacia un ruteador. El paradigma general es que todos los anfitriones envían datagramas de manera indirecta al ruteador más cercano; los datagramas viajan a través de la red de redes de un ruteador a otro hasta que pueden ser entregados de manera directa sobre una red física.

Cuando el software IP busca una ruta, el algoritmo genera la dirección IP de la siguiente máquina (por ejemplo, la dirección del salto siguiente) a la que se debe enviar el datagrama; el IP pasa el datagrama y la dirección del salto siguiente al software de interfaz de red. La transmisión de un datagrama de una máquina a la siguiente siempre comprende la encapsulación del datagrama en una trama física, transformando la dirección del salto siguiente en una dirección física y enviando la trama al utilizar el hardware subyacente.

El algoritmo de ruteo en una red de redes utiliza sólo direcciones IP y se controla por medio de tablas. Aunque es posible que una tabla de ruteo contenga una dirección de destino de un anfitrión específico, la mayor parte de ellas solamente contienen direcciones de red para mantenerse de un tamaño reducido. La utilización de una ruta asignada por omisión también puede ser útil para mantener reducida una tabla de ruteo, especialmente para los anfitriones que pueden accesar sólo un ruteador.

PARA CONOCER MÁS

El ruteo es un tema importante. Frank y Chou (1971) y Schwartz y Stern (1980) tratan el ruteo en forma general; Postel (1980) analiza el ruteo en una red de redes. Braden y Postel (RFC 1009) proporcionan un resumen de cómo los ruteadores de Internet manejan los datagramas IP. Almquist (RFC 1716) ofrece un resumen sobre estudios más recientes. Narten (1989) contiene una encuesta sobre el ruteo en Internet. Fultz y Kleinrock (1971) analizan esquemas adaptables de ruteo; y McQuillan, Richer, y Rosen (1980) describen el algoritmo adaptable de ruteo ARPANET.

La idea de utilizar afirmaciones sobre políticas para formular reglas sobre el ruteo se considera a menudo. Leiner (RFC 1124) considera las políticas para redes interconectadas. Braun (RFC 1104) analiza modelos de políticas de ruteo para redes de redes; Rekhter (RFC 1092) relaciona las políticas de ruteo con la segunda columna vertebral NSFNET, y Clark (RFC 1102) describe la utilización de políticas de ruteo con el IP.

EJERCICIOS

8.1 Complete las tablas de ruteo para todos los ruteadores en la figura 8.2. ¿Qué ruteadores se beneficiarían más utilizando una ruta asignada por omisión?

8.2 Examine el algoritmo de ruteo utilizado en su sistema local. ¿Están cubiertos todos los casos mencionados aquí? ¿Permite el algoritmo cualquier acción no mencionada?

8.3 ¿Qué es lo que hace un ruteador con el valor de *tiempo de vida* en un encabezado IP?

8.4 Considere una máquina con dos conexiones a redes físicas y con dos direcciones IP, I_1 e I_2. ¿Es posible que esa máquina reciba un datagrama destinado para I_2 sobre la red con la dirección I_1? Explíquelo.

8.5 Considere que dos anfitriones, *A* y *B*, se conectan a una misma red física, *N*. ¿Es posible que, al utilizar nuestro algoritmo de ruteo, *A* reciba un datagrama destinado para *B*? Explíquelo.

8.6 Modifique el algoritmo de ruteo para incorporar las opciones de ruteo de fuente IP que se trataron en el capítulo 7.

8.7 Un ruteador IP debe realizar un cómputo que toma tiempo, proporcional a la longitud del encabezado del datagrama, cada vez que procesa un datagrama. Explíquelo.

8.8 Un administrador de red arguye que, para facilitar el monitoreo y la depuración de su red local, quiere reescribir el algoritmo de ruteo para que compruebe las rutas de anfitrión específico *antes* de comprobar la entrega directa. ¿Se puede imaginar cómo podría utilizar el algoritmo revisado para diseñar un monitor de red?

8.9 ¿Es posible direccionar un datagrama a una dirección IP de un ruteador? ¿Tiene algún sentido hacerlo?

8.10 Considere un algoritmo modificado de ruteo que examine las rutas de anfitrión específico antes de comprobar la entrega en redes conectadas directamente. ¿Bajo qué circunstancias se desearía un algoritmo así? ¿Bajo qué circunstancias no?

8.11 Juegue al detective: una tarde, después de monitorear el tráfico IP en una red de área local por 10 minutos, alguien se da cuenta de que todas las tramas destinadas para la máquina *A* llevan datagramas IP que tienen un destino igual a la dirección IP de *A*, mientras que todas las tramas destinadas para la máquina *B* llevan datagramas IP que tienen un destino igual a la dirección IP de *B*. Los usuarios informan que tanto *A* como *B* se pueden comunicar. Explíquelo.

8.12 ¿Cómo podría cambiar el formato de datagrama IP de manera que pudiera aceptar la conmutación de datos en alta velocidad en los ruteadores? Pista: un ruteador debe recomputar la suma de verificación del encabezado después de disminuir el campo de tiempo de vida.

8.13 Compare el CLNP, protocolo ISO de entrega sin conexión (estándar ISO 8473) con el IP. ¿Qué tan bien aceptaría el protocolo ISO la conmutación a alta velocidad? Pista: los campos de longitud variable son caros.

9

Protocolo Internet: mensajes de error y de control (ICMP)

9.1 Introducción

En el capítulo anterior se mostró cómo el software del Protocolo Internet proporciona un servicio de entrega de *datagramas, no confiable y sin conexión*, al hacer que cada ruteador direccione datagramas. Un datagrama viaja de ruteador en ruteador hasta que llega a uno que lo puede entregar directamente a su destino final. Si un ruteador no puede rutear o entregar un datagrama, o si el ruteador detecta una condición anormal que afecta su capacidad para direccionarlo (por ejemplo, congestionamiento de red), necesita informar a la fuente original para que evite o corrija el problema. En este capítulo se analiza un mecanismo que utilizan los ruteadores y los anfitriones de red de redes para comunicar la información de control o de error. Veremos que los ruteadores utilizan el mecanismo para reportar problemas y que los anfitriones lo emplean para comprobar si los destinos son accesibles.

9.2 El Protocolo de mensajes de control de Internet

En el sistema sin conexión que hemos descrito hasta ahora, cada ruteador opera de manera autónoma, ruteando o entregando los datagramas que llegan sin coordinarse con el transmisor original. El sistema trabaja bien si todas las máquinas funcionan de manera correcta y si están de acuerdo respecto a las rutas. Por desgracia, ningún sistema funciona bien todo el tiempo. Además de las fallas en las líneas de comunicación y en los procesadores, el IP tiene fallas en la entrega de datagramas

cuando la máquina de destino está desconectada temporal o permanentemente de la red, cuando el contador de tiempo de vida expira, o cuando los ruteadores intermedios se congestionan tanto que no pueden procesar el tráfico entrante. La más importante diferencia entre tener una sola red implantada con hardware dedicado y tener una red de redes implantada con software es que, en el primer caso, el diseñador puede añadir hardware especial para informar a los anfitriones conectados cuando surge un problema. En una red de redes, que no tiene un mecanismo de hardware como el anterior, un transmisor no puede indicar si ocurrió una falla en la entrega, originada por un mal funcionamiento local o uno remoto. La depuración se vuelve muy difícil. El protocolo IP, por sí mismo, no contiene nada para ayudar al transmisor a comprobar la conectividad ni para ayudarle a aprender sobre dichas fallas.

Para permitir que los ruteadores en una red de redes reporten los errores o proporcionen información sobre circunstancias inesperadas, los diseñadores agregaron a los protocolos TCP/IP un mecanismo de mensajes de propósito especial. El mecanismo, conocido como *Protocolo de Mensajes de Control Internet (ICMP)*, se considera como parte obligatoria del IP y se debe incluir en todas las implantaciones IP.

Al igual que el resto del tráfico, los mensajes ICMP viajan a través de la red de redes en la porción de datos de los datagramas IP. Sin embargo, el destino final de un mensaje ICMP no es un programa de aplicación ni un usuario en la máquina de destino, sino el software de Protocolo Internet en dicha máquina. Esto es, cuando llega un mensaje de error ICMP, el módulo de software ICMP lo maneja. Por supuesto, si el ICMP determina que un protocolo de un nivel más alto o un programa de aplicación causaron un problema, notificará al módulo apropiado. Podemos resumir que:

> *El Protocolo de Mensajes de Control Internet permite que los ruteadores envíen mensajes de error o de control hacia otros ruteadores o anfitriones; el ICMP proporciona comunicación entre el software del Protocolo Internet en una máquina y el mismo software en otra.*

En principio diseñado para permitir que los ruteadores reporten a los anfitriones las causas de los errores en la entrega, el ICMP no se restringe sólo a los ruteadores. Aunque las reglas y normas limitan el uso de algunos mensajes ICMP, cualquier máquina puede enviar un mensaje ICMP a cualquier otra. Por lo tanto, un anfitrión puede utilizar el ICMP para comunicarse con un ruteador o con otro anfitrión. La mayor ventaja de permitir que los anfitriones utilicen el ICMP es que proporciona un solo mecanismo que se utiliza para todos los mensajes de información y de control.

9.3 Reporte de errores contra corrección de errores

Técnicamente, el ICMP es un *mecanismo de reporte de errores*. Proporciona una forma para que los ruteadores que encuentren un error lo reporten a la fuente original. Aunque la especificación del protocolo subraya los usos deseables del ICMP y sugiere acciones posibles para responder a los reportes de error, el ICMP no especifica del todo la acción que debe tomarse para cada posible error. En resumen,

Cuando un datagrama causa un error, el ICMP sólo puede reportar la condición del error a la fuente original del datagrama; la fuente debe relacionar el error con un programa de aplicación individual o debe tomar alguna otra acción para corregir el problema.

La mayor parte de los errores provienen de la fuente original, pero otros no. Sin embargo, debido a que el ICMP reporta los problemas a la fuente original, no se puede utilizar para informar los problemas a los ruteadores intermedios. Por ejemplo, suponga que un datagrama sigue un camino a través de una secuencia de ruteadores, R_1, R_2..., R_k. Si R_k tiene información de ruteo incorrecta y, por error, rutea el datagrama hacia el ruteador R_E, éste no podrá utilizar el ICMP para reportar el error a R_k; el ICMP sólo puede enviar un reporte a la fuente original. Por desgracia, la fuente original no tiene ninguna responsabilidad sobre el problema ni sobre el control del ruteador que se equivocó. De hecho, quizá la fuente no sea capaz de determinar qué ruteador causó el problema.

¿Por qué restringir el ICMP para comunicarse sólo con la fuente original? La respuesta debe ser clara si recordamos nuestro análisis sobre formatos de datagramas y sobre ruteo en los capítulos anteriores. Un datagrama sólo contiene campos que especifican la fuente original y el último destino; no contiene un registro completo de su viaje a través de la red de redes (a excepción de casos inusuales en los que se utiliza la opción de registro de ruta). Además, como los ruteadores pueden establecer y cambiar sus propias tablas de ruteo, no existe un conocimiento global de las rutas. Por lo tanto, cuando un datagrama llega a un ruteador, es imposible conocer el camino que siguió para llegar hasta ahí. Si el ruteador detecta un problema, no puede saber que grupo de máquinas intermedias procesaron el datagrama, así que no puede informarles del problema. En vez de descartar discretamente el datagrama, el ruteador utiliza el ICMP para informar a la fuente original que ocurrió un problema, y confía en que los administradores del anfitrión cooperarán con los administradores de red para localizarlo y corregirlo.

9.4 Entrega de mensajes ICMP

Los mensajes ICMP requieren dos niveles de encapsulación, como se muestra en la figura 9.1. Cada mensaje ICMP viaja a través de la red de redes en la porción de datos de un datagrama IP, el cual viaja a través de cada red física en la porción de datos de una trama. Los datagramas que llevan mensajes ICMP se rutean exactamente como los que llevan información de usuario; no existe ni una confiabilidad ni una prioridad adicionales. Por lo tanto, los mensajes de error se pueden perder o descartar. Además, en una red congestionada, el mensaje de error puede causar congestionamiento adicional. Hay una excepción en los procedimientos de manejo de errores si un datagrama IP que lleva un mensaje ICMP causa un error. Esta excepción, diseñada para evitar el problema de tener mensajes de error sobre mensajes de error, especifica que los mensajes ICMP no se generan por errores resultantes de datagramas que llevan mensajes de error ICMP.

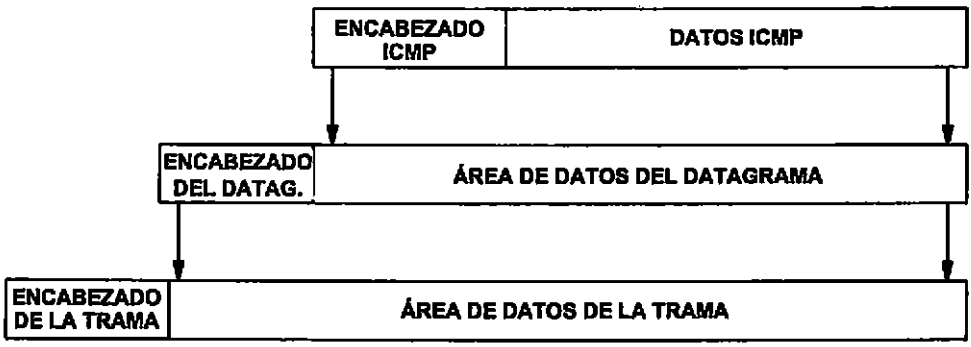


Figura 9.1 Dos niveles de la encapsulación ICMP. El mensaje ICMP se encapsula en un datagrama IP que, a su vez, se encapsula en una trama para su transmisión. Para identificar el ICMP, el campo de protocolo del datagrama contiene el valor *1*.

Es importante tener en mente que aunque los mensajes ICMP se encapsulan y envían mediante el IP, el ICMP no se considera como un protocolo de nivel más alto sino como una parte obligatoria del IP. La razón de utilizar el IP para entregar mensajes ICMP es que quizá necesiten viajar a través de muchas redes físicas para alcanzar su destino final. Por lo tanto, no se pueden entregar sólo por medio de transporte físico.

9.5 Formato de los mensajes ICMP

Aunque cada mensaje ICMP tiene su propio formato, todos comienzan con los mismos tres campos; un campo *TYPE (TIPO)* de mensaje, de 8 bits y números enteros, que identifica el mensaje; un campo *CODE (CÓDIGO)*, de 8 bits, que proporciona más información sobre el tipo de mensaje, y un campo *CHECKSUM (SUMA DE VERIFICACIÓN)*, de 16 bits (el ICMP utiliza el mismo algoritmo aditivo de suma de verificación que el IP, pero la suma de verificación del ICMP sólo abarca el mensaje ICMP). Además, los mensajes ICMP que reportan errores siempre incluyen el encabezado y los primeros 64 bits de datos del datagrama que causó el problema.

La razón de regresar más que el encabezado del datagrama únicamente es para permitir que el receptor determine de manera más precisa qué protocolo(s) y qué programa de aplicación son responsables del datagrama. Como veremos más adelante, los protocolos de más alto nivel del grupo TCP/IP están diseñados para codificar información crucial en los primeros 64 bits.

El campo *TYPE (TIPO)* de ICMP define el significado del mensaje así como su formato. Los tipos incluyen:

Campo de tipo	Tipo de Mensaje ICMP
0	Respuesta de Eco
3	Destino inaccesible
4	Disminución de origen
5	Redireccionar (cambiar una ruta)
8	Solicitud de Eco
11	Tiempo excedido para un datagrama
12	Problema de parámetros en un datagrama
13	Solicitud de timestamp
14	Respuesta de timestamp
15	Solicitud de información (obsoleto)
16	Respuesta de información (obsoleto)
17	Solicitud de máscara de dirección
18	Respuesta de máscara de dirección

En las siguientes secciones se describe cada uno de estos mensajes y se proporciona detalles sobre su formato y su significado.

9.6 Prueba de accesabilidad y estado de un destino (Ping)

Los protocolos TCP/IP proporcionan funciones para ayudar a los gerentes o usuarios de redes a identificar los problemas que ocurran en la red. Una de las herramientas de depuración más utilizadas incluye los mensajes ICMP de *echo request (solicitud de eco)* y *echo reply (respuesta de eco)*. Un anfitrión o un ruteador envía un mensaje ICMP de solicitud de eco hacia un destino específico. Cualquier máquina que recibe una solicitud de eco, formula una respuesta y la regresa al transmisor original. La solicitud contiene un área opcional de datos: la respuesta contiene una copia de los datos enviados en la solicitud. La solicitud de eco y su respuesta asociada se pueden utilizar para comprobar si un destino es alcanzable y si responde. Debido a que tanto la solicitud como la respuesta viajan en datagramas IP, la recepción exitosa de una respuesta verifica que las piezas principales del sistema de transporte están funcionando bien. Primero, el software IP en la computadora de origen debe rutear el datagrama. Segundo, los ruteadores intermedios entre el origen y el destino deben funcionar bien y rutear correctamente el datagrama. Tercero, la máquina de destino debe estar funcionando (al menos debe responder a las interrupciones), y tanto el software ICMP como el IP deben estar funcionando. Por último, todos los ruteadores a lo largo del camino de regreso deben tener rutas correctas.

En muchos sistemas, el comando que llama el usuario para enviar solicitudes de eco ICMP se conoce como *ping*. Las versiones más sofisticadas de ping envían una serie de solicitudes de eco ICMP, capturan las respuestas y proporcionan estadísticas sobre la pérdida de datagramas. Permiten que el usuario especifique la longitud de los datos que se envían, así como el intervalo entre solicitudes. Las versiones menos sofisticadas sólo envían una solicitud de eco ICMP y esperan la respuesta.

9.7 Formato de los mensajes de solicitud de eco y de respuesta

En la figura 9.2, se muestra el formato de los mensajes de solicitud de eco y de respuesta.

0	8	16 31
TIPO (8 o 0)	CÓDIGO (0)	SUMA DE VERIFICACIÓN
IDENTIFICADOR		NÚMERO DE SECUENCIA
DATOS OPCIONALES		
...		

Figura 9.2 Formato del mensaje ICMP de solicitud de eco o de respuesta.

El campo indicado como *OPTIONAL DATA (DATOS OPCIONALES)* es un campo de longitud variable que contiene los datos que se regresarán al transmisor. Una respuesta de eco siempre regresa exactamente los mismos datos que se recibieron en la solicitud. Los campos *IDENTIFIER (IDENTIFICADOR)* y *SEQUENCE NUMBER (NÚMERO DE SECUENCIA)* los utiliza el transmisor para responder a las solicitudes. El valor del campo *TYPE (TIPO)* especifica si el mensaje es una solicitud (*8*) o una respuesta (*0*).

9.8 Reporte de destinos no accesibles

Cuando un ruteador no puede direccionar o entregar un datagrama IP, envía un mensaje de *destino no accesible* a la fuente original, utilizando el formato que se muestra en la figura 9.3.

0	8	16 31
TIPO (3)	CÓDIGO (0-12)	SUMA DE VERIFICACIÓN
NO UTILIZADO (DEBE SER CERO)		
ENCABEZADO DE RED DE REDES + PRIMEROS 64 BITS DEL DATAGRAMA		
...		

Figura 9.3 Formato del mensaje ICMP de destino inaccesible.

El campo *CODE (CÓDIGO)* de un mensaje de destino no accesible contiene un número entero que describe con más detalle el problema. Los valores posibles son:

Valor de Código	Significado
0	Red inaccesible
1	Anfitrión inaccesible
2	Protocolo inaccesible
3	Puerto inaccesible
4	Se necesita fragmentación y configuración DF
5	Falla en la ruta de origen
6	Red de destino desconocida
7	Anfitrión de destino desconocido
8	Anfitrión de origen aislado
9	Comunicación con la red de destino administrativamente prohibida
10	Comunicación con el anfitrión de destino administrativamente prohibida
11	Red inaccesible por el tipo de servicio
12	Anfitrión inaccesible por el tipo de servicio

Aunque el IP es un mecanismo de entrega con el mejor esfuerzo, el descarte de datagramas no se debe tomar a la ligera. Siempre que un error evite que un ruteador direccione o entregue un datagrama, el ruteador envía al origen un mensaje de destino no accesible y luego *suelta* (por ejemplo, descarta) el datagrama. Los errores de red no accesible por lo general implican fallas en el ruteo.[1] Debido a que los mensajes de error ICMP contienen un prefijo del datagrama que causó el problema, la fuente sabrá exactamente qué dirección no es accesible.

Los destinos pueden no ser accesibles ya sea porque el hardware esté temporalmente fuera de servicio, porque el transmisor haya especificado una dirección de destino no existente o (en circunstancias poco comunes) porque el ruteador no tenga una ruta para la red de destino. Nótese que aunque los ruteadores reportan las fallas que encuentran, quizá no tengan conocimiento de todas las fallas de entrega. Por ejemplo, si la máquina de destino se conecta a una red Ethernet, el hardware de red no proporciona acuses de recibo. Por lo tanto, un ruteador puede seguir enviando paquetes hacia un destino cuando éste se encuentre apagado, sin recibir ninguna indicación de que los paquetes no se están entregando. En resumen:

> *Aunque un ruteador envía un mensaje de destino no accesible cuando encuentra un datagrama que no se puede direccionar o entregar, no puede detectar la totalidad de dichos errores.*

El significado de los mensajes de protocolo y puerto no accesibles se aclarará cuando estudiemos cómo los protocolos de un nivel más alto utilizan puntos abstractos de destino, llamados *puertos*. La mayor parte de los mensajes restantes se explican por sí mismos. Si el datagrama contiene una opción de ruta de origen con una ruta incorrecta, activará un mensaje de falla en la *ruta de origen*. Si un ruteador necesita fragmentar un datagrama pero está activado el bit de "no fragmentar", el ruteador enviará un mensaje de *necesidad de fragmentación* hacia la fuente.

[1] Existe una excepción para los ruteadores que utilizan el esquema de direccionamiento de subred en el capítulo 10. Reportan una falla en el ruteo de subred con un mensaje ICMP de anfitrión no accesible.

9.9 Control de congestionamientos y de flujo de datagramas

Debido a que el IP funciona sin conexión, un ruteador no puede reservar memoria o recursos de comunicación antes de recibir datagramas. Como resultado, los ruteadores se pueden saturar con el tráfico, condición conocida como *congestionamiento*. Es importante entender que el congestionamiento puede surgir por dos razones totalmente diferentes. Primero, una computadora de alta velocidad puede ser capaz de generar tráfico de forma más rápida de lo que una red lo puede transferir. Por ejemplo, imagínese una supercomputadora que genera tráfico para la red de redes. Los datagramas pueden necesitar pasar a través de una red de área amplia (WAN) más lenta, aunque la supercomputadora se conecte a una red de área local de alta velocidad. El congestionamiento ocurrirá en el ruteador que conecta la LAN con la WAN, ya que los datagramas llegan más rápido de lo que se pueden enviar. Segundo, si muchas computadoras necesitan enviar datagramas al mismo tiempo a través de un solo ruteador, éste se puede congestionar, aunque ningún origen por sí mismo cause el problema.

Cuando los datagramas llegan demasiado rápido para que un anfitrión o un ruteador los procesen, éstos los ponen temporalmente en una cola de espera en memoria. Si los datagramas son parte de una racha pequeña, este procedimiento de memorización temporal soluciona el problema. Si el tráfico continúa, llega un momento en el que se le acaba la memoria al anfitrión o al ruteador, y deben descartar los demás datagramas que lleguen. Una máquina utiliza mensajes ICMP de *disminución de tasa al origen* (*source quench*) para reportar el congestionamiento a la fuente original. Un mensaje de disminución de tasa al origen es una solicitud para que la fuente reduzca la velocidad de transmisión de datagramas. Por lo general, los ruteadores congestionados envían un mensaje de disminución de tasa al origen por cada datagrama que descartan. Los ruteadores también pueden utilizar técnicas más sofisticadas para el control de congestionamientos. Algunos, monitorean el tráfico entrante y reducen las fuentes que tienen las velocidades más altas de transmisión de datagramas. Otros, intentan evitar los congestionamientos al enviar solicitudes de disminución cuando sus colas de espera crecen, pero antes de que se saturen.

No existe ningún mensaje ICMP para revertir el efecto de una disminución de tasa al origen. En vez de eso, un anfitrión que reciba mensajes de disminución para un destino *D*, baja la velocidad de envío de datagramas hacia *D*, hasta que deja de recibir los mensajes de disminución de tasa al origen: luego, aumenta de manera gradual la velocidad en tanto no reciba más solicitudes de disminución de tasa al origen.

9.10 Formato de disminución de tasa al origen

Además de los campos normales, ICMP como *TYPE, CODE, CHECKSUM*, y un campo no utilizado de 32 bits, los mensajes de disminución de tasa al origen tienen un campo que contiene un prefijo de datagrama. En la figura 9.4 se ilustra el formato. Como sucede en la mayor parte de los mensajes ICMP que reportan un error, el campo antes mencionado contiene un prefijo del datagrama que activó la solicitud de disminución de origen.

0	8	16 31
TIPO (4)	CÓDIGO (0)	SUMA DE VERIFICACIÓN
NO UTILIZADO (DEBE SER CERO)		
ENCABEZADO DE RED DE REDES + PRIMEROS 64 BITS DEL DATAGRAMA		
...		

Figura 9.4 Formato del mensaje ICMP de disminución de origen. Un ruteador congestionado envía un mensaje de disminución de origen cada vez que descarta un datagrama; el prefijo de datagrama identifica el datagrama que se descartó.

9.11 Solicitudes para cambio de ruta desde los ruteadores

Por lo general, las tablas de ruteo de una red de redes se mantienen sin cambios por grandes periodos de tiempo. Los anfitriones las inician desde un archivo de configuración en el arranque del sistema y los administradores de sistema muy esporádicamente hacen cambios de ruteo durante la operación normal. Si cambia la topología de la red, las tablas de ruteo en un ruteador o en un anfitrión pueden volverse incorrectas. Un cambio puede ser temporal (por ejemplo, cuando se necesita reparar el hardware) o permanente (cuando se agrega una nueva red a la red de redes). Como veremos en los siguientes capítulos, los ruteadores intercambian en forma periódica información de ruteo para incorporar los cambios en la red y para mantener actualizadas sus rutas. Por lo tanto, como regla general:

Se asume que los ruteadores conocen rutas correctas; los anfitriones comienzan con información mínima de ruteo y aprenden nuevas rutas de los ruteadores.

Para ayudar a que sigan esta ruta y para evitar la duplicación de información de ruteo en el archivo de configuración de cada anfitrión, esta configuración especifica la menor información posible de ruteo necesaria para comunicarse (por ejemplo, la dirección de un solo ruteador). Por lo tanto, el anfitrión arranca con información mínima y confía en los ruteadores para actualizar su tabla de ruteo. En un caso especial, cuando un ruteador detecta un anfitrión que utiliza una ruta no óptima, le envía al anfitrión un mensaje ICMP, llamado *redireccionar (redirect)*, solicitándole que cambie sus rutas. El ruteador también direcciona al datagrama original hacia su destino.

La ventaja del esquema de redireccionamiento ICMP es la simplicidad: permite que un anfitrión inicie conociendo solamente un ruteador en la red local. El ruteador inicial genera mensajes de redireccionamiento siempre que un anfitrión envía un datagrama para el que existe una ruta mejor. La tabla de ruteo del anfitrión permanece reducida y, aun así, contiene rutas óptimas para todos los destinos en uso.

Sin embargo, redireccionar mensajes no soluciona el problema de propagar rutas de manera general, ya que están limitados a la interacción entre un ruteador y un anfitrión en una red conectada directamente. En la figura 9.5, se ilustra esta limitación. En la figura, asuma que la fuente *S* le

envía un datagrama al destino *D*. También asuma que el ruteador R_1 rutea de manera incorrecta el datagrama a través del ruteador R_2, en vez de hacerlo a través del ruteador R_4 (por ejemplo, R_1 selecciona de manera incorrecta un camino más largo). Cuando el ruteador R_5 recibe el datagrama, no puede enviar un mensaje ICMP de redireccionamiento a R_1, ya que no conoce su dirección. En los capítulos siguientes, se explora el problema de cómo propagar rutas a través de muchas redes.

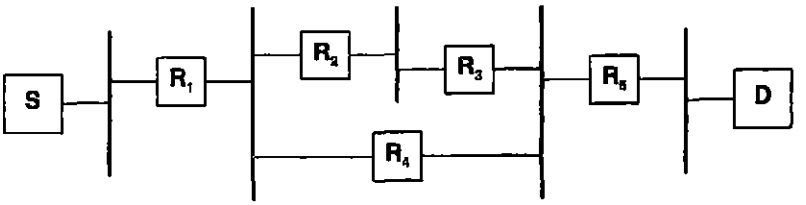


Figura 9.5 Los mensajes ICMP de redireccionamiento no proporcionan ruteo entre ruteadores. En este ejemplo, el ruteador R_5 no puede redireccionar hacia R_1 para utilizar el camino más corto para los datagramas *S* al *D*.

Además de los campos obligatorios de *TYPE (TIPO), CODE (CÓDIGO) y CHECKSUM (SUMA DE VERIFICACIÓN)*, cada mensaje de redireccionamiento contiene un campo de 32 bits, llamado *ROUTER INTERNET ADDRESS (DIRECCIÓN DE RED DE REDES DEL RUTEADOR)*, y un campo *HEADER (ENCABEZADO)* como se muestra en la figura 9.6.

0	8	16 31
TIPO (5)	CÓDIGO (0-3)	SUMA DE VERIFICACIÓN
DIRECCIÓN DE RED DE REDES DEL RUTEADOR		
ENCABEZADO DE RED DE REDES + PRIMEROS 64 BITS DEL DATAGRAMA		
...		

Figura 9.6 Formato del mensaje ICMP de redireccionamiento.

El campo *ROUTER INTERNET ADDRESS* contiene la dirección de un ruteador que el anfitrión utilizará para alcanzar el destino mencionado en el encabezado del datagrama. El campo *INTERNET HEADER* contiene el encabezado IP, más los siguientes 64 bits del datagrama que activó el mensaje. Por lo tanto, un anfitrión que recibe un redireccionamiento ICMP examina el prefijo del datagrama para determinar la dirección de destino. El campo *CODE* de un mensaje ICMP de redireccionamiento especifica con mayor detalle cómo interpretar la dirección de destino, basándose, como se muestra a continuación, en los valores asignados:

Valor de Código	Significado
0	Redireccionar datagramas para la red (ahora obsoleto)
1	Redireccionar datagramas para el anfitrión
2	Redireccionar datagramas para el tipo de servicio[2] y la red
3	Redireccionar datagramas para el tipo de servicio y el anfitrión

Como regla general, los ruteadores envían solicitudes ICMP de redireccionamiento sólo a los anfitriones y no a otros ruteadores. En los siguientes capítulos, veremos que los ruteadores utilizan otros protocolos para intercambiar información de ruteo.

9.12 Detección de rutas circulares o excesivamente largas

Debido a que los ruteadores en una red de redes computan un salto al siguiente ruteador, utilizando tablas locales, los errores en dichas tablas pueden producir un *ciclo de ruteo* para algún destino, *D*. Un ciclo de ruteo puede consistir en dos ruteadores, cada uno ruteando al otro un datagrama para el destino *D*, o puede consistir en muchos ruteadores haciendo lo mismo. Cuando muchos ruteadores forman un ciclo, cada uno rutea un datagrama para el destino *D* y hacia el siguiente ruteador dentro del ciclo. Si un datagrama entra en un ciclo de ruteo, recorrerá indefinidamente y de manera circular todos los ruteadores. Como se mencionó con anterioridad, para evitar que los datagramas circulen indefinidamente en una red de redes TCP/IP, cada datagrama IP contiene un contador de tiempo de vida, conocido como *conteo de saltos*. Un ruteador disminuye el contador de tiempo de vida siempre que procese el datagrama y lo descarta cuando el conteo llega a cero.

Siempre que un ruteador descarta un datagrama ya sea porque su conteo de saltos llega a cero o porque ocurre una terminación de tiempo mientras espera fragmentos de un datagrama, envía un mensaje ICMP de *tiempo excedido* a la fuente del datagrama, utilizando el formato que se muestra en la figura 9.7.

0	8	16 … 31
TIPO (11)	CÓDIGO (0 o 1)	SUMA DE VERIFICACIÓN
NO UTILIZADO (DEBE SER CERO)		
ENCABEZADO DE RED DE REDES + PRIMEROS 64 BITS DEL DATAGRAMA		
...		

Figura 9.7 Formato del mensaje ICMP de tiempo excedido. Un ruteador envía este mensaje siempre que se descarte un datagrama cuando el campo de tiempo de vida en el encabezado del datagrama llega a cero o cuando su temporizador de reensamblado expira mientras éste espera fragmentos.

[2] Recuerde que cada encabezado IP especifica un tipo de servicio utilizado para el ruteo.

En el campo *CODE* se explica la naturaleza de la terminación de tiempo:

Valor de Código	Significado
0	Conteo de tiempo de vida excedido
1	Tiempo para el reensamblado de fragmentos exceed'do

El reensamblado de fragmentos se refiere a la tarea de recolectar todos los fragmentos de un datagrama. Cuando llega el primer fragmento de un datagrama, el anfitrión que lo recibe arranca un temporizador y considera como error que dicho temporizador expire antes de que lleguen todas las piezas del datagrama. El valor *1* para el campo Code se utiliza para informar dichos errores al transmisor; se envía un mensaje por cada error.

9.13 Reporte de otros problemas

Cuando un ruteador o un anfitrión encuentran problemas que no se han cubierto con los mensajes ICMP de error anteriores (por ejemplo, un datagrama con encabezado incorrecto), envían un mensaje de *problema de parámetros* a la fuente original. Una causa posible de dichos problemas ocurre cuando los argumentos para una opción son incorrectos. El mensaje, formateado como se muestra en la figura 9.8, sólo se envía cuando el problema es tan severo que se tiene que descartar el datagrama.

0	8	16 … 31
TIPO (12)	CÓDIGO (0 o 1)	SUMA DE VERIFICACIÓN
INDICADOR	NO UTILIZADO (DEBE SER CERO)	
ENCABEZADO DE RED DE REDES + PRIMEROS 64 BITS DEL DATAGRAMA		
...		

Figura 9.8 Formato del mensaje ICMP de problema de parámetros. Dichos mensajes sólo se envían cuando el problema origina que se descarte el datagrama.

Para lograr que el mensaje no sea ambiguo, el transmisor utiliza el campo *POINTER* en el encabezado del mensaje para identificar el octeto del datagrama que causó el problema. El código *1* se utiliza para informar que falta la opción requerida (por ejemplo, una opción de seguridad en la comunidad militar); el campo *POINTER* no se utiliza para el código *1*.

9.14 Sincronización de relojes y estimación del tiempo de tránsito

Aunque las máquinas en una red de redes se pueden comunicar, por lo general operan de forma independiente, con cada máquina, manteniendo su propia noción de la hora actual. Los relojes que varían demasiado pueden confundir a los usuarios de software de sistemas distribuidos. El grupo de protocolos TCP/IP incluye muchos protocolos que se pueden utilizar para sincronizar los relojes. Una de las técnicas más sencillas se vale de un mensaje ICMP para obtener la hora de otra máquina. Una máquina solicitante envía un mensaje ICMP de *solicitud de timestamp (marca de hora)* a otra, solicitándole que informe su valor actual para la hora del día. La máquina receptora envía una *respuesta de timestamp (marca de hora)* a quien la solicitó. En la figura 9.9 se muestra el formato de los mensajes de solicitud y respuesta de timestamp (marca de hora).

0	8	16 … 31
TIPO (13 o 14)	CÓDIGO (0)	SUMA DE VERIFICACIÓN
IDENTIFICADOR		NÚMERO DE SECUENCIA
ORIGINAR TIMESTAMP		
RECIBIR TIMESTAMP		
TRANSMITIR TIMESTAMP		

Figura 9.9 Formato del mensaje ICMP de solicitud de timestamp o de respuesta de timestamp.

El campo *TYPE* identifica el mensaje como solicitud (13) o como respuesta (14): los campos *IDENTIFIER y SEQUENCE NUMBER* los utiliza la fuente para asociar las solicitudes con las respuestas. Los campos restantes especifican la hora, en milisegundos desde la media noche, en Tiempo Universal.[3] El campo *ORIGINATE TIMESTAMP* es llenado por la fuente original justo antes de transmitir el paquete, el campo *RECEIVE TIMESTAMP* se llena inmediatamente al recibir una solicitud y el campo *TRANSMIT TIMESTAMP se llena justo antes de transmitir la respuesta.*

Los anfitriones utilizan estos tres campos para computar estimaciones del tiempo de retraso entre ellos y para sincronizar sus relojes. Debido a que la respuesta incluye el campo *ORIGINATE TIMESTAMP*, un anfitrión puede computar el tiempo total requerido para que una solicitud viaje hasta un destino, se transforme en una respuesta y regrese. Debido a que la respuesta lleva tanto la hora en la que la solicitud ingresó a la máquina remota como la hora en la que se transmitió, el anfitrión puede computar el tiempo de tránsito de la red y, con ese valor, estimar las diferencias entre el reloj local y los remotos.

En la práctica, el cálculo preciso del retraso en los viajes redondos puede ser difícil y substancialmente restringe la utilidad de los mensajes ICMP timestamp. Claro está, para obtener un cálculo preciso del retraso en viajes redondos, se deben tomar medidas y promediarlas. Sin embargo,

---

[3] El Tiempo Universal se llamaba antes Tiempo del Meridiano de Greenwich; es la hora del día en el meridiano central.

el retraso del viaje redondo entre dos máquinas que se conectan a una gran red de redes puede variar de formar dramática, inclusive entre cortos periodos de tiempo. Además, recuerde que debido a que el IP es una tecnología de mejor esfuerzo, los datagramas se pueden perder, retrasar o entregarse en desorden. Por lo tanto, aún tomando muchas medidas no se garantiza la consistencia; quizá se necesite un análisis estadístico sofisticado para obtener cálculos precisos.

9.15 Solicitud de información y mensajes de respuesta

Los mensajes ICMP de *solicitud de información* y de *respuesta de información* (tipos 15 y 16) actualmente se consideran como obsoletos y no se deben utilizar. Originalmente se permitía que los anfitriones descubrieran su dirección de red en el arranque del sistema. Los protocolos actuales para la determinación de direcciones son RARP, descrito en el capítulo 6, y BOOTP, descrito en el capítulo 21.

9.16 Obtención de una máscara de subred

En el capítulo 10 se tratan los motivos para el direccionamiento de subred, así como los detalles de operación de las subredes. Por ahora, sólo es importante entender que, cuando los anfitriones utilizan el direccionamiento de subred, algunos bits en la porción hostid de su dirección IP identifican una red física. Para participar en el direccionamiento de subred, un anfitrión necesita saber qué bits de la dirección de red de redes de 32 bits corresponden a la red física, así como qué bits corresponden a los identificadores del anfitrión. La información necesaria para interpretar la dirección se representa en una cantidad de 32 bits llamada *máscara de subred (subnet mask)*.

Para aprender la máscara de subred utilizada para la red local, una máquina puede enviar un mensaje de *solicitud de máscara de subred* a un ruteador y recibir una *respuesta de máscara de subred*. La máquina que hace la solicitud puede enviar directamente el mensaje, si conoce la dirección del ruteador, o transmitir el mensaje por difusión. En la figura 9.10 se muestra el formato de un mensaje de máscara de subred.

0	8	16 31
TIPO (17 ó 18)	CÓDIGO (0)	SUMA DE VERIFICACIÓN
IDENTIFICADOR		NÚMERO DE SECUENCIA
MÁSCARA DE DIRECCIÓN		

Figura 9.10 Formato del mensaje ICMP de solicitud de máscara de red o de respuesta de máscara de red. Por lo general, los anfitriones transmiten por difusión una solicitud sin saber qué ruteador específico responderá.

El campo *TYPE* en un mensaje de máscara de dirección especifica si el mensaje es una solicitud (17) o una respuesta (18). Una respuesta contiene la máscara de dirección de subred en el campo *ADDRESS MASK*. Como es usual, los campos *IDENTIFIER* y *SEQUENCE NUMBER* permiten que una máquina asocie las solicitudes con las respuestas.

9.17 Resumen

La comunicación normal a través de una red de redes comprende el envío de mensajes de una aplicación en un anfitrión a otro anfitrión. Los ruteadores quizá necesiten comunicarse directamente con el software de red en un anfitrión en particular para reportar condiciones anormales o para enviar al anfitrión nueva información de ruteo.

El Protocolo de Mensajes de Control de Internet proporciona una comunicación extranormal entre ruteadores y anfitriones; es una parte integral y obligatoria del IP. El ICMP incluye mensajes de *disminución de tasa al origen* que retardan la velocidad de transmisión, mensajes de *redireccionamiento* que pueden utilizar los anfitriones cambiar su mesa de enrutado, y mensajes de "echo request/reply" que los anfitriones para determinar si se puede accesar un destino. Un mensaje ICMP viaja en el área de datos de un datagrama IP y tiene tres campos de longitud fija al comienzo del mensaje: el campo *type (tipo)*, un campo *code (código)* y el campo ICMP *checksum (suma de verificación)*. El tipo de mensaje determina el formato del resto del mensaje, así como su significado.

PARA CONOCER MÁS

Tanto Tanenbaum (1981) como Stallings (1985) tratan de manera general los mensajes de control y los relacionan con varios protocolos de red. El tema central no es cómo enviar mensajes de control sino cuándo. Grange y Gien (1979), así como Driver, Hopewell y Iaquinto (1979) se concentran en un problema para el que los mensajes de control son esenciales, a saber, el control de flujo. Gerla y Kleinrock (1980) comparan de forma analítica las estrategias para el control de flujo.

El Protocolo de Mensajes de Control Internet que aquí se describe es un estándar TCP/IP definido por Postel (RFC 792) y actualizado por Braden (RFC (1122). Nagle (RFC 896) analiza los mensajes ICMP de disminución de origen y muestra cómo los ruteadores deberían utilizarlos para manejar el control de congestionamientos. Prue y Postel (RFC 1016) analizan una técnica más reciente que emplean los ruteadores en respuesta a la disminución de origen. Nagle (1987) arguye que el congestionamiento siempre es importante en las redes de paquetes conmutados. Mogul y Postel (RFC 950) tratan las subredes y mensajes de respuesta. Por último, Jain, Ramakrishnan y Chiu (1987) exponen cómo los ruteadores y los protocolos de transporte podrían cooperar para evitar el congestionamiento.

Para obtener un análisis sobre los protocolos para la sincronización de relojes, consulte Mills (RFC 956, 957 y 1305).

EJERCICIOS

9.1 Diseñe un experimento para registrar cuántos tipos de mensajes ICMP aparecen en su red local durante un día.

9.2 Experimente si puede enviar paquetes a través de un ruteador, lo suficientemente rápido como para activar un mensaje ICMP de disminución de origen.

9.3 Diseñe un algoritmo que sincronice los relojes utilizando mensajes ICMP timestamp (marca de hora).

9.4 Revise si su computadora local contiene un comando *ping*. ¿Cómo es la interfaz del programa con los protocolos del sistema operativo? En particular, ¿el mecanismo permite que cualquier usuario cree un programa *ping* o dicho programa requiere de un privilegio especial? Explíquelo.

9.5 Asuma que todos los ruteadores envían mensajes ICMP de terminación de tiempo y que su software TCP/IP local devolverá dichos mensajes a un programa de aplicación. Utilice este esquema para construir un comando *traceroute* que reporte la lista de ruteadores entre la fuente y un destino en particular.

9.6 Si usted tiene conexión con Internet, intente utilizar el comando ping para llegar al anfitrión 128.10.2.1 (una máquina en la Universidad de Purdue).

9.7 ¿Un ruteador debe dar mayor prioridad a los mensajes ICMP que al tráfico normal? ¿Por qué?

9.8 Considere una Ethernet que tenga un anfitrión convencional, *H*, y 12 ruteadores conectados a ella. Encuentre una sola trama (ligeramente ilegal), que lleve un paquete IP de manera que, cuando el anfitrión *H* la envíe, provoque que *H* reciba exactamente 24 paquetes.

9.9 Compare los paquetes ICMP de disminución de origen con el esquema de 1 bit de Jain, utilizado en DECNET. ¿Cuál es una estrategia más efectiva para manejar los congestionamientos? ¿Por qué?

9.10 No existe ningún mensaje ICMP que permita que una máquina informe a la fuente que los errores de transmisión están provocando que los datagramas lleguen corrompidos. Explique por qué.

9.11 Según la pregunta anterior, ¿bajo qué circunstancias sería útil dicho mensaje?

9.12 ¿Los mensajes ICMP de error deberían contener una timestamp (marca de hora) que especifique cuándo se enviaron? ¿Por qué?

9.13 Trate de accesar un servidor en un anfitrión inexistente en su red local. También intente de comunicarse con un anfitrión inexistente en una red remota. ¿En qué caso recibe un mensaje de error? ¿Por qué?

9.14 Trate de utilizar *ping* con una dirección de difusión de red. ¿Cuántas computadoras contestan? Lea los documentos del protocolo para determinar si contestar una solicitud de difusión es obligatorio, recomendable, no recomendable o está prohibido.

10

Extensiones de dirección de subred y superred

10.1 Introducción

En el capítulo 4, se analizó el esquema original de direccionamiento en Internet y se presentó los tres formatos principales de las direcciones IP. En este capítulo, se examinan cuatro extensiones del esquema de direcciones IP, que permiten que una localidad utilice una sola dirección IP para muchas redes físicas. En él, se considera la motivación para las extensiones de dirección y se describen los mecanismos básicos para cada una. En particular, en este capítulo se presentan los detalles del esquema de subred que actualmente es parte del estándar TCP/IP.

10.2 Reseña de hechos importantes

En el capítulo 4 se trató el direccionamiento en las redes de redes y se presentó los fundamentos del esquema actual de las direcciones IP. Se dijo que las direcciones de 32 bits se asignan con cuidado para que las direcciones IP de todos los anfitriones de una red física tengan un prefijo en común. En el esquema original de las direcciones IP, los diseñadores pensaron al prefijo como la definición de la porción de red de una dirección de red de redes, y al remanente como la porción de anfitrión. La consecuencia que nos interesa es que:

En el esquema original de direccionamiento IP, cada red física tiene asignada una dirección única; cada anfitrión en la red tiene la dirección de red como prefijo de su dirección individual.

La mayor ventaja de dividir una dirección IP en dos partes surge del tamaño de las tablas de ruteo que necesitan los ruteadores. En vez de almacenar un registro de ruteo por cada anfitrión de destino, un ruteador puede tener un registro por cada red y examinar sólo la porción de red de la dirección de destino cuando tome decisiones de ruteo.

Recuerde que el TCP/IP incorpora muchos tamaños de red por el hecho de tener tres tipos principales de direcciones. Las redes que tienen asignadas direcciones tipo *A* dividen los 32 bits en una porción de red de 8 bits y una porción de anfitrión de 24 bits. Las direcciones tipo *B* dividen los 32 bits en porciones de red y de anfitrión de 16 bits; y las direcciones tipo *C* dividen la dirección en una porción de red de 24 bits y una porción de anfitrión de 8 bits.

Para entender las extensiones de dirección de este capítulo, es importante darse cuenta que las localidades tienen la libertad de modificar las direcciones y las rutas, siempre y cuando dichas modificaciones permanezcan ocultas para las demás localidades. Esto es, una localidad puede asignar y utilizar internamente direcciones IP de manera no usual siempre y cuando:

- Todos los anfitriones y los ruteadores en dicha localidad estén de acuerdo en seguir el esquema de direccibnamiento.
- Otras localidades en Internet puedan manejar las direcciones como en el esquema original.

10.3 Minimización de números de red

El esquema original de direccionamiento IP parece incluir todas las posibilidades, pero tiene una debilidad menor. ¿Cómo surgió esta debilidad? ¿Qué es lo que los diseñadores no vislumbraron? La respuesta es simple: el crecimiento. Debido a que los diseñadores trabajaban en un mundo de computadoras mainframe caras, visualizaron una red con cientos de redes y miles de anfitriones. No pensaron en las decenas de miles de redes pequeñas de computadoras personales que aparecerían de manera repentina en los años siguientes al diseño del TCP/IP.

El crecimiento es más visible en cuanto a las conexiones a Internet, cuyo tamaño se duplica cada nueve meses. La gran población de redes pequeñas resalta la importancia del esquema de Internet, ya que significa: (1) que se requiere mucho trabajo administrativo para manejar las direcciones de red, (2) que las tablas de ruteo de los ruteadores son muy grandes, y (3) que el espacio para las direcciones se acabará eventualmente. El segundo problema es importante porque significa que, cuando los ruteadores intercambian información de sus tablas de ruteo, la carga en la red de redes es alta, así como también lo es el esfuerzo computacional requerido por los ruteadores participantes. El tercer problema es crucial ya que el esquema original de direcciones no puede incorporar el número actual de redes en la red global Internet. En particular, no existen suficientes prefijos tipo B para cubrir todas las redes de tamaño mediano en Internet. La pregunta es: ¿cómo se puede minimizar el número de direcciones asignadas de red, en especial las de tipo B, sin destruir el esquema original de direccionamiento?

Para minimizar las direcciones de red, muchas redes físicas deben compartir el mismo prefijo IP de red. Para minimizar las direcciones tipo B, se deben utilizar direcciones tipo C. Claro está, se deben modificar los procedimientos de ruteo y todas las máquinas que se conectan a las redes afectadas deben entender las normas utilizadas.

La idea de compartir una dirección de red entre muchas redes físicas no es nueva y ha tomado muchas formas. Examinaremos tres de ellas: ruteadores transparentes, ARP sustituto (proxy ARP) y subredes IP estándar. También consideraremos el direccionamiento sin tipo, que es asignar muchas direcciones tipo C en vez de direcciones tipo B.

10.4 Ruteadores transparentes

El esquema de *ruteador transparente* se basa en la observación de que *una red que tiene asignada* una dirección IP tipo *A* se puede extender mediante un sencillo truco, ilustrado en la figura 10.1.

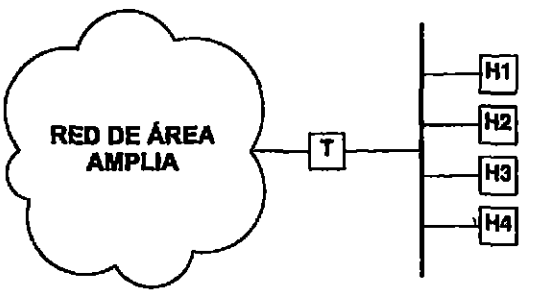


Figura 10.1 Ruteador transparente *T* que extiende una red de área amplia a muchos anfitriones en una localidad. Cada anfitrión parece tener una dirección IP en la WAN.

El truco consiste en hacer que una red física, por lo general una WAN, realice el multiplexado de muchas conexiones de anfitrión a través de un solo puerto. Como se muestra en la figura 10.1, un ruteador *T*, de propósito especial, conecta un solo puerto de anfitrión de la red de área amplia a una red de área local. *T* se conoce como *ruteador transparente*, debido a que los otros anfitriones y ruteadores en la WAN no saben que existe.

La red de área local no posee su propio prefijo IP; los anfitriones conectados tienen asignadas direcciones como si se conectaran de manera directa con la WAN. El ruteador transparente realiza el demultiplexado de los datagramas que llegan de la WAN al enviarlos hacia el anfitrión apropiado (por ejemplo, utilizando una tabla de direcciones). El ruteador transparente también acepta datagramas de los anfitriones en la red de área local y los rutea a través de la WAN hacia su destino.

Para realizar de manera eficiente el demultiplexado, los ruteadores transparentes a menudo dividen la dirección IP en muchas partes y codifican la información dentro de las partes no utiliza-

das. Por ejemplo, ARPANET tenía asignada la dirección de red tipo *A 10.0.0.0*. Cada nodo de conmutación de paquetes (PSN) tenía una dirección única de números enteros. Internamente, ARPANET trataba cualquier dirección IP de 4 octetos con la forma *10.p.u.i*, como cuatro octetos separados que especificaban una red (*10*), un puerto específico en el PSN de destino (*p*), y un PSN de destino (*i*). El octeto *u* no tenía interpretación. Por consiguiente, tanto la dirección de ARPANET *10.2.5.37* como la *10.2.9.37*, se refieren al anfitrión *2* en el PSN *37*. Un ruteador transparente conectado al PSN *37* en el puerto *2* puede utilizar el octeto *u* para decidir qué anfitrión real debe recibir un datagrama. La WAN por sí misma no necesita enterarse de todos los anfitriones que se encuentran más allá del PSN.

Los ruteadores transparentes tienen ventajas y desventajas cuando se les compara con los ruteadores convencionales. La ventaja principal es que requieren menos direcciones de red, ya que la red de área local no necesita un prefijo IP por separado. Otra ventaja es que pueden incorporar el balanceo de carga. Esto es, si dos ruteadores transparentes se conectan a la misma red de área local, se puede dividir el tráfico hacia ellos. En comparación, los ruteadores convencionales sólo pueden manejar una ruta hacia cierta red.

Una desventaja de los ruteadores transparentes es que sólo trabajan con redes que tienen un espacio de direcciones grande, de donde escoger las de los anfitriones. Por lo tanto, trabajan bien con las redes tipo *A*, y no así con las redes tipo *C*. Otra desventaja es que, como no son ruteadores convencionales, los ruteadores transparentes no proporcionan los mismos servicios. En particular, los ruteadores transparentes quizá no participen del todo en los protocolos ICMP, o de manejo de red como SNMP. Por lo tanto, no generan respuestas de eco ICMP (por ejemplo, no se puede utilizar "ping" para determinar si un ruteador transparente está operando).

10.5 ARP sustituto (proxy ARP)

Los términos *ARP sustituto (proxy, ARP) promiscuo* y *ARP hack*, se refieren a la segunda técnica utilizada para transformar un solo prefijo IP de red en dos direcciones físicas. La técnica, que sólo se aplica en redes que utilizan ARP para convertir direcciones de red en direcciones físicas, se puede explicar mejor mediante un ejemplo. En la figura 10.2 se ilustra la situación.

En la figura, dos redes comparten una sola dirección IP. Imagine que la etiquetada como *Red Principal* era la red original y segunda, etiquetada como *Red Oculta*, se agregó después. *R*, que es el ruteador que conecta las dos redes, sabe qué anfitriones residen en cada red física y utiliza ARP para mantener la ilusión de que solamente existe una red. Para dar esa apariencia, *R* mantiene totalmente oculta la localización de los anfitriones, permitiendo que las demás máquinas en la red se comuniquen como si estuvieran conectadas de manera directa. En nuestro ejemplo, cuando el anfitrión H_1 necesita comunicarse con el anfitrión H_4, primero llama a ARP para convertir la dirección IP de H_4 en una dirección física. Una vez que tiene la dirección física, H_1 puede enviarle directamente el datagrama.

Debido a que el ruteador *R* corre software proxy ARP, *R* captura la solicitud transmitida por difusión de H_1, decide que la máquina en cuestión reside en la otra red física y responde la solicitud ARP enviando su propia dirección física. H_1 recibe la respuesta ARP, instala la asociación en su tabla ARP y la utiliza para enviar a *R* los datagramas destinados a H_4. Cuando *R* recibe un datagrama, busca en una tabla especial de ruteo para determinar cómo rutear el datagrama. *R* debe encami-

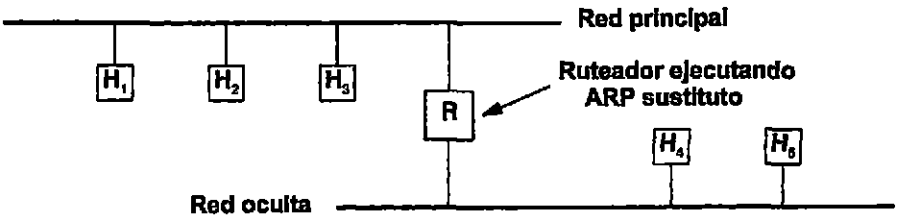


Figura 10.2 La técnica de ARP sustituto (ARP hack) permite que una dirección de red se comparta entre dos redes físicas. El ruteador *R* contesta solicitudes ARP en cada red para los anfitriones en otra, proporcionando su dirección de hardware y ruteando datagramas de manera correcta en cuanto llegan. En esencia, *R* miente sobre las transformaciones de dirección IP a dirección física.

nar los datagramas destinados a H_4 a través de la red oculta. a fin de permitir que los anfitriones en la red oculta alcancen anfitriones en la red principal, *R* también realiza el servicio de ARP sustituto (proxy ARP) en dicha red.

Los ruteadores que utilizan la técnica de ARP sustituto, toman ventaja de una característica importante del protocolo ARP, a saber, la confianza. ARP está basado en la idea de que todas las máquinas cooperan y de que cualquier respuesta es legítima. La mayor parte de los anfitriones instalan asociaciones obtenidas por medio de ARP sin verificar su validez y sin mantener una consistencia. Por lo tanto, puede suceder que la tabla ARP asocie muchas direcciones IP en la misma dirección física, sin embargo, esto no viola las especificaciones del protocolo.

Algunas implantaciones de ARP no son tan poco exigentes como otras. En particular, las implementaciones ARP diseñadas para alertar a los administradores de posibles violaciones de seguridad les informarán siempre que dos direcciones IP distintas se transformen en la misma dirección física de hardware. El propósito de alertar al administrador es avisarle sobre el *spoofing*, situación en la que una máquina indica ser otra para poder interceptar paquetes. Las implantaciones de ARP en anfitriones que alertan a los administradores del posible *spoofing* no se pueden utilizar en redes que tienen ruteadores sustitutos ARP, ya que el software generaría mensajes con gran frecuencia.

La principal ventaja de ARP sustituto es que se puede agregar a un solo ruteador en una red sin alterar las tablas de ruteo en otros anfitriones o ruteadores en esa red. Por lo tanto, el software ARP sustituto (proxy ARP) oculta completamente los detalles de las conexiones físicas.

La principal desventaja de ARP sustituto es que no trabaja para las redes a menos que utilicen ARP para la definición de direcciones. Además, no se generaliza para topologías de red más complejas (por ejemplo, muchos ruteadores que interconectan dos redes físicas), ni incorpora una forma razonable para el ruteo. De hecho, la mayor parte de las implantaciones de ARP confía en los administradores para el mantenimiento manual de máquinas y direcciones, haciendo que se ocupe tiempo y se tenga propensión a los errores.

10.6 Direccionamiento de subred

La tercera técnica utilizada para permitir que una sola dirección de red abarque muchas redes físicas se conoce como *direccionamiento de subred, ruteo de subred o utilización de subredes* (*subnetting*). Esta última técnica es la más empleada de las tres, ya que es la más general y la que se ha estandarizado. De hecho, el direccionamiento de subred es una parte obligatoria del direccionamiento IP.

La manera más sencilla de entender el direccionamiento de subred es imaginándose que una localidad tiene asignada una sola dirección de red IP tipo *B*, pero tiene dos o más redes físicas. Sólo los ruteadores locales saben que existen muchas redes físicas y cómo rutear el tráfico entre ellas; los ruteadores en otros sistemas autónomos rutean todo el tráfico como si sólo hubiera una red física. En la figura 10.3 se muestra un ejemplo.

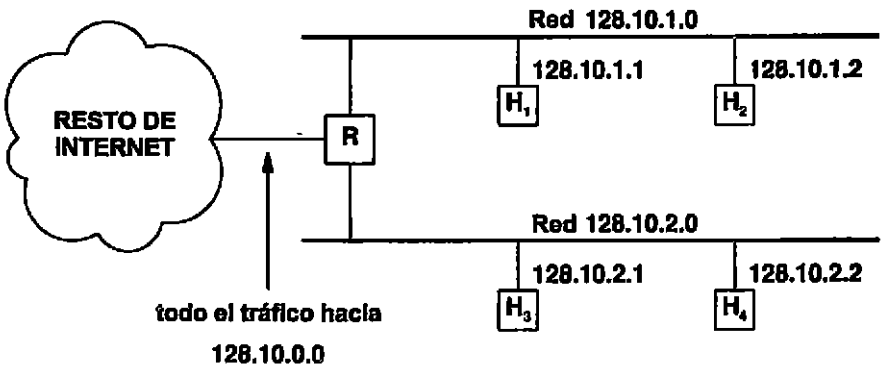


Figura 10.3 Localidad con dos redes físicas que utilizan el direccionamiento de subred para etiquetarlas con una sola dirección de red tipo *B*. El ruteador *R* acepta todo el tráfico para la red 128.10.0.0 y elige una red física, basándose en el tercer octeto de la dirección.

En el ejemplo, la localidad solamente utiliza la dirección de red tipo *B 128.10.0.0* para referirse a dos redes. Con excepción del ruteador *R*, todos los demás rutean como si fueran una sola red física. Una vez que un paquete llega a *R*, lo debe enviar a su destino a través de la red física correcta. Para hacer que la elección sea eficiente, el sitio local utiliza el tercer octeto de la dirección para distinguir entre las dos redes. El administrador asigna a las máquinas, en una red física, una dirección con la forma *128.10.1.X*, y a las máquinas en la otra red *128.10.2.X*, donde *X* representa un número entero pequeño, utilizado para identificar un anfitrión específico. Para escoger una red física, *R* examina el tercer octeto de la dirección de destino, rutea los datagramas que tengan el valor *1* hacia la red *128.10.1.0* y los que tengan el valor *2* hacia la red *128.10.2.0*.

Conceptualmente, agregar subredes sólo cambia ligeramente la interpretación de direcciones IP. En vez de dividir la dirección IP de 32 bits en un prefijo de red y un sufijo de anfitrión, el direccionamiento de subred divide la dirección en una *porción de red* y una *porción local*. La interpretación de la porción de red permanece igual que en las redes que no utilizan el direccionamiento de

subred. Como se dijo antes, la accesibilidad a la red se debe indicar a los sistemas autónomos del exterior; todo el tráfico que se destine para la red seguirá la ruta indicada. La interpretación de la porción local de una dirección se somete al criterio de la localidad (dentro de las limitaciones del estándar formal para el direccionamiento de subred). En resumen:

> *Pensamos que una dirección IP de 32 bits tiene una porción de red de redes y una porción local, en donde la porción de red identifica una localidad, posiblemente con muchas redes físicas, y la porción local identifica una red física y un anfitrión en dicha localidad.*

En el ejemplo de la figura 10.3, se mostró el direccionamiento de subred con una dirección tipo *B* que tenía una porción de red de redes de 2 octetos y una porción local de 2 octetos. En nuestro ejemplo, para lograr que el ruteo entre las redes físicas sea eficaz, el administrador de la localidad utilizó un octeto de la porción local a fin de identificar una red física y el otro octeto para identificar un anfitrión en dicha red, como se muestra en la figura 10.4.

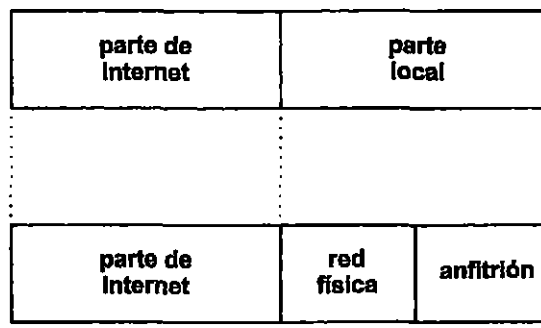


Figura 10.4 (a) Interpretación conceptual de una dirección IP de 32 bits siguiendo el esquema original de dirección IP, y (b), interpretación conceptual de direcciones que utilizan el esquema de subred mostrado en la figura 10.3. La porción local se divide en dos partes que identifican una red física y un anfitrión en dicha red.

El resultado es una forma de *direccionamiento jerárquico* que lleva al correspondiente *ruteo jerárquico*. El nivel superior del ruteo jerárquico (por ejemplo, otros sistemas autónomos en la red de redes), utiliza los primeros dos octetos cuando rutea y el siguiente nivel (por ejemplo, el sitio local) utiliza un octeto adicional. Finalmente, el nivel más bajo (por ejemplo, la entrega a través de una red física) utiliza toda la dirección.

El direccionamiento jerárquico no es nuevo; muchos sistemas lo han utilizado antes. El mejor ejemplo es el sistema telefónico de Estados Unidos, en donde un número telefónico de 10 dígitos se divide en un código de área de 3 dígitos, una serie de 3 dígitos y una conexión de 4 dígitos. La ventaja de utilizar el direccionamiento jerárquico es que puede incorporar un gran crecimiento, ya que significa que una ruta no necesita saber muchos detalles sobre destinos distantes, lo mismo

que sobre destinos locales. Una desventaja es que seleccionar una estructura jerárquica es difícil como, también, es difícil cambiar una jerarquía ya establecida.

10.7 Flexibilidad en la asignación de direcciones de subred

El estándar TCP/IP para el direccionamiento de subred reconoce que no todas las localidades tienen la misma necesidad de una jerarquía de direcciones; permite que tengan flexibilidad al poder escoger cómo asignarlas. Para entender por qué se necesita dicha flexibilidad, imagine una localidad con 5 redes interconectadas, como se muestra en la figura 10.5. Suponga que dicha localidad tiene una sola dirección de red tipo *B* que desea utilizar para todas las redes físicas. ¿Cómo se tiene que dividir la parte local para hacer que el ruteo sea eficiente?

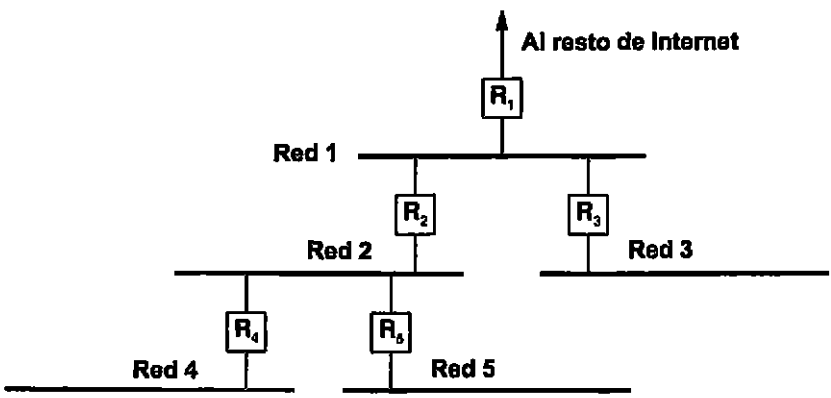


Figura 10.5 Localidad con cinco redes físicas dispuestas en tres "niveles." . La simple división de direcciones en partes de red física y de anfitrión puede no ser óptima en estos casos.

En el ejemplo, la localidad escogerá una partición de la parte local de la dirección IP, basándose en su futuro crecimiento. La división de la parte local de 16 bits, en un identificador de red de 8 bits y un identificador de anfitrión de 8 bits, como se muestra en la figura 10.4, permite hasta 256 redes, con hasta 256 anfitriones cada una.[1] La utilización de 3 bits para identificar una red física y 13 bits para identificar un anfitrión en dicha red, permite incluso 8 redes con hasta 8192 anfitriones cada una.

Ninguna partición de la parte local de la dirección trabajará por sí sola para todas las localidades, ya que algunas tienen muchas redes con unos cuantos anfitriones en cada una y otras tienen pocas redes con muchos anfitriones conectados a cada una. También es importante considerar que

[1] En la práctica, el límite es de 254 subredes con 254 anfitriones cada una, debido a que las direcciones de anfitrión de todos 1 y todos 0 están reservadas para la difusión, y no se recomiendan las subredes con todos 1 o todos 0.

se puede dar el caso de que, dentro de una localidad, algunas redes tengan muchos anfitriones y otras tengan pocos. Para permitir una máxima autonomía, el estándar TCP/IP de subred permite que la partición se seleccione basándose en cada red particular. Una vez que se escogió una partición para una red en particular, todos los anfitriones y ruteadores conectados a ella la deben utilizar. Si no lo hacen, los datagramas se pueden perder o rutear equivocadamente. Podemos resumir que:

> *Para permitir una máxima flexibilidad al particionar las direcciones de subred, el estándar TCP/IP de subred permite que la interpretación se escoja de forma independiente para cada red física. Una vez que se selecciona una partición de subred, todas las máquinas la deben utilizar.*

10.8 Implantaciones de subredes con máscaras

Hemos dicho que escoger un esquema de direccionamiento de subred es lo mismo que escoger cómo dividir la porción local de una dirección IP en dos partes, red física y anfitrión. De hecho, la mayor parte de las localidades que utilizan las direcciones de subred lo hacen, pero el direccionamiento de subred también permite asignaciones más complejas. El estándar especifica que una localidad que utiliza el direccionamiento de subred, debe escoger una *máscara de subred* de 32 bits para cada red. Los bits en la máscara de subred se indican como *1*, si la red trata al bit correspondiente de la dirección IP como parte de la dirección de red, y se indican como *0*, si se trata al bit como parte del identificador de anfitrión. Por ejemplo, la máscara de subred de 32 bits:

11111111 11111111 11111111 00000000

especifica que los tres primeros octetos identifican a la red y el cuarto a un anfitrión en dicha red. Una máscara de subred debe tener *1* para todos los bits que correspondan a la porción de red de la dirección (por ejemplo, la máscara de subred para una red tipo *B* tendrá *1* en los primeros dos octetos y adicionalmente uno o más bits en los dos últimos).

Este giro interesante en el direccionamiento de subred surge porque el estándar no restringe a las máscaras de subred para que seleccionen bits contiguos de la dirección. Por ejemplo, una red puede tener asignada la máscara:

11111111 11111111 00011000 01000000

la cual selecciona los primeros dos octetos, dos bits del tercer octeto y un bit del cuarto. Aunque tal flexibilidad hace posible que se puedan realizar asignaciones interesantes de direcciones, también causa que la asignación de direcciones de anfitrión y que el entendimiento de las tablas de ruteo sean un poco confusos. Por lo tanto, se recomienda que las localidades utilicen máscaras contiguas de subred y empleen la misma máscara a lo largo de todo un grupo de redes físicas que compartan una sola dirección IP.

10.9 Representación de máscaras de subred

Especificar máscaras de subred de forma binaria es molesto y favorece los errores. Por lo tanto, la mayor parte del software permite representaciones alternativas. Algunas veces, la representación sigue cualquier norma que el sistema operativo utilice para la representación de cantidades binarias (por ejemplo, notación hexadecimal).

La representación decimal con puntos también es popular para las máscaras de subred; funciona mejor cuando las localidades alinean el direccionamiento de subred en grupos de octetos. Por ejemplo, muchas localidades asignan direcciones tipo *B* para subred al utilizar el tercer octeto a fin de identificar la red física y el cuarto para identificar a los anfitriones, como se indica en la página anterior. En dichos casos, la máscara de subred tiene una representación decimal con puntos *255.255.255.0*, lo que facilita su escritura y comprensión.

El texto también contiene ejemplos de direcciones y máscaras de subred representadas por tres partes entre corchetes:

{ <número de red>, <número de subred>, <número de anfitrión> }

En esta representación, el valor *-1* representa "todos unos". Por ejemplo, si la máscara de subred para una red tipo *B* es 255.255.255.0, se puede escribir *{-1, -1, 0}*.

La principal desventaja de la representación de tres partes es que no especifica con precisión cuántos bits se utilizan para cada parte de la dirección; la ventaja es que no entra en detalles sobre los campos de bits y que enfatiza los valores de las tres partes de la dirección. Para ver por qué, algunas veces, los valores son más importantes que los campos de bits, considere este conjunto de tres partes:

{ 128.10, -1, 0 }

que denota una dirección con un número de red *128.10*, todos unos en el campo de subred, y todos ceros en el campo de anfitrión. Expresar el mismo valor de dirección, utilizando otra representación, requiere una dirección IP de 32 bits y una máscara de subred de 32 bits, lo que obliga a los lectores a decodificar los campos de bits antes de que puedan deducir los valores de los campos individuales. Además, la representación de tres partes es independiente del tipo de dirección IP, así como del tamaño del campo de subred. Por lo tanto, se puede utilizar para representar grupos de direcciones o ideas abstractas. Por ejemplo, el conjunto de tres partes:

{ <número de red>, -1, -1 }

denota "direcciones con un número válido de red, un campo de subred que contiene sólo unos y un campo de anfitrión que contiene solo unos". Más adelante en este capítulo veremos más ejemplos.

10.10 Ruteo con la presencia de subredes

Se debe modificar el algoritmo estándar de ruteo IP para trabajar con direcciones de subred. Todos los anfitriones y ruteadores conectados a una red que utilice el direccionamiento de subred deben emplear dicho algoritmo modificado, al cual se le conoce como *ruteo de subred*. Lo que puede no ser obvio es que, a menos que se agreguen restricciones para utilizar el direccionamiento de subred, otros anfitriones y ruteadores en la localidad también necesiten utilizar el ruteo de subred. Para ver por qué, considere el ejemplo de un grupo de redes que se muestra en la figura 10.6.

En la figura, las redes físicas *2* y *3* tienen asignadas direcciones de subred desde una sola dirección IP de red, *N*. Aunque el anfitrión *H* no se conecta de manera directa a una red que tenga una dirección de subred, debe utilizar el ruteo de subred para decidir a dónde enviar los datagramas destinados para la red *N* si al ruteador R_1 o al ruteador R_2. Se puede argüir que *H* puede enviarlos a cualquier ruteador y dejar que ellos resuelvan el problema, pero esta solución significa que no todo el tráfico seguirá el camino más corto. En rutas más largas, la diferencia entre un camino óptimo y uno que no lo es puede ser significativa.

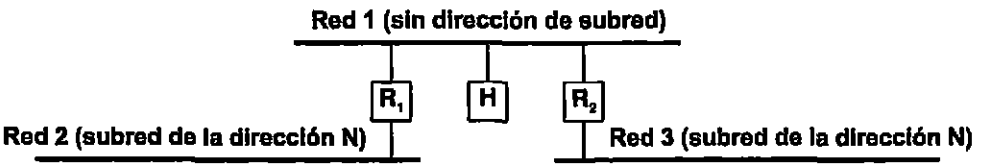


Figura 10.6 Topología de ejemplo (ilegal) con tres redes, donde las redes *2* y *3* son subredes de una sola dirección de red IP, *N*. Si se permitieran topologías como ésta, el anfitrión *H* necesitaría utilizar el ruteo de subred aun cuando la Red *1* no tenga una dirección de subred.

En teoría, una ruta simple determina cuándo las máquinas necesitan utilizar el ruteo de subred. La regla de subred es que:

> *Para lograr un ruteo óptimo, una máquina* M *debe utilizar el ruteo de subred para una dirección IP de red* N, *a menos que exista un solo camino,* P, *que sea el más corto entre* M *y cualquier red física que sea subred de* N.

Desafortunadamente, entender la restricción teórica no ayuda a la asignación de subredes. Primero, los caminos más cortos pueden cambiar si el hardware falla o si los algoritmos de ruteo redireccionan el tráfico alrededor de un congestionamiento. Tales cambios dinámicos dificultan el uso de la regla de subred, a excepción de algunos casos insignificantes. Segundo, la regla de subred no considera las fronteras entre localidades, ni las dificultades comprendidas en la propagación de máscaras de subred. Es imposible propagar rutas de subred más allá de la frontera de una organización, ya que los protocolos de ruteo que trataremos más adelante no lo permiten. Objetivamente, es en extremo difícil propagar información de subred más allá de cierta red física. Por lo tanto, los diseñadores recomiendan que, si una localidad utiliza el direccionamiento de subred, las subredes se

deben mantener tan simples como sea posible. En particular, los administradores de red deben apegarse a los siguientes lineamientos:

> *Todas las subredes en una dirección IP de red deben ser contiguas, las máscaras de subred deben ser uniformes a través de todas las redes y todas las máquinas deben participar en el ruteo de subred.*

Los lineamientos implican una dificultad especial para una gran corporación que tiene muchas localidades, cada una conectada con Internet, pero no conectadas directamente una con otra. Dicha corporación no puede utilizar subredes de una sola dirección para todas sus localidades, ya que las redes físicas no son contiguas.

10.11 El algoritmo de ruteo de subred

Al igual que el algoritmo estándar de ruteo IP, el algoritmo de ruteo en subredes basa sus decisiones en una tabla de rutas. Recuerde que en el algoritmo estándar, las rutas por anfitrión y las rutas asignadas por omisión son casos especiales que se deben verificar de manera explícita; para los demás casos se lleva a cabo la búsqueda en tablas. Una tabla convencional de ruteo contiene registros que tienen la siguiente forma:

(dirección de red, dirección de salto siguiente)

donde el campo de *dirección de red* especifica la dirección IP de la red de destino, *N*, y el campo de *dirección de salto siguiente* especifica la dirección de un ruteador al que se deben enviar los datagramas destinados para *N*. El algoritmo estándar de ruteo compara la porción de red de una dirección destino con el campo de *dirección de red* de cada registro en la tabla de ruteo, hasta que encuentra una correspondencia. Debido a que el campo de *dirección de salto siguiente* está reservado sólo para especificar una máquina que se puede accesar a través de una red conectada de manera directa, solamente es necesaria una búsqueda en tabla.

El algoritmo estándar sabe que una dirección está dividida en una porción de red y una porción local, ya que los primeros tres bits codifican el tipo y formato de la dirección (por ejemplo, los tipos *A, B, C* o *D*). Con las subredes, no es posible decidir qué bits corresponden a la red ni cuáles corresponden al anfitrión sólo con la dirección. En cambio, el algoritmo modificado que se utiliza con las subredes guarda información adicional en la tabla de ruteo. Cada registro dentro de la tabla contiene un campo adicional que especifica la máscara de subred utilizada con la red:

(máscara de subred, dirección de red, dirección de salto siguiente)

Cuando el algoritmo modificado elige rutas, utiliza la *máscara de subred* para extraer bits de la dirección de destino y compararlos con el registro en la tabla. Esto es, realiza una operación booleana inteligente *y*, con los 32 bits de la dirección IP de destino así como con el campo de *máscara de subred* de un registro; luego, verifica si el resultado es igual al valor del campo de *dirección de red*

en ese registro. Si así es, rutea el datagrama a la dirección especificada en el campo de *dirección de salto siguiente*[2] del registro.

10.12 Un algoritmo unificado de ruteo

Algunos lectores habrán adivinado que, si permitimos máscaras arbitrarias, el algoritmo de ruteo de subred puede manejar todos los casos especiales del algoritmo estándar. Puede manejar rutas hacia anfitriones individuales, rutas asignadas por omisión y rutas a redes conectadas directamente,

```
                    Algoritmo:

          Ruta_IP_Datagrama (datagrama, tabla_ruteo)

Extraer la dirección IP de destino, ID, del datagrama:
Computar la dirección IP de la red de destino, IN:
Si IN corresponde a cualquier dirección de red conectada enviar
   el datagrama a su destino a través de dicha red (Esto
   involucra la transformación de ID en una dirección física,
   encapsular el datagrama y enviar la trama.)
De otra forma
   para cada registro en la tabla de ruteo hacer lo siguiente
     Dejar que N sea el bitwise-and de ID y de la máscara
        de subred
     Si N es igual al campo de dirección de red del
        registro, entonces rutear el datagrama a la dirección
        especificada de salto siguiente
  fin-de-ciclo
Si no se encuentran correspondencias, declarar un error de
   ruteo;
```

Figura 10.7 Algoritmo unificado de ruteo IP. Con un datagrama IP y una tabla de ruteo con máscaras, este algoritmo selecciona un ruteador de salto siguiente al que debe enviarse el datagrama. El salto siguiente debe residir en una red conectada de manera directa.

[2] Al igual que en el algoritmo estándar de ruteo, el ruteador de salto al siguiente debe ser accesible para una red conectada directamente.

utilizando la misma técnica de enmascaramiento que utiliza para las subredes. Además, las máscaras pueden manejar rutas hacia redes convencionales (por ejemplo, redes que no emplean el direccionamiento de subred). La flexibilidad surge de la capacidad para combinar valores arbitrarios de 32 bits en un campo de *máscara de subred,* con direcciones arbitrarias de 32 bits en un campo de *dirección de red.* Por ejemplo, para instaurar una ruta para un solo anfitrión, se utiliza una máscara con todos *1* y con la dirección de red igual a la dirección IP del anfitrión. Para instaurar una ruta asignada por omisión, se utiliza una máscara de subred con todos *0* y una dirección de red con todos *0* (debido a que cualquier dirección de destino *más* cero es igual a cero). Para instaurar una ruta hacia una red tipo *B*, estándar y no subred, se especifica una máscara con dos octetos de *1* y dos octetos de *0*. Debido a que la tabla contiene más información, el algoritmo de ruteo contiene menos casos especiales, como se muestra en la figura 10.7.

De hecho, las implementaciones realizadas de manera inteligente pueden eliminar la prueba explícita de destinos en las redes conectadas directamente, al agregar registros en la tabla con valores apropiados para la máscara y la dirección de red.

10.13 Mantenimiento de las máscaras de subred

¿Cómo se asignan y propagan las máscaras de subred? En el capítulo 9 se contestó la segunda parte de la pregunta al mostrar que un anfitrión puede obtener la máscara de subred para una red al enviar una *solicitud de máscara de subred* ICMP al ruteador en dicha red. La solicitud se puede transmitir por difusión si el anfitrión no conoce la dirección específica de un ruteador. Sin embargo, no existe un protocolo estándar para propagar la información de un ruteador a otro.

La primera parte de la pregunta es más difícil de responder. Cada localidad tiene la libertad de escoger máscaras de subred para sus redes. Cuando hacen asignaciones, los administradores intentan balancear los tamaños de las redes, los números de redes físicas, el crecimiento esperado y el mantenimiento. La dificultad surge porque las máscaras no uniformes proporcionan una flexibilidad máxima, pero posibilitan hacer asignaciones que llevan a rutas ambiguas. O peor aún, permiten que las asignaciones válidas se vuelvan no válidas si se agregan más anfitriones a las redes. No existen reglas fáciles, por lo que la mayor parte de las localidades hacen selecciones conservadoras. Por lo general, para identificar una red, una localidad selecciona bits contiguos de la porción local de una dirección y utiliza la misma división (por ejemplo, la misma máscara) para todas las redes físicas. Por ejemplo, muchas localidades utilizan un solo octeto de subred cuando manejan una dirección tipo *B*.

10.14 Difusión a las subredes

La difusión es más difícil en una arquitectura de subred. Recuerde que, en el esquema original de direccionamiento IP, una dirección con una porción de todos *1* denota difusión a todos los anfitriones en la red especificada. Desde el punto de vista de un observador externo, de una localidad de subredes, la difusión hacia la dirección de red todavía tiene sentido. Esto es, la dirección:

{ red, –1, –1 }

significa "entregar una copia a todas las máquinas que tienen *red* como su dirección de red, inclusive si residen en redes físicas separadas". Operacionalmente, la difusión hacia una dirección así sólo tiene sentido si los ruteadores que interconectan las subredes están de acuerdo en propagar el datagrama hacia todas las redes físicas. Claro está, se debe tener cuidado para no rutear en ciclos. En particular, un ruteador no puede sólo propagar un paquete de difusión que llega a su interfaz, hacia todas las interfaces que comparten el prefijo de subred. Para prevenir dichos ciclos, los ruteadores utilizan el *direccionamiento de camino reversible* (*reverse path forwarding*). El ruteador extrae, del datagrama, el origen de la difusión y busca la fuente en su tabla de ruteo. Luego, descarta el datagrama, a menos que haya llegado por la interfaz utilizada para rutear hacia el origen (por ejemplo, si llegó por el camino más corto).

Dentro de un grupo de redes con subredes, es posible transmitir por difusión hacia una subred específica (por ejemplo, transmitir por difusión hacia todos los anfitriones en una red física que tienen asignada una de las direcciones de subred). El estándar de direcciones de subred se vale de un campo de anfitrión de todos *1* para denotar la difusión de subred. Ahora bien, una dirección de difusión de subred es:

{ red, subred, -1 }

La consideración de las direcciones de difusión de subred, así como la difusión de subred, aclara la recomendación para utilizar una máscara consistente de subred a través de todas las redes que comparten una dirección IP de subred. Mientras los campos de subred y de anfitrión sean idénticos, las direcciones de difusión de subred no serán ambiguas. Las asignaciones más complejas de dirección de subred pueden o no permitir la difusión a subgrupos seleccionados de las redes físicas que las comprenden.

10.15 Direccionamiento de superred

El direccionamiento de superred se desarrolló a principios de los años ochenta para ayudar a conservar el espacio de las direcciones IP. Para 1993, parecía que el direccionamiento de subred no evitaría que el crecimiento de Internet finalmente acabara con el espacio para direcciones tipo B. Se comenzó a trabajar para definir una versión totalmente nueva de IP con direcciones más grandes. Sin embargo, para incorporar el crecimiento hasta que se estandarice y adapte la nueva versión de IP, se encontró una solución temporal.

El esquema, llamado *direccionamiento de superred*, tiene un enfoque opuesto al del direccionamiento de subred. En vez de utilizar una sola dirección IP de red para muchas redes físicas en una organización, el direccionamiento de superred permite la utilización de muchas direcciones IP de red para una sola organización. Para entender por qué se adoptó este esquema, se necesitan saber tres cosas. Primero, el IP no divide las direcciones de red en tipos iguales. Aunque sólo se pueden asignar menos de 17 mil números tipo B, existen más de 2 millones de números de red tipo C. Segundo, los números tipo C se solicitaban lentamente; sólo un pequeño porcentaje se había asignado. Tercero, los estudios demuestran que a la velocidad en que se asignaban números tipo B, és-

tos se acabarían en unos cuantos años. El problema se conocía como *Terminación del Espacio para Direcciones (ROADS,* por sus siglas en inglés *Running Out of Address Space).*

Para entender cómo funciona el direccionamiento de superred, considere una organización mediana que se une a Internet. Ésta preferiría utilizar una sola dirección tipo B por dos razones: una dirección tipo C no puede incorporar más de 254 anfitriones y una dirección tipo B tiene suficientes bits para que el direccionamiento de superred sea conveniente. Para conservar los números tipo B, el esquema de direccionamiento de superred asigna a la organización un grupo de direcciones tipo C en vez de un solo número tipo B. El grupo debe ser lo suficientemente grande para numerar todas las redes que eventualmente conectará a Internet. Por ejemplo, suponga que una organización solicita una dirección tipo B, que piensa direccionar por subred utilizando el tercer octeto como campo de subred. En vez de asignar un solo número tipo B, el direccionamiento de superred asigna a la organización un grupo de 256 números tipo C, para que ésta los asigne a las redes físicas.

Aunque el direccionamiento de superred es fácil de entender cuando se ve desde el punto de vista de una sola localidad, los que lo proponen piensan que se debe utilizar en un contexto más amplio. Ellos inventaron una Internet jerárquica en la que los *Proveedores de Servicios de Red* proporcionan conectividad a Internet. Para conectar sus redes con Internet, una organización contrataría los servicios de un Proveedor de Conexión; éste maneja los detalles de la asignación de direcciones IP, así como la instalación de conexiones físicas. Los diseñadores del direccionamiento de superred proponen que se permita que los Proveedores de Servicios de Red obtengan gran parte del espacio para direcciones (por ejemplo, un grupo de direcciones que abarque muchos números de redes tipo C). Entonces, el Proveedor de Servicios de Red podrá proporcionar una o más direcciones a cada uno de sus suscriptores.

10.16 El efecto de trabajar con superredes en el ruteo

Asignar muchas direcciones tipo C en vez de una sola tipo B conserva los números tipo B y resuelve el problema inmediato de la terminación de espacio para direcciones. Sin embargo, crea un nuevo problema: la información que los ruteadores almacenan e intercambian aumenta dramáticamente. En particular, una tabla de ruteo, en vez de tener un registro por cada organización, contiene muchos registros para cada una.

Una técnica conocida como *Ruteo sin tipo de inter-dominio (CIDR, Classless Inter-Domain Routing)*[3] resuelve el problema. Conceptualmente, la CIDR colapsa un grupo de direcciones contiguas tipo C en un solo registro representado por dos datos:

(dirección de red, conteo)

en donde la *dirección de red* es la dirección de la red más pequeña del grupo y *conteo* especifica el número total de direcciones en el grupo. Por ejemplo, el par de datos:

(192.5.48.0 , 3)

se puede utilizar para especificar las tres direcciones de red 192.5.48.0, 192.5.49.0, y 192.5.50.0.

[3] El nombre es ligeramente incorrecto ya que el esquema especifica el direccionamiento así como el ruteo.

Si unos cuantos proveedores de servicio forman el núcleo de Internet y cada uno es dueño de un gran grupo de números contiguos de red IP, el beneficio del direccionamiento de superred es evidente. Considere registros de una tabla de ruteo del proveedor de servicio *P*. Por supuesto, la tabla debe tener una ruta correcta hacia cada suscriptor de *P*. La tabla no necesita contener un registro para cada uno de los demás proveedores. El registro identifica el grupo de direcciones del proveedor.

En la práctica, la CIDR no restringe los números de red sólo a direcciones tipo C, ni utiliza un conteo de números enteros para especificar el tamaño de un grupo. Por el contrario, la CIDR requiere que cada grupo de direcciones sea una potencia de dos y utiliza una máscara de bit para identificar el tamaño del grupo. Por ejemplo, suponga que una organización tiene asignado un grupo de 2048 direcciones contiguas, comenzando en la dirección 234.170.168.0. En la tabla de la figura 10.8 se muestran los valores binarios de las direcciones en dicho rango.

	Notación decimal con puntos	Equivalencia binaria de 32 bits
más baja	234.170.168.0	11101010 10101010 10101000 00000000
más alta	234.170.175.255	11101010 10101010 10101111 11111111

Figura 10.8 Grupo de 2048 direcciones. La tabla muestra la dirección más alta y la más baja del rango, expresadas en notación decimal con puntos y con valores binarios.

En la figura 10.8, la CIDR requiere que dos valores especifiquen el rango: la dirección más baja y una máscara de 32 bits. La máscara opera como una máscara estándar de subred al delimitar el fin del prefijo. Para el rango mostrado, la máscara CIDR tiene el grupo de 21 bits:[4]

11111111 11111111 11111000 00000000

Para hacer uso de todas las direcciones posibles de anfitrión en un rango, los ruteadores en una localidad que utilizan direccionamiento sin tipo se deben cambiar. Cuando el software de ruteo busca una ruta, no interpreta el tipo de dirección de destino. En vez de eso, cada registro en la tabla de ruteo contiene una dirección y una máscara, y el software de ruteo utiliza un paradigma de *correspondencia mayor* para seleccionar la ruta. Por lo tanto, un grupo de direcciones se puede subdividir y pueden introducirse rutas separadas para cada subdivisión. Como resultado, aunque el grupo de computadoras en una red tendrá asignadas direcciones en un rango fijo, éste no necesita corresponder a un valor binario.

El direccionamiento de superred trata las direcciones IP como números enteros arbitrarios y permite que un administrador de red asigne un grupo de números contiguos a una localidad o a una red dentro de una localidad. Los anfitriones y ruteadores que utilizan el direccionamiento de superred necesitan software de ruteo no convencional que entienda los rangos de direcciones.

---

[4] En notación decimal con puntos, el valor de la máscara es *255.255.248.0*.

10.17 Resumen

En este capítulo, se examinaron cuatro técnicas que extienden el esquema de direccionamiento IP. El esquema original asigna una dirección única de 32 bits a cada red física y requiere de una tabla de ruteo IP, proporcional al número de redes en la red. Tres de las técnicas que examinamos se diseñaron para conservar direcciones: permiten que una localidad comparta una dirección de red de redes entre muchas redes físicas. La primera, utiliza ruteadores transparentes para extender el espacio de direcciones de una sola red, por lo general una WAN, para poder incluir anfitriones en una red local conectada. La segunda, llamada ARP sustituto, permite que un ruteador local personifique computadoras en otra red física al contestar mensajes ARP direccionados a ellas. ARP sustituto es útil en las redes que utilicen ARP para la definición de direcciones y sólo para aplicaciones ARP que no tengan problemas cuando muchas direcciones de red de redes se transformen en la misma dirección de hardware. La tercera técnica, un estándar de TCP/IP llamado direccionamiento de subred, permite que una localidad comparta una sola dirección de red IP entre muchas redes físicas, siempre y cuando cooperen todos los anfitriones y ruteadores en dichas redes. El direccionamiento de subred requiere que los anfitriones utilicen un algoritmo modificado de ruteo, en el que los registros de la tabla de ruteo contengan una máscara de subred. El algoritmo se puede ver como una generalización del algoritmo de ruteo original, ya que maneja casos especiales, como rutas asignadas por omisión o rutas de anfitrión específico.

Por último, examinamos la técnica de direccionamiento de superred, en la que una localidad tiene asignado un grupo de varios números tipo C. También conocido como direccionamiento sin tipo, este esquema requiere hacer cambios en el software de ruteo de los anfitriones y los ruteadores.

PARA CONOCER MÁS

El estándar para el direccionamiento de subred se deriva de Mogul (RFC 950), con actualizaciones en Braden (RFC 1122), Clark (RFC 932), Karels (RFC 936), Gads (RFC 940) y Mogul (RFC 917); todos contienen propuestas iniciales para los esquemas de direccionamiento de subred. Mogul (RFC 922) analiza la difusión en presencia de subredes. Postel (RFC 925) considera la utilización de ARP sustituto para subredes. Carl-Mitchell y Quaterman (RFC 1027) tratan la utilización de ARP sustituto para implantar ruteadores transparentes de subred. Fuller, Li, Yu y Varadhan (RFC 1519) especifican las extensiones de direcciones de superred y el ruteo sin tipo inter-dominio.

EJERCICIOS

10.1 Si los ruteadores que utilizan el software ARP sustituto (proxy ARP) usan una tabla de direcciones de anfitriones para decidir si responden o no a las solicitudes ARP, la tabla del ruteador debe cambiarse siempre que se agregue un nuevo ruteador a una de las redes. Explique cómo asignar direcciones IP para que se puedan agregar anfitriones sin cambiar las tablas. Pista: piense en las subredes.

10.2 Aunque el estándar permite asignar sólo 0 como un número de subred, el software de algunos fabricantes no opera correctamente. Trate de asignar una subred cero a su localidad y vea si la ruta se difunde correctamente.

10.3 ¿Se pueden utilizar ruteadores transparentes con redes de área local como Ethernet? ¿Por qué?

10.4 Demuestre que el ARP sustituto se puede utilizar con tres redes físicas interconectadas por medio de dos ruteadores.

10.5 Considere la partición fija de subred de un número de red tipo *B*, que incorporará cuando menos 76 redes. ¿Cuántos anfitriones puede haber en cada red?

10.6 ¿Tiene algún sentido direccionar por subred una dirección de red tipo *C*? ¿Por qué?

10.7 Una persona en una localidad en la que se eligió direccionar por subred su dirección tipo *B*, al utilizar el tercer octeto para la red física, se desilusionó al saber que no podría incorporar 255 o 256 redes. Explique por qué.

10.8 Diseñe un esquema de direccionamiento de subred para su organización, asumiendo que se utilizará una dirección tipo *B*.

10.9 ¿Es razonable que un solo ruteador utilice tanto ARP sustituto como direccionamiento de subred? Si así es, explique cómo. Si no, explique por qué.

10.10 Demuestre que cualquier red que utilice ARP sustituto es vulnerable al "spoofing" (por ejemplo, una máquina cualquiera puede personificar a cualquier otra).

10.11 ¿Puede imaginar una implantación (no estándar) que implique un uso normal, pero que prohíba la utilización de ARP sustituto?

10.12 Un fabricante decidió agregar el direccionamiento de subred a su software IP, al asignar una máscara de subred utilizada para todas las direcciones IP de red. El fabricante modificó el software estándar de ruteo IP para que la subred verificara un caso especial. Encuentre un ejemplo simple en el que dicha implantación no trabajaría correctamente. Pista: piense en un anfitrión multi-homed.

10.13 Caracterice las situaciones (restringidas) en las que la implantación de subred tratada en el ejercicio anterior trabajaría correctamente.

10.14 Lea el estándar para conocer más sobre la difusión en presencia de subredes. ¿Puede caracterizar asignaciones de direcciones de subred que permitan especificar una dirección de difusión para todas las subredes posibles?

10.15 El estándar permite la asignación arbitraria de máscaras de subred para redes que contengan una dirección IP de subred. ¿El estándar debería restringir las máscaras de subred a sólo cubrir bits contiguos en la dirección? ¿Por qué?

10.16 Considere con cuidado el ruteo asignado por omisión en presencia de subredes. ¿Qué puede suceder si llega un paquete destinado a una subred no existente?

10.17 Compare las arquitecturas que utilizan direccionamiento de subred y ruteadores para interconectar muchas Ethernets con la arquitectura que utiliza puentes como se describió en el capítulo 2. ¿Bajo qué circunstancias se preferiría una arquitectura más que la otra?

10.18 Considere una localidad que elige direccionar por subred una dirección de red tipo *B*, pero decide que algunas redes físicas utilizarán *6* bits de la porción local para identificar la red física, mientras que otras utilizarán *8*. Encuentre una asignación de direcciones de anfitrión que haga que las direcciones de destino sean ambiguas.

10.19 El algoritmo de ruteo de subred mostrado en la figura 10.7 utiliza un rastreo secuencial de registros en la tabla de ruteo, permitiendo que un administrador coloque rutas de anfitrión específico antes que rutas de red específica o de subred específica. Diseñe una estructura de datos que tenga la misma flexibilidad pero que utilice la comprobación aleatoria para hacer eficiente la búsqueda. (Este ejercicio lo sugirió Dave Mills.)

10.20 Si todos los proveedores de servicio Internet utilizaran subredes y asignaran números de suscriptor desde su grupo de direcciones, ¿qué problema ocurriría cuando un suscriptor cambiara de proveedor?

11

Estratificación de protocolos por capas

11.1 Introducción

En capítulos anteriores, revisamos los fundamentos de la arquitectura del enlace de redes, describimos cómo los anfitriones y ruteadores transmiten datagramas en Internet y presentamos los mecanismos utilizados para asociar direcciones IP a direcciones de la red física. En este capítulo, se considera la estructura de los fundamentos del software en los anfitriones y ruteadores que hacen posible las comunicaciones en red. Se presentan los principios generales de la estratificación por capas, se muestra cómo la estratificación por capas hace que el software de Protocolo de Internet sea fácil de entender y construir, y se sigue la ruta que los datagramas encuentran al pasar por una red de redes TCP/IP, a través del software de protocolo.

11.2 Necesidad de manejar varios protocolos

Hemos dicho que los protocolos permiten especificar o entender una forma de comunicación sin conocer los detalles del hardware de red de un vendedor en particular. Éstos son para las comunicaciones entre computadoras, lo que los lenguajes de programación para la computación. En este punto, debe ser claro porque esta analogía es válida. Como en el lenguaje ensamblador, algunos protocolos describen la comunicación a través de una red física. Por ejemplo, los detalles del formato de trama de la red Ethernet, las políticas de acceso a la red y el manejo de los errores de trama, se incluyen en un protocolo que describe la comunicación en una red Ethernet. De la misma

forma, los detalles de las direcciones IP, el formato de los datagramas, y el concepto de entrega no confiable y sin conexión, se incluyen en el Protocolo Internet.

Los sistemas complejos de comunicación de datos no utilizan un solo protocolo para manejar todas las tareas de transmisión, sino que requieren de un conjunto de protocolos cooperativos, a veces llamados *familia de protocolos* o *conjunto de protocolos*. Para entender por qué, piense en los problemas que pueden presentarse cuando las máquinas se comunican a través de una red de datos:

- *Fallas en el Hardware.* Un anfitrión o un ruteador puede fallar, ya sea porque el hardware falle o porque el sistema operativo quede fuera de servicio. Un enlace de transmisión de red puede fallar o desconectarse accidentalmente. El software de protocolo necesita detectar estas fallas y restablecer el funcionamiento.
- *Congestionamiento en la red.* Aun cuando el hardware y el software funcionen correctamente, éstos tienen una capacidad finita que puede ser excedida. El software de protocolo debe implantar un arreglo en las vías de transmisión para que una máquina congestionada no entorpezca el tráfico.
- *Paquetes retrasados o perdidos.* Algunas veces, el envío de paquetes tiene retrasos muy largos o éstos se pierden. El software de protocolo necesita aprender acerca de las fallas o debe adaptarse a los retardos.
- *Corrupción de datos.* La interferencia eléctrica, magnética o las fallas en el hardware pueden ocasionar errores de transmisión que alteran el contenido de los datos transmitidos. El software de protocolo necesita detectar y reparar estos errores.
- *Errores en la secuencia de los datos o duplicación de datos.* Las redes que ofrecen múltiples rutas pueden entregar los datos fuera de secuencia o entregar paquetes duplicados. El software de protocolo necesita reordenar los paquetes y suprimir los duplicados.

Si se consideran en conjunto, todos estos problemas parecen abrumadores. Es difícil entender cómo se podría escribir un solo protocolo para manejar todos estos problemas. A partir de una analogía con los lenguajes de programación podremos ver cómo superar la complejidad de este problema. Consideremos la subdivisión en cuatro subproblemas conceptuales de las transformaciones realizadas por un programa, identificando cada una de las partes con el software que maneja cada subproblema: compilador, ensamblador, editor de enlace y cargador. Esta división hace posible que el diseñador se concentre en un subproblema por vez, y también hace posible que el desarrollador construya y pruebe cada parte del software de manera independiente. Veremos cómo el software de protocolo se divide en forma similar.

Dos observaciones finales acerca de nuestra analogía con los lenguajes de programación ayudarán a aclarar la organización de protocolos. En primer lugar, debe quedar claro que las partes del software deben mantener un acuerdo sobre el formato exacto de los datos que pasan entre ellas. Por ejemplo, los datos que pasan del compilador hacia el ensamblador provienen de un programa definido por el lenguaje de programación ensamblador. Así, puede verse cómo el proceso de transferencia comprende varios lenguajes de programación. La analogía se sostiene para el caso del software de comunicación, en el cual se puede ver que varios protocolos definen la interfaz entre los módulos de software de comunicación. En segundo lugar, las cuatro partes de la transferencia siguen una secuencia lineal, en la que la salida del compilador es la entrada del ensamblador y así sucesivamente. El software de protocolo también utiliza una secuencia lineal.

11.3 Las capas conceptuales del software de protocolo

Pensemos los módulos del software de protocolo en una máquina como una pila vertical constituida por *capas*, como se muestra en la figura 11.1. Cada capa tiene la responsabilidad de manejar una parte del problema.

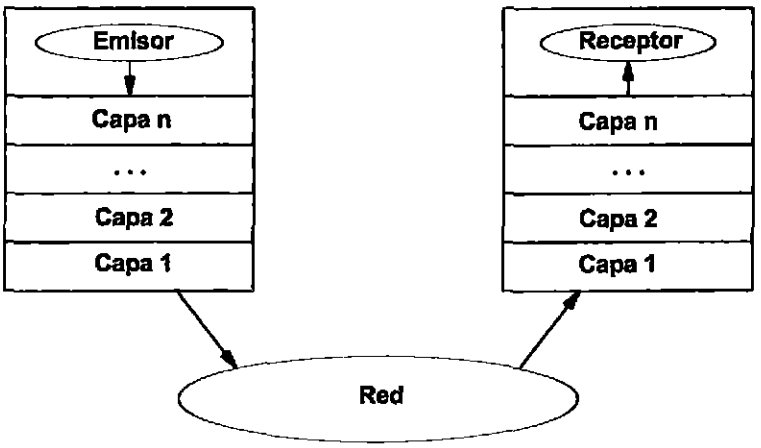


Figura 11.1 Organización conceptual del software de protocolo en capas.

Conceptualmente, enviar un mensaje desde un programa de aplicación en una máquina hacia un programa de aplicación en otra, significa transferir el mensaje hacia abajo, por las capas sucesivas del software de protocolo en la máquina emisora, transferir el mensaje a través de la red y, luego, transferir el mensaje hacia arriba, a través de las capas sucesivas del software de protocolo en la máquina receptora.

En la práctica, el software de protocolo es mucho más complejo de lo que se muestra en el modelo simplificado de la figura 11.1. Cada capa toma decisiones acerca de lo correcto del mensaje y selecciona una acción apropiada con base en el tipo de mensaje o la dirección de destino. Por ejemplo, una capa en la máquina de recepción debe decidir cuándo tomar un mensaje o enviarlo a otra máquina. Otra capa debe decidir qué programa de aplicación deberá recibir el mensaje.

Para entender la diferencia entre la organización conceptual del software de protocolo y los detalles de implantación, consideremos la comparación que se muestra en la figura 11.2. El diagrama conceptual en la figura 11.2a muestra una capa de Internet entre una capa de protocolo de alto nivel y una capa de interfaz de red. El diagrama realista de la figura 11.2b muestra el hecho de que el software IP puede comunicarse con varios módulos de protocolo de alto nivel y con varias interfaces de red.

Aun cuando un diagrama conceptual de la estratificación por capas no muestra todos los detalles, sirve como ayuda para explicar los conceptos generales. Por ejemplo, la figura 11.3 muestra las capas del software de protocolo utilizadas por un mensaje que atraviesa tres redes. El diagrama

muestra sólo la interfaz de red y las capas del Protocolo Internet en los ruteadores debido a que sólo estas capas son necesarias para recibir, rutear y enviar los datagramas. Se entiende que cualquier máquina conectada hacia dos redes debe tener dos módulos de interfaz de red, aunque el diagrama de estratificación por capas muestra sólo una capa de interfaz de red en cada máquina.

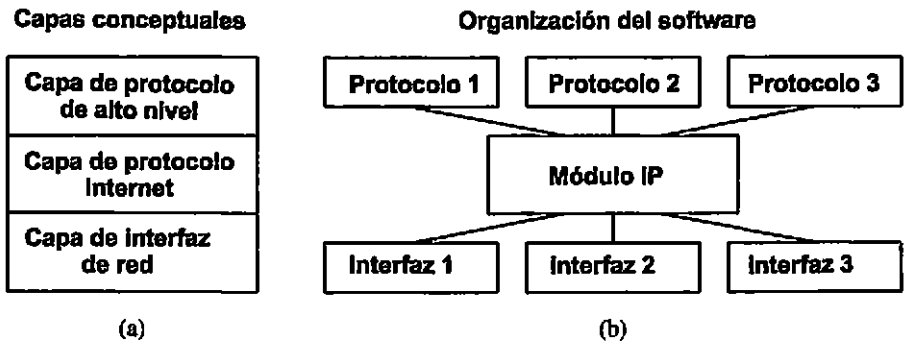


Figura 11.2 Una comparación de (a) estratificación por capas conceptual de protocolos y, (b) una visión realista de la organización del software que muestra varias interfaces de red entre IP y varios protocolos.

Como se muestra en la figura 11.3, un emisor en la máquina original transmite un mensaje que la capa de IP coloca en un datagrama y envía a través de la red *1*. En las máquinas intermedias el datagrama pasa hacia la capa IP, la cual rutea el datagrama de regreso, nuevamente (hacia una red diferente). Sólo cuando se alcanza la máquina en el destino final IP extrae el mensaje y lo pasa hacia arriba, hacia las capas superiores del software de protocolo.

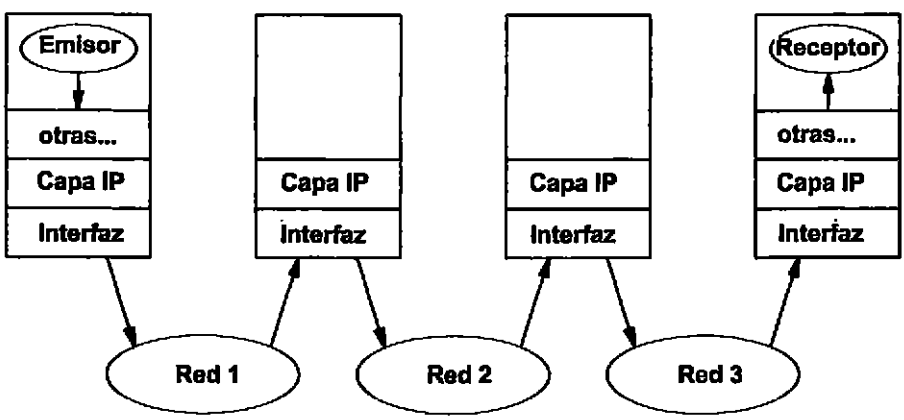


Figura 11.3 Trayectoria de un mensaje que atraviesa Internet desde un emisor, a través de dos máquinas intermedias, hasta un receptor. Las máquinas intermedias sólo envían el datagrama hacia la capa de software IP.

11.4 Funcionalidad de las capas

Una vez que se ha tomado la decisión de subdividir los problemas de comunicación en cuatro subproblemas y organizar el software de protocolo en módulos, de manera que cada uno maneje un subproblema, surge la pregunta: "¿qué tipo de funciones deben instalarse en cada módulo?" La pregunta no es fácil de responder por varias razones. En primer lugar, un conjunto de objetivos y condiciones determinan un problema de comunicación en particular, es posible elegir una organización que optimice el software de protocolo para ese problema. Segundo, incluso cuando se consideran los servicios generales a nivel de red, como un transporte confiable, es posible seleccionar entre distintas maneras de resolver el problema. Tercero, el diseño de una arquitectura de red (o de una red de redes) y la organización del software de protocolo están interrelacionados; no se puede diseñar a uno sin considerar al otro.

11.4.1 Modelo de referencia ISO de 7 capas

Existen dos ideas dominantes sobre la estratificación por capas de protocolos. La primera, basada en el trabajo realizado por la International Organization for Standardization (Organización Internacional para la Estandarización o ISO, por sus siglas en inglés), conocida como *Reference Model of Open System Interconnection* (*Modelo de referencia de interconexión de sistemas abiertos*) de ISO, denominada frecuentemente *modelo ISO*. El modelo ISO contiene 7 capas conceptuales organizadas como se muestra en la figura 11.4.

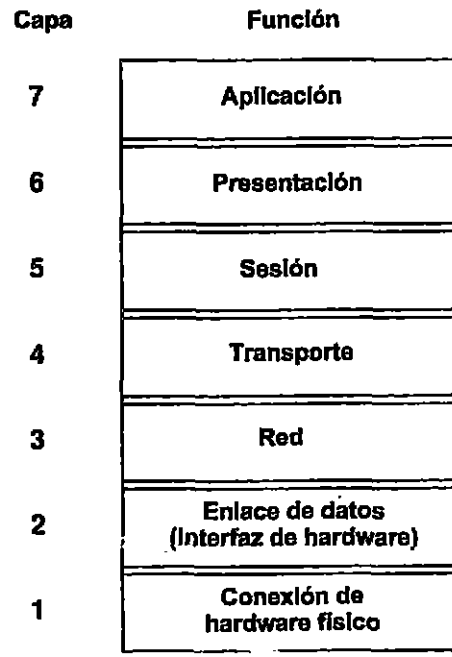


Figura 11.4 *Modelo de referencia de 7 capas ISO, para software de protocolo.*

El modelo ISO, elaborado para describir protocolos para una sola red, no contiene un nivel específico para el ruteo en el enlace de redes, como sucede con el protocolo TCP/IP.

11.5 X.25 y su relación con el modelo ISO

Aun cuando fue diseñado para proporcionar un modelo conceptual y no una guía de implementación, el esquema de estratificación por capas de ISO ha sido la base para la implementación de varios protocolos. Entre los protocolos comúnmente asociados con el modelo ISO, el conjunto de protocolos conocido como X.25 es probablemente el mejor conocido y el más ampliamente utilizado. X.25 fue establecido como una recomendación de la *Telecommunications Section* de la *International Telecommunications Union*[1] (ITU-TS), una organización internacional que recomienda estándares para los servicios telefónicos internacionales. X.25 ha sido adoptado para las redes públicas de datos y es especialmente popular en Europa. Consideraremos a X.25 para ayudar a explicar la estratificación por capas de ISO.

Dentro de la perspectiva de X.25, una red opera en gran parte como un sistema telefónico. Una red X.25 se asume como si estuviera formada por complejos conmutadores de paquetes que tienen la capacidad necesaria para el ruteo de paquetes. Los anfitriones no están comunicados de manera directa a los cables de comunicación de la red. En lugar de ello, cada anfitrión se comunica con uno de los conmutadores de paquetes por medio de una línea de comunicación serial. En cierto sentido la comunicación entre un anfitrión y un conmutador de paquetes X.25 es una red miniatura que consiste en un enlace serial. El anfitrión puede seguir un complicado procedimiento para transferir paquetes hacia la red.

- *Capa física.* X.25 especifica un estándar para la interconexión física entre computadoras anfitrión y conmutadores de paquetes de red, así como los procedimientos utilizados para transferir paquetes de una máquina a otra. En el modelo de referencia, el nivel *1* especifica la interconexión física incluyendo las características de voltaje y corriente. Un protocolo correspondiente, X.21, establece los detalles empleados en las redes públicas de datos.
- *Capa de enlace de datos.* El nivel *2* del protocolo X.25 especifica la forma en que los datos viajan entre un anfitrión y un conmutador de paquetes al cual está conectado. X.25 utiliza el término *trama* para referirse a la unidad de datos cuando ésta pasa entre un anfitrión y un conmutador de paquetes (es importante entender que la definición de X.25 de *trama* difiere ligeramente de la forma en que la hemos empleado hasta aquí). Dado que el hardware, como tal, entrega sólo un flujo de bits, el nivel de protocolo *2* debe definir el formato de las tramas y especificar cómo las dos máquinas reconocen las fronteras de la trama. Dado que los errores de transmisión pueden destruir los datos, el nivel de protocolo *2* incluye una detección de errores (esto es, una suma de verificación de trama). Finalmente, dado que la transmisión es no confiable, el nivel de protocolo *2* especifica un intercambio de acuses de recibo que permite a las dos máquinas saber cuándo se ha transferido una trama con éxito.

Hay protocolo de nivel 2, utilizado comúnmente, que se conoce como *High Level Data Link Communication (Comunicación de enlace de datos de alto nivel)*, mejor conocido por sus siglas, *HDLC*. Existen varias versiones del HDLC, la más reciente es conocida como *HDLC/LAPB*. Es

[1] La International Telecommunications Union fue llamada formalmente *Consultative Committee on International Telephony and Telegraphy (CCITT)*.

importante recordar que una transferencia exitosa en el nivel *2* significa que una trama ha pasado hacia un conmutador de paquetes de red para su entrega; esto no garantiza que el conmutador de paquetes acepte el paquete o que esté disponible para rutearlo.

- *Capa de red.* El modelo de referencia ISO especifica que el tercer nivel contiene funciones que completan la interacción entre el anfitrión y la red. Conocida como capa de *red* o *subred de comunicación,* este nivel define la unidad básica de transferencia a través de la red e incluye el concepto de direccionamiento de destino y ruteo. Debe recordarse que en el mundo de X.25 la comunicación entre el anfitrión y el conmutador de paquetes está conceptualmente aislada respecto al tráfico existente. Así, la red permitiría que paquetes definidos por los protocolos del nivel 3 sean mayores que el tamaño de la trama que puede ser transferida en el nivel *2*. El software del nivel 3 ensambla un paquete en la forma esperada por la red y utiliza el nivel *2* para transferirlo (quizás en fragmentos) hacia el conmutador de paquetes. El nivel 3 también debe responder a los problemas de congestionamiento en la red.
- *Capa de transporte.* El nivel 4 proporciona confiabilidad punto a punto y mantiene comunicados al anfitrión de destino con el anfitrión fuente. La idea aquí es que, así como en los niveles inferiores de protocolos se logra cierta confiabilidad verificando cada transferencia, la capa punto a punto duplica la verificación para asegurarse de que ninguna máquina intermedia ha fallado.
- *Capa de sesión.* Los niveles superiores del modelo ISO describen cómo el software de protocolo puede organizarse para manejar todas las funciones necesarias para los programas de aplicación. El comité ISO consideró el problema del acceso a una terminal remota como algo tan importante que asignó la capa 5 para manejarlo. De hecho, el servicio central ofrecido por las primeras redes públicas de datos consistía en una terminal para la interconexión de anfitriones. Las compañías proporcionaban en la red, mediante una línea de marcación, una computadora anfitrión de propósito especial, llamada *Packet Assembler and Disassembler (Ensamblador y desensamblador de paquetes o PAD,* por sus siglas en inglés). Los suscriptores, por lo general viajeros que transportaban su propia computadora y su módem, se ponían en contacto con la PAD local, haciendo una conexión de red hacia el anfitrión con el que deseaban comunicarse. Muchas compañías prefirieron comunicarse por medio de la red para su comunicación por larga distancia, porque resultaba menos cara que la marcación directa.
- *Capa de presentación.* La capa 6 de ISO está proyectada para incluir funciones que muchos programas de aplicación necesitan cuando utilizan la red. Los ejemplos comunes incluyen rutinas estándar que comprimen texto o convierten imágenes gráficas en flujos de bits para su transmisión a través de la red. Por ejemplo, un estándar ISO, conocido como *Abstract Syntax Notation 1* (*Notación de sintaxis abstracta 1* o *ASN.1,* por sus siglas en inglés), proporciona una representación de datos que utilizan los programas de aplicación. Uno de los protocolos TCP/IP, SNMP, también utiliza ASN.1 para representar datos.
- *Capa de aplicación.* Finalmente, la capa 7 incluye programas de aplicación que utilizan la red. Como ejemplos de esto se tiene al correo electrónico o a los programas de transferencia de archivos. En particular, el ITU-TS tiene proyectado un protocolo para correo electrónico, conocido como estándar *X.400*. De hecho, el ITU y el ISO trabajan juntos en el sistema de manejo de mensajes; la versión de ISO es conocida como *MOTIS*.

11.5.1 El modelo de estratificación por capas de TCP/IP de Internet

El segundo modelo mayor de estratificación por capas no se origina de un comité de estándares, sino que proviene de las investigaciones que se realizan respecto al conjunto de protocolos de TCP/IP. Con un poco de esfuerzo, el modelo ISO puede ampliarse y describir el esquema de estratificación por capas del TCP/IP, pero los presupuestos subyacentes son lo suficientemente distintos para distinguirlos como dos diferentes.

En términos generales, el software TCP/IP está organizado en cuatro capas conceptuales que se construyen sobre una quinta capa de hardware. La figura 11.5 muestra las capas conceptuales así como la forma en que los datos pasan entre ellas.

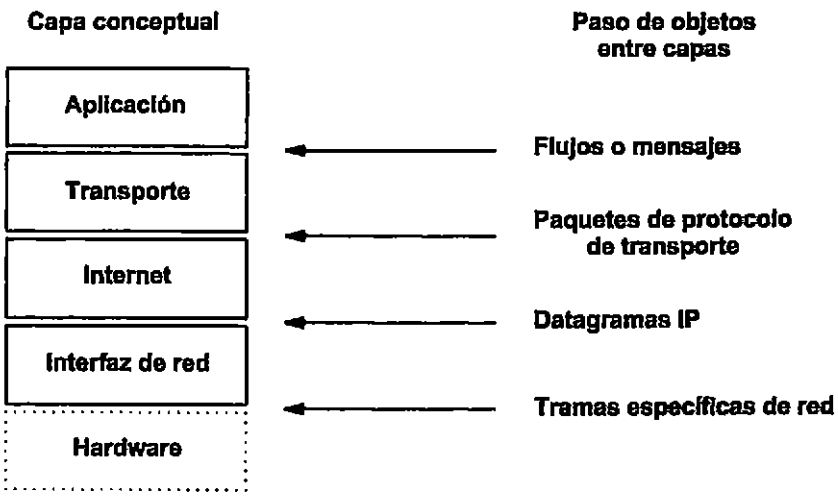


Figura 11.5 Las cuatro capas conceptuales del software TCP/IP y la forma en que los objetos pasan entre capas. La capa con el nombre *interfaz de red* se conoce con frecuencia con el nombre de capa de *enlace de datos*.

- *Capa de aplicación.* En el nivel más alto, los usuarios llaman a una aplicación que accesa servicios disponibles a través de la red de redes TCP/IP. Una aplicación interactúa con uno de los protocolos de nivel de transporte para enviar o recibir datos. Cada programa de aplicación selecciona el tipo de transporte necesario, el cual puede ser una secuencia de mensajes individuales o un flujo continuo de octetos. El programa de aplicación pasa los datos en la forma requerida hacia el nivel de transporte para su entrega.
- *Capa de transporte.* La principal tarea de *la capa de transporte* es proporcionar la comunicación entre un programa de aplicación y otro. Este tipo de comunicación se conoce frecuentemente como comunicación *punto a punto*. La capa de transporte regula el flujo de información. Puede también proporcionar un transporte confiable, asegurando que los datos lleguen sin errores y en secuencia. Para hacer esto, el software de protocolo de transporte tiene el lado de recepción enviando

acuses de recibo de retorno y la parte de envío retransmitiendo los paquetes perdidos. El software de transporte divide el flujo de datos que se está enviando en pequeños fragmentos (por lo general conocidos como *paquetes*) y pasa cada paquete, con una dirección de destino, hacia la siguiente capa de transmisión.

Aun cuando en la figura 11.5 se utiliza un solo bloque para representar la capa de aplicación, una computadora de propósito general puede tener varios programas de aplicación accesando la red de redes al mismo tiempo. La capa de transporte debe aceptar datos desde varios programas de usuario y enviarlos a la capa del siguiente nivel. Para hacer esto, se añade información adicional a cada paquete, incluyendo códigos que identifican qué programa de aplicación envía y qué programa de aplicación debe recibir, así como una suma de verificación. La máquina de recepción utiliza la suma de verificación para verificar que el paquete ha llegado intacto y utiliza el código de destino para identificar el programa de aplicación en el que se debe entregar.

- *Capa Internet.* Como ya lo hemos visto, la capa Internet maneja la comunicación de una máquina a otra. Ésta acepta una solicitud para enviar un paquete desde la capa de transporte, junto con una identificación de la máquina, hacia la que se debe enviar el paquete. Encapsula el paquete en un datagrama IP, llena el encabezado del datagrama, utiliza un algoritmo de ruteo para determinar si puede entregar el datagrama directamente o si debe enviarlo a un ruteador y pasar el datagrama hacia la interfaz de red apropiada para su transmisión. La capa Internet también maneja la entrada de datagramas, verifica su validez y utiliza un algoritmo de ruteo para decidir si el datagrama debe procesarse de manera local o debe ser transmitido. Para el caso de los datagramas direccionados hacia la máquina local, el software de la capa de red de redes borra el encabezado del datagrama y selecciona, de entre varios protocolos de transporte, un protocolo con el que manejará el paquete. Por último, la capa Internet envía los mensajes ICMP de error y control necesarios y maneja todos los mensajes ICMP entrantes.
- *Capa de interfaz de red.* El software TCP/IP de nivel inferior consta de una capa de interfaz de red responsable de aceptar los datagramas IP y transmitirlos hacia una red específica. Una interfaz de red puede consistir en un dispositivo controlador (por ejemplo, cuando la red es una red de área local a la que las máquinas están conectadas directamente) o un complejo subsistema que utiliza un protocolo de enlace de datos propio (por ejemplo, cuando la red consiste de conmutadores de paquetes que se comunican con anfitriones utilizando HDLC).

11.6 Diferencias entre X.25 y la estratificación por capas de Internet

Hay dos diferencias importantes y sutiles entre el esquema de estratificación por capas del TCP/IP y el esquema X.25. La primera diferencia gira en torno al enfoque de la atención de la confiabilidad, en tanto que la segunda comprende la localización de la inteligencia en el sistema completo.

11.6.1 Niveles de enlace y confiabilidad punto a punto

Una de las mayores diferencias entre los protocolos TCP/IP y X.25 reside en su enfoque respecto a los servicios confiables de entrega de datos. En el modelo X.25, el software de protocolo detecta y maneja errores en todos los niveles. En el nivel de enlace, protocolos complejos garantizan que la

transferencia, entre un anfitrión y un conmutador de paquetes que están conectados, se realice correctamente. Una suma de verificación acompaña a cada fragmento de datos transferido y el receptor envía acuses de recibo de cada segmento de datos recibido. El protocolo de nivel de enlace incluye intervalos de tiempo y algoritmos de retransmisión que evitan la pérdida de datos y proporcionan una recuperación automática después de las fallas de hardware y su reiniciación.

Los niveles sucesivos de X.25 proporcionan confiabilidad por sí mismos. En el nivel *3*, X.25 también proporciona detección de errores y recuperación de transferencia de paquetes en la red mediante el uso de sumas de verificación así como de intervalos de tiempo y técnicas de retransmisión. Por último, el nivel *4* debe proporcionar confiabilidad punto a punto pues tiene una correspondencia entre la fuente y el destino final para verificar la entrega.

En contraste con este esquema, el TCP/IP basa su estratificación por capas de protocolos en la idea de que la confiabilidad punto a punto es un problema. La filosofía de su arquitectura es sencilla: una red de redes se debe construir de manera que pueda manejar la carga esperada, pero permitiendo que las máquinas o los enlaces individuales pierdan o alteren datos sin tratar repetidamente de recuperarlos. De hecho, hay una pequeña o nula confiabilidad en la mayor parte del software de las capas de interfaz de red. En lugar de esto, las capas de transporte manejan la mayor parte de los problemas de detección y recuperación de errores.

El resultado de liberar la capa de interfaz de la verificación hace que el software TCP/IP sea mucho más fácil de entender e implementar correctamente. Los ruteadores intermedios pueden descartar datagramas que se han alterado debido a errores de transmisión. Pueden descartar datagramas que no se pueden entregar o que, a su llegada, exceden la capacidad de la máquina y pueden rutear de nuevo datagramas a través de vías con retardos más cortos o más largos sin informar a la fuente o al destino.

Tener enlaces no confiables significa que algunos datagramas no llegarán a su destino. La detección y la recuperación de los datagramas perdidos se establece entre el anfitrión fuente y el destino final y se le llama verificación *end-to-end*.[2] El software extremo a extremo que se ubica en la capa de transporte utiliza sumas de verificación, acuses de recibo e intervalos de tiempo para controlar la transmisión. Así, a diferencia del protocolo X.25, orientado a la conexión, el software TCP/IP enfoca la mayor parte del control de la confiabilidad hacia una sola capa.

11.6.2 Localización de la inteligencia y la toma de decisiones

Otra diferencia entre el modelo X.25 y el modelo TCP/IP se pone de manifiesto cuando consideramos la localización de la autoridad y el control. Como regla general, las redes que utilizan X.25 se adhieren a la idea de que una red es útil porque proporciona un servicio de transporte. El vendedor que ofrece el servicio controla el acceso a la red y monitorea el tráfico para llevar un registro de cantidades y costos. El prestador de servicios de la red también maneja internamente problemas como el ruteo, el control de flujo y los acuses de recibo, haciendo la transferencia confiable. Este enfoque hace que los anfitriones puedan (o necesiten) hacer muy pocas cosas. De hecho, la red es un sistema complejo e independiente en el que se pueden conectar computadoras anfitrión relativamente simples; los anfitriones por sí mismos participan muy poco en la operación de la red.

---

[2] N del T: podría entenderse como *verificación punto a punto* o extremo a extremo, a diferencia de la *confiabilidad punto a punto* de la que se habla en párrafos anteriores).

En contraste con esto, el TCP/IP requiere que los anfitriones participen en casi todos los protocolos de red. Ya hemos mencionado que los anfitriones implementan activamente la detección y la corrección de errores de extremo a extremo. También participan en el ruteo puesto que deben seleccionar una ruta cuando envían datagramas y participan en el control de la red dado que deben manejar los mensajes de control ICMP. Así, cuando la comparamos con una red X.25, una red de redes TCP/IP puede ser vista como un sistema de entrega de paquetes relativamente sencillo, el cual tiene conectados anfitriones inteligentes.

11.7 El principio de la estratificación por capas de protocolos

Independientemente del esquema de estratificación por capas que se utilice o de las funciones de las capas, la operación de los protocolos estratificados por capas se basa en una idea fundamental. La idea, conocida como *principio de estratificación por capas* puede resumirse de la siguiente forma:

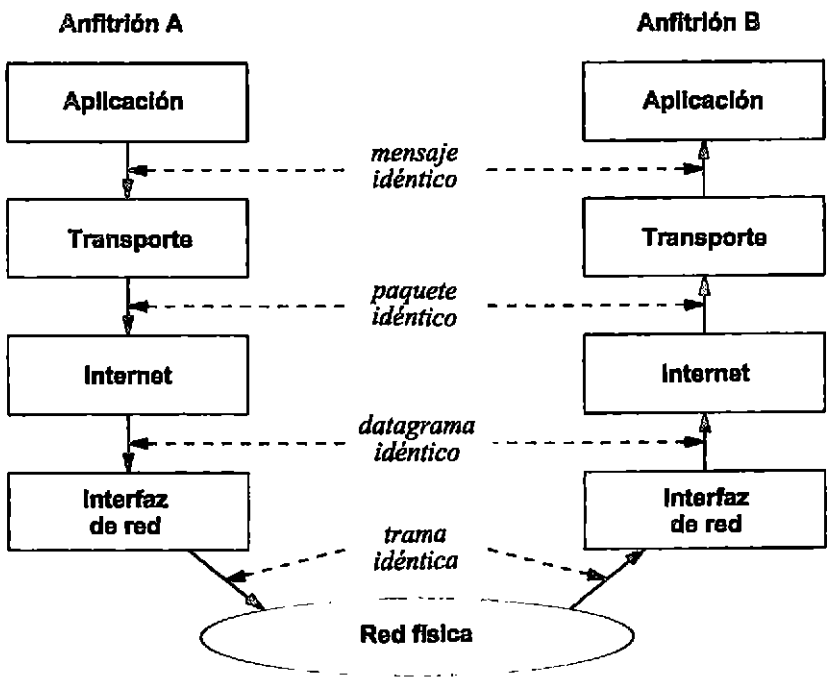


Figura 11.6 Trayectoria de un mensaje cuando pasa de la aplicación en un anfitrión a la aplicación en otro. Una capa *n* en el anfitrión *B* recibe exactamente el mismo objeto que la capa *n* correspondiente del anfitrión emisor *A*.

Los protocolos estratificados por capas están diseñados de modo que una capa n *en el receptor de destino reciba exactamente el mismo objeto enviado por la correspondiente capa* n *de la fuente.*

El principio de estratificación por capas explica por qué la estratificación por capas es una idea poderosa. Ésta permite que el diseñador de protocolos enfoque su atención hacia una capa a la vez, sin preocuparse acerca del desempeño de las capas inferiores. Por ejemplo, cuando se construye una aplicación para transferencia de archivos, el diseñador piensa sólo en dos copias del programa de aplicación que se correrá en dos máquinas y se concentrará en los mensajes que se necesitan intercambiar para la transferencia de archivos. El diseñador asume que la aplicación en el anfitrión receptor es exactamente la misma que en el anfitrión emisor.

La figura 11.6 ilustra cómo trabaja el principio de estratificación por capas:

11.7.1 Estratificación por capas en un ambiente de Internet TCP/IP

Nuestro planteamiento sobre el principio de estratificación por capas es un tanto vago y la ilustración de la figura 11.6 toca un tema importante dado que permite distinguir entre la transferencia desde una fuente hasta un destino final y la transferencia a través de varias redes. La figura 11.7.

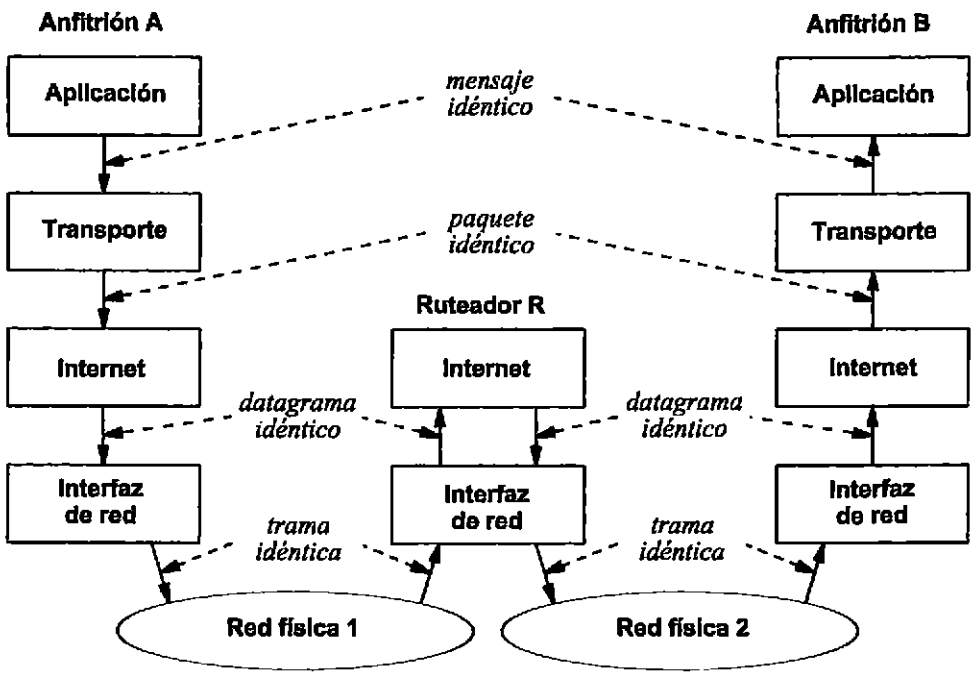


Figura 11.7 Principio de estratificación por capas cuando se utiliza un ruteador. La trama entregada al ruteador *R* es exactamente la trama enviada desde el anfitrión *A*, pero difiere de la trama enviada entre *R* y *B*.

ilustra la distinción y muestra el trayecto de un mensaje enviado desde un programa de aplicación en un anfitrión hacia la aplicación en otro a través de un ruteador.

Como se muestra en la figura, la entrega del mensaje utiliza dos estructuras de red separadas, una para la transmisión desde el anfitrión *A* hasta el ruteador *R* y otra del ruteador *R* al anfitrión *B*. El siguiente principio de trabajo de estratificación de capas indica que el marco entregado a *R* es idéntico al enviado por el anfitrión *A*. En contraste, las capas de aplicación y transporte cumplen con la condición punto a punto y están diseñados de modo que el software en la fuente se comunique con su par en el destino final. Así, el principio de la estratificación por capas establece que el paquete recibido por la capa de transporte en el destino final es idéntico al paquete enviado por la capa de transporte en la fuente original.

Es fácil entender que, en las capas superiores, el principio de estratificación por capas se aplica a través de la transferencia punto a punto y que en las capas inferiores se aplica en una sola transferencia de máquina. No es tan fácil ver cómo el principio de estratificación de capas se aplica a la estratificación Internet. Por un lado, hemos dicho que los anfitriones conectados a una red de redes deben considerarse como una gran red virtual, con los datagramas IP que hacen las veces de tramas de red. Desde este punto de vista, los datagramas viajan desde una fuente original hacia un destino final y el principio de la estratificación por capas garantiza que el destino final reciba exactamente el datagrama que envió la fuente. Por otra parte, sabemos que el encabezado "datagram" contiene campos, como "*time to live*", que cambia cada vez que el "*datagram*" pasa a través de un ruteador. Así, el destino final no recibirá exactamente el mismo diagrama que envió la fuente. Debemos concluir que, a pesar de que la mayor parte de los datagramas permanecen intactos cuando pasan a través de una red de redes, el principio de estratificación por capas sólo se aplica a los datagramas que realizan transferencias de una sola máquina. Para ser precisos, no debemos considerar que las capas de Internet proporcionan un servicio punto a punto.

11.8 Estratificación por capas en presencia de una subestructura de red

Recordemos, del capítulo 2, que algunas redes de área amplia contienen varios conmutadores de paquetes. Por ejemplo, una WAN puede consistir en ruteadores conectados a una red local en cada localidad así como a otros ruteadores que utilizan líneas en serie arrendadas. Cuando un ruteador recibe un datagrama, éste puede entregar el datagrama en su destino o en la red local, o transferir el datagrama a través de una línea serial hacia otro ruteador. La cuestión es la siguiente: "¿cómo se ajusta el protocolo utilizado en una línea serial con respecto al esquema de estratificación por capas del TCP/IP?" La respuesta depende de cómo considere el diseñador la interconexión con la línea serial.

Desde la perspectiva del IP, el conjunto de conexiones punto a punto entre ruteadores puede funcionar como un conjunto de redes físicas independientes o funcionar colectivamente como una sola red física. En el primer caso, cada enlace físico es tratado exactamente como cualquier otra red en una red de redes. A ésta se le asigna un número único de red (por lo general de clase C) y los dos anfitriones que comparten el enlace tiene cada uno una dirección única IP asignada para su conexión. Los ruteadores se añaden a la tabla de ruteo IP como lo harían para cualquier otra red. Un nuevo módulo de software se añade en la capa de interfaz de red para controlar el nuevo enlace

de hardware, pero no se realizan cambios sustanciales en el esquema de estratificación por capas. La principal desventaja del enfoque de redes independientes es la proliferación de números de red (uno por cada conexión entre dos máquinas), lo que ocasiona que las tablas de ruteo sean tan grandes como sea necesario. Tanto la *línea serial IP* (*Serial Line IP o SLIP*) como el *protocolo punto a punto* (*Point to Point Protocol o PPP*) tratan a cada enlace serial como una red separada.

El segundo método para ajustar las conexiones punto a punto evita asignar múltiples direcciones IP al cableado físico. En lugar de ello, se tratan a todas las conexiones colectivamente como una sola red independiente IP con su propio formato de trama, esquema de direccionamiento de hardware y protocolos de enlace de datos. Los ruteadores que emplean el segundo método necesitan sólo un número de red IP para todas las conexiones punto a punto.

Usar el enfoque de una sola red significa extender el esquema de estratificación por capas de protocolos para añadir una nueva capa de ruteo dentro de la red, entre la capa de interfaz de red y los dispositivos de hardware. Para las máquinas con una sola conexión punto a punto, una capa adicional parece innecesaria. Para comprender por qué es necesaria, considere una máquina con varias conexiones físicas punto a punto y recuerde de la figura 11.2 cómo la capa de interfaz de red se divide en varios módulos de software que controlan cada uno una red. Necesitamos añadir una interfaz de red nueva para la nueva red punto a punto, pero la interfaz nueva debe controlar varios dispositivos de hardware. Además, dado un datagrama para envío, la nueva interfaz debe seleccionar el enlace correcto por el que el datagrama será enviado. La figura 11.8 muestra la organización.

El software de la capa Internet pasa hacia la interfaz de red todos los datagramas que deberán enviarse por cualquier conexión punto a punto. La interfaz los pasa hacia el módulo de ruteo dentro de la red que, además, debe distinguir entre varias conexiones físicas y rutear el datagrama a través de la conexión correcta.

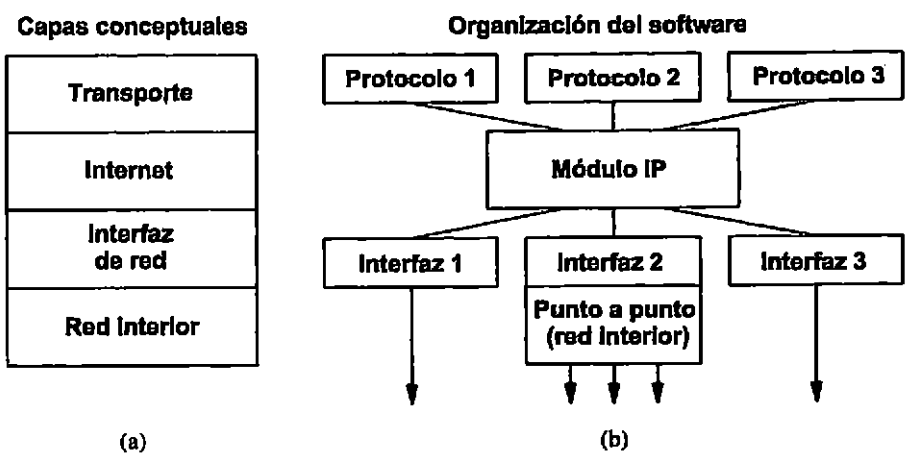


Figura 11.8 (a) posición de los conceptos de un protocolo de red interior, para conexiones punto a punto, cuando el IP la trata como una sola red IP, y (b) diagrama detallado de los módulos de software correspondientes. Cada flecha corresponde a un dispositivo físico.

El programador que diseña software de ruteo dentro de la red determina exactamente cómo selecciona el software un enlace físico. Por lo general, el algoritmo conduce a una tabla de ruteo dentro de la red. La tabla de ruteo dentro de la red es análoga a una tabla de ruteo de una red de redes en la que se especifica una transformación de la dirección de destino hacia la ruta. La tabla contiene pares de enteros, *(D, L)*, donde *D* es una dirección de destino de un anfitrión y *L* especifica una de las líneas físicas utilizadas para llegar al destino.

Las diferencias entre una tabla de ruteo de red de redes y una tabla de ruteo dentro de la red es que esta última, es mucho más pequeña. Contiene solamente información de ruteo para los anfitriones conectados directamente a la red punto a punto. La razón es simple: la capa Internet realiza la transformación de una dirección de destino arbitraria hacia una ruta de dirección específica antes de pasar el datagrama hacia una interfaz de red. De esta manera, la capa dentro de la red sólo debe distinguir entre máquinas en una sola red punto a punto.

11.9 Dos fronteras importantes en el modelo TCP/IP

La estratificación por capas conceptual incluye dos fronteras que podrían no ser obvias: una frontera de dirección de protocolo que separa los direccionamientos de alto nivel y de bajo nivel, y una frontera de sistema operativo que separa al sistema de los programas de aplicación.

11.9.1 Frontera de dirección de protocolo de alto nivel

Ahora que hemos visto la capa de software TCP/IP, podemos precisar una idea introducida en el capítulo 8: la partición de una frontera conceptual entre el software que utiliza direcciones de bajo nivel (físicas), con respecto a un software que utiliza direcciones de alto nivel (IP). Como se muestra en la figura 11.9, la frontera aparece entre la capa de interfaz de red y la capa de Internet. Esto es,

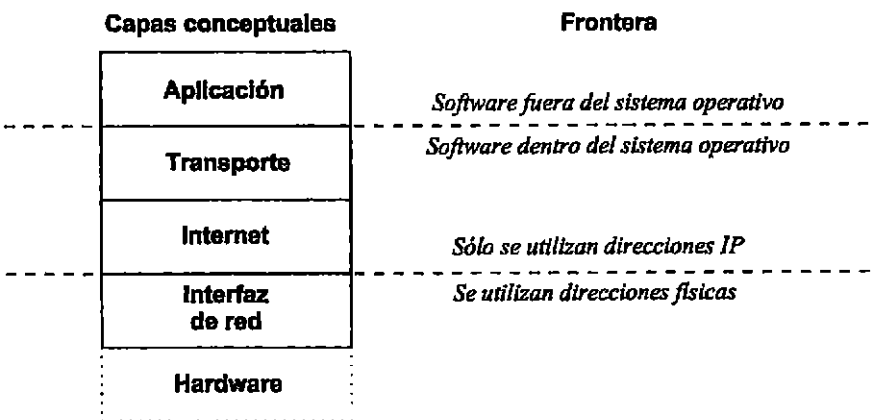


Figura 11.9 Relación entre la estratificación por capas conceptual y las fronteras, para el sistema operativo y las direcciones de protocolo de alto nivel.

Los programas de aplicación, así como todo el software del protocolo desde la capa de Internet hacia arriba, utiliza sólo direcciones IP; la capa de interfaz de red maneja direcciones físicas.

Así, protocolos como ARP pertenecen a la capa de interfaz de red. Estos no son parte del IP.

11.9.2 Frontera de sistema operativo

La figura 11.9 muestra otra frontera también importante, la división entre el software que generalmente se considera parte del sistema operativo respecto al software que no lo es. En tanto que cada implantación del TCP/IP determina cómo se establece la distinción, muchos siguen el esquema mostrado. Dado que los colocan dentro el sistema operativo, el paso de datos entre las capas inferiores del software de protocolo es mucho menos caro que su paso entre un programa de aplicación y una capa de transporte. En el capítulo 20, se trata el problema con mayor detalle y se describe un ejemplo de la interfaz de un sistema operativo.

11.10 La desventaja de la estratificación por capas

Se ha mencionado el hecho de que la estratificación por capas es una idea fundamental que proporciona las bases para el diseño de protocolos. Permite al diseñador dividir un problema complicado en subproblemas y resolver cada parte de manera independiente. Por desgracia, el software resultante de una estratificación por capas estrictas puede ser muy ineficaz. Como ejemplo, considere el trabajo de la capa de transporte. Debe aceptar un flujo de octetos desde un programa de aplicación, dividir el flujo en paquetes y enviar cada paquete a través de la red de redes. Para optimizar la transferencia, la capa de transporte debe seleccionar el tamaño de paquete más grande posible que le permita a un paquete viajar en una trama de red. En particular, si la máquina de destino está conectada a una máquina de la misma red de la fuente, sólo la red física se verá involucrada en la transferencia, así, el emisor puede optimizar el tamaño del paquete para esta red. Si el software preserva una estricta estratificación por capas, sin embargo, la capa de transporte no podrá saber cómo ruteará el módulo de Internet el tráfico o qué redes están conectadas directamente. Más aún, la capa de transporte no comprenderá el datagrama o el formato de trama ni será capaz de determinar cómo deben ser añadidos muchos octetos de encabezado a un paquete. Así, una estratificación por capas estricta impedirá que la capa de transporte optimice la transferencia.

Por lo general, las implantaciones atenúan el esquema estricto de la estratificación por capas cuando construyen software de protocolo. Permiten que información como la selección de ruta y la MTU de red se propaguen hacia arriba. Cuando los búfers realizan el proceso de asignación, generalmente dejan espacio para encabezados que serán añadidos por los protocolos de las capas de bajo nivel y pueden retener encabezados de las tramas entrantes cuando pasan hacia protocolos de capas superiores. Tal optimización puede producir mejoras notables en la eficiencia siempre y cuando conserve la estructura básica en capas.

11.11 La idea básica detrás del multiplexado y el demultiplexado

Los protocolos de comunicación utilizan técnicas de *multiplexado* y *demultiplexado* a través de la jerarquía de capas. Cuando envía un mensaje, la computadora fuente incluye bits extras que codifican el tipo de mensaje, el programa de origen y los protocolos utilizados. Finalmente, todos los mensajes son colocados dentro de tramas de red para transferirse y combinarse en flujos de paquetes. En el extremo de recepción, la máquina destino se vale de información extra para guiar el proceso.

Considere el ejemplo de demultiplexado que se muestra en la figura 11.10.

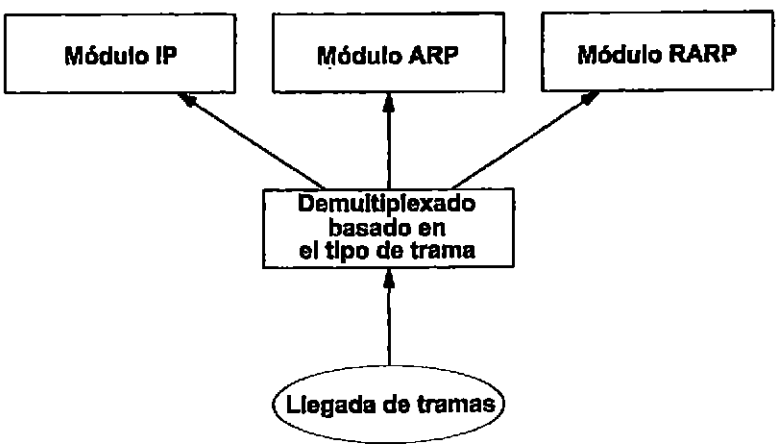


Figura 11.10 Demultiplexado de tramas entrantes basado en el campo de tipo que se encuentra en el encabezado de la trama.

La figura muestra de qué manera utiliza el software, en la capa de interfaz de red, el tipo de trama para seleccionar un procedimiento que permita manejar las tramas entrantes. Se dice que la interfaz de red *demultiplexa* la trama con base en este tipo. Para hacer posible la selección, el software en la máquina fuente debe establecer el campo del tipo de trama antes de la transmisión. Así, cada módulo de software que envía tramas emplea el campo de tipo para especificar el contenido de la trama.

El multiplexado y el demultiplexado se presentan en casi todas las capas de protocolo. Por ejemplo, luego de que la interfaz de red demultiplexa tramas y pasa las tramas que contienen datagramas IP hacia el módulo IP, el software IP extrae el datagrama y lo demultiplexa con base en el protocolo de transporte. La figura 11.11 muestra el multiplexado en la capa Internet.

Para decidir cómo manejar un datagrama, el software de red de redes examina el encabezado de un datagrama y, para su manejo, selecciona un protocolo con base en el tipo de datagrama. En el ejemplo, los tipos posibles de datagramas son: *ICMP*, que ya hemos examinado, y *UDP*, *TCP* y *EGP*, que examinaremos en capítulos posteriores.

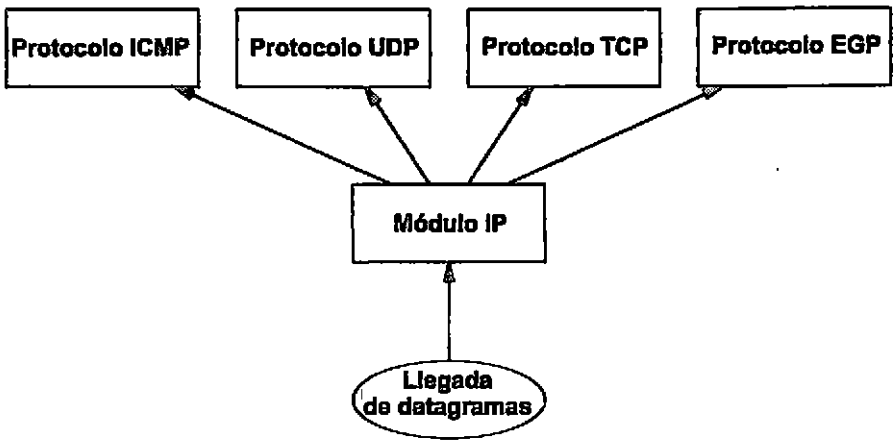


Figura 11.11 Demultiplexado en la capa Internet. El software IP selecciona un procedimiento apropiado para manejar un datagrama, basándose en el campo de tipo de protocolo, localizado en el encabezado del datagrama.

11.12 Resumen

Los protocolos son los estándares que especifican cómo se representan los datos cuando son transferidos de una máquina a otra. También, especifican cómo se da la transferencia, cómo se detectan los errores y cómo se envían los acuses de recibo. Para simplificar el diseño y la implantación de los protocolos, los problemas de comunicación se transfieren hacia subproblemas que se pueden resolver de manera independiente. Cada problema se asigna a un protocolo separado.

La idea de la estratificación por capas es fundamental porque proporciona una estructura conceptual para el diseño de protocolos. En el modelo por capas, cada capa maneja una parte de los problemas de comunicación y generalmente se asocian a un protocolo. Los protocolos siguen el principio de la estratificación por capas, el cual establece que la implantación del software de una capa *n* en una máquina de destino recibe exactamente la implementación del software de la capa *n* en la fuente de la máquina emisora.

Examinamos el modelo de referencia de Internet de 4 capas, así como el modelo de referencia de 7 capas ISO. En ambos casos el modelo de estratificación por capas proporciona sólo una estructura conceptual para el software de protocolo. Los protocolos X.25 de ITU-TS siguen el modelo de referencia ISO y proporcionan un ejemplo de servicio de comunicación confiable ofrecido por una infraestructura comercial, mientras que los protocolos TCP/IP proporcionan un ejemplo diferente de un esquema de estratificación por capas.

En la práctica, el software de protocolo utiliza el multiplexado y el demultiplexado para distinguir entre varios protocolos dentro de una capa dada, haciendo el software de protocolo más complejo que como lo sugiere el modelo de estratificación por capas.

PARA CONOCER MÁS

Postel (RFC 791) ofrece un bosquejo del esquema de estratificación por capas del Protocolo Internet y Clark (RFC 817) analiza los efectos de la estratificación por capas en las implantaciones. Saltzer, Reed y Clark (1984) plantean que la verificación extremo a extremo es importante. Chesson (1987) hace una exposición controvertida, según la cual la estratificación por capas produce un rendimiento total de la red malo e intolerable. En el volumen 2 de esta obra, se examina la estratificación por capas a detalle y se muestra un ejemplo de implantación con el que se logra la eficiencia mediante un compromiso entre la estratificación por capas estricta y el paso de punteros entre capas.

Los documentos de protocolo ISO (1987a) y (1987b) describen al ASN.1 en detalle. Sun (RFC 1014) describe XDR, un ejemplo de lo que podría llamarse un protocolo de presentación TCP/IP. Clark trata el paso de la información hacia arriba a través de las capas (Clark 1985).

EJERCICIOS

11.1 Estudie el modelo de estratificación por capas con mayor detalle. ¿Cómo describiría el modelo de comunicación en una red de área local como Ethernet?

11.2 Elabore un caso en el que TCP/IP se mueva hacia una arquitectura de protocolo de cinco niveles que incluya una capa de presentación. (Sugerencia: varios programas utilizan el protocolo XDR, Courier y ASN.1)

11.3 ¿Piensa usted que un solo protocolo de presentación emergería eventualmente remplazando a todos los demás? ¿Por qué sí, o por qué no?

11.4 Compare y contraste el formato de datos etiquetado utilizado por el esquema de presentación ASN.1, con el formato no etiquetado, utilizado por XDR. Especifique situaciones en que uno sea mejor que el otro.

11.5 Encuentre cómo utiliza un sistema UNIX la estructura *mbuf* para hacer eficiente el software de protocolo de estratificación por capas.

11.6 Lea acerca del mecanismo *streams* del sistema UNIX V. ¿En qué forma ayuda a hacer más fácil la implantación del protocolo? ¿Cuál es su mayor desventaja?

12

Protocolo de datagrama de usuario (UDP)

12.1 Introducción

En los capítulos anteriores, se describió una red de redes TCP/IP capaz de transferir datagramas IP entre computadores anfitriones, donde cada datagrama se rutea a través de la red, basándose en la dirección IP de destino. En la capa del Protocolo Internet, una dirección de destino identifica una computadora anfitrión; no se hace ninguna otra distinción con respecto a qué usuario o qué programa de aplicación recibirá el datagrama. En este capítulo se amplía el grupo de protocolos TCP/IP al agregar un mecanismo que distingue entre muchos destinos dentro de un anfitrión, permitiendo que varios programas de aplicación que se ejecutan en una computadora envíen y reciban datagramas en forma independiente.

12.2 Identificación del destino final

Los sistemas operativos de la mayor parte de las computadoras aceptan la multiprogramación, que significa permitir que varios programas de aplicación se ejecuten al mismo tiempo. Utilizando la jerga de los sistemas operativos, nos referimos a cada programa en ejecución como un *proceso, tarea, programa de aplicación* o *proceso a nivel de usuario*; a estos sistemas se les llama sistemas multitarea. Puede parecer natural decir que un proceso es el destino final de un mensaje. Sin embargo, especificar que un proceso en particular en una máquina en particular es el destino final para un datagrama es un poco confuso. Primero, por que los procesos se crean y destruyen de manera dinámica, los transmisores rara vez saben lo suficiente para identificar un proceso en otra máquina. Se-

gundo, nos gustaría poder reemplazar los procesos que reciben datagramas, sin tener que informar a todos los transmisores (por ejemplo, reiniciar una máquina puede cambiar todos los procesos, pero los transmisores no están obligados a saber sobre los nuevos procesos). Tercero, necesitamos identificar los destinos de las funciones que implantan sin conocer el proceso que implanta la función (por ejemplo, permitir que un transmisor contacte un servidor de archivos sin saber qué proceso en la máquina de destino implanta la función de servidor de archivos). También es importante saber que, en los sistemas que permiten que un solo proceso maneje dos o más funciones, es esencial que encontremos una forma para que un proceso decida exactamente qué función desea el transmisor.

En vez de pensar en un proceso como destino final, imaginaremos que cada máquina contiene un grupo de puntos abstractos de destino, llamados *puertos de protocolo*. Cada puerto de protocolo se identifica por medio de un número entero positivo. El sistema operativo local proporciona un mecanismo de interfaz que los procesos utilizan para especificar o accesar un puerto.

La mayor parte de los sistemas operativos proporciona un acceso síncrono a los puertos. Desde el punto de vista de un proceso en particular, el acceso síncrono significa que los cómputos se detienen durante una operación de acceso a puerto. Por ejemplo, si un proceso intenta extraer datos de un puerto antes de que llegue cualquier dato, el sistema operativo detiene (bloquea) temporalmente el proceso hasta que lleguen datos. Una vez que esto sucede, el sistema operativo pasa los datos al proceso y lo vuelve a iniciar. En general, los puertos tienen *memoria intermedia*, para que los datos que llegan antes de que un proceso esté listo para aceptarlos no se pierdan. Para lograr la colocación en memoria intermedia, el software de protocolo, localizado dentro del sistema operativo, coloca los paquetes que llegan de un puerto de protocolo en particular en una cola de espera (finita) hasta que un proceso los extraiga.

Para comunicarse con un puerto externo, un transmisor necesita saber tanto la dirección IP de la máquina de destino como el número de puerto de protocolo del destino dentro de la máquina. Cada mensaje debe llevar el número del *puerto de destino* de la máquina a la que se envía, así como el número de *puerto de origen* de la máquina fuente a la que se deben direccionar las respuestas. Por lo tanto, es posible que cualquier proceso que recibe un mensaje conteste al transmisor.

12.3 Protocolo de datagrama de usuario

En el grupo de protocolos TCP/IP, el *Protocolo de datagrama de usuario o UDP* proporciona el mecanismo primario que utilizan los programas de aplicación para enviar datagramas a otros programas de aplicación. El UDP proporciona puertos de protocolo utilizados para distinguir entre muchos programas que se ejecutan en la misma máquina. Esto es, además de los datos, cada mensaje UDP contiene tanto el número de puerto de destino como el número de puerto de origen, haciendo posible que el software UDP en el destino entregue el mensaje al receptor correcto y que éste envíe una respuesta.

El UDP utiliza el Protocolo Internet subyacente para transportar un mensaje de una máquina a otra y proporciona la misma semántica de entrega de datagramas, sin conexión y no confiable que el IP. No emplea acuses de recibo para asegurarse de que llegan mensajes, no ordena los mensajes entrantes, ni proporciona retroalimentación para controlar la velocidad a la que fluye la información entre las máquinas. Por lo tanto, los mensajes UDP se pueden perder, duplicar o llegar sin orden.

Además, los paquetes pueden llegar más rápido de lo que el receptor los puede procesar. Podemos resumir que:

> *El protocolo de datagrama de usuario (UDP) proporciona un servicio de entrega sin conexión y no confiable, utilizando el IP para transportar mensajes entre máquinas. Emplea el IP para llevar mensajes, pero agrega la capacidad para distinguir entre varios destinos dentro de una computadora anfitrión.*

Un programa de aplicación que utiliza el UDP acepta toda la responsabilidad por el manejo de problemas de confiabilidad, incluyendo la pérdida, duplicación y retraso de los mensajes, la entrega fuera de orden y la pérdida de conectividad. Por desgracia, los programadores de aplicaciones a menudo olvidan estos problemas cuando diseñan software. Además, como los programadores a menudo prueban el software de red utilizando redes de área local, altamente confiables y de baja demora, el procedimiento de pruebas puede no evidenciar las fallas potenciales. Por lo tanto, muchos programas de aplicación que confían en el UDP trabajan bien en un ambiente local, pero fallan dramáticamente cuando se utilizan en una red de redes TCP/IP más grande.

12.4 Formato de los mensajes UDP

Cada mensaje UDP se conoce como *datagrama de usuario*. Conceptualmente, un datagrama de usuario consiste de dos partes: un encabezado UDP y un área de datos UDP. Como se muestra en la figura 12.1, el encabezado se divide en cuatro campos de 16 bits, que especifican el puerto desde el que se envió el mensaje, el puerto para el que se destina el mensaje, la longitud del mensaje y una suma de verificación UDP.

0	16 31
PUERTO UDP DE ORIGEN	PUERTO UDP DE DESTINO
LONGITUD DEL MENSAJE UDP	SUMA DE VERIFICACIÓN UDP
DATOS	
...	

Figura 12.1 Formato de los campos en un datagrama UDP.

Los campos *PUERTO DE ORIGEN* y *PUERTO DE DESTINO* contienen los números de puerto del protocolo UDP utilizados para el demultiplexado de datagramas entre los procesos que los esperan recibir. El *PUERTO DE ORIGEN* es opcional. Cuando se utiliza, especifica la parte a la que se deben enviar las respuestas, de lo contrario, puede tener valor de cero.

El campo de *LONGITUD* contiene un conteo de los octetos en el datagrama UDP, incluyendo el encabezado y los datos del usuario UDP. Por lo tanto, el valor mínimo para el campo *LONGITUD* es de ocho, que es la longitud del encabezado.

La suma de verificación UDP es opcional y no es necesario utilizarla; un valor de cero en el campo *SUMA DE VERIFICACIÓN* significa que la suma no se computó. Los diseñadores decidieron hacer opcional la suma de verificación a fin de permitir que las implantaciones operen con poco trabajo computacional cuando utilicen UDP en una red de área local altamente confiable. Sin embargo, recuerde que el IP no computa una suma de verificación de la porción de datos de un datagrama IP. Así que, la suma de verificación UDP proporciona la única manera de garantizar que los datos lleguen intactos, por lo que se debe utilizar.

Los principiantes, a menudo, se preguntan qué sucede con los mensajes UDP en los que la suma de verificación computada es cero. Un valor computacional de cero es posible debido a que el UDP utiliza el mismo algoritmo de suma de verificación que el IP: divide los datos en cantidades de 16 bits y computa el complemento de los unos de su suma de complementos de los unos. De manera sorprendente, el cero no es un problema debido a que la aritmética de los unos tiene dos representaciones para el cero: todos los bits como cero o todos los bits como uno. Cuando la suma de verificación computada es igual a cero, el UDP utiliza la representación con todos los bits como uno.

12.5 Pseudo-encabezado UDP

La suma de verificación UDP abarca más información de la que está presente en el datagrama UDP por sí solo. Para computar la suma de verificación, el UDP añade un *pseudo-encabezado* al datagrama UDP, adjunta un octeto de ceros para rellenar el datagrama y alcanzar exactamente un múltiplo de 16 bits, y computa la suma de verificación sobre todo el conjunto. El octeto utilizado como relleno y el pseudo-encabezado no se transmiten con el datagrama UDP, ni se incluyen en su longitud. Para computar una suma de verificación, el software primero almacena un cero en el campo de *SUMA DE VERIFICACIÓN*, luego, acumula una suma de complemento de 16 bits de todo el conjunto, incluyendo el pseudo-encabezado, el encabezado UDP y los datos del usuario.

El propósito de utilizar un pseudo-encabezado es para verificar que el datagrama UDP llegó a su destino correcto. La clave para entender el uso del pseudo-encabezado reside en darse cuenta de que el destino correcto consiste en una máquina específica y en un puerto de protocolo específico dentro de dicha máquina. Por sí mismo, el encabezado UDP sólo especifica el número de puerto de protocolo. Por lo tanto, para verificar un destino, el UDP en la máquina transmisora computa una suma de verificación que cubre tanto la dirección IP de destino como el datagrama UDP. En el destino final, el software UDP revisa la suma de verificación utilizando la dirección IP de destino, obtenida del encabezado del datagrama IP que transportó el mensaje UDP. Si la suma concuerda, debe ser verdad que el datagrama llegó al anfitrión de destino deseado, así como al puerto de protocolo correcto dentro del anfitrión.

El pseudo-encabezado utilizado en el cómputo de la suma de verificación UDP consiste en 12 octetos de datos, distribuidos como se muestra en la figura 12.2. Los campos en el pseudo-encabezado etiquetados como *DIRECCIÓN IP DE ORIGEN* y *DIRECCIÓN IP DE DESTINO*, contienen las direcciones IP que se utilizarán cuando se envíe el mensaje UDP. El campo *PROTO* contiene el código del tipo de protocolo IP (17 para UDP) y el campo *LONGITUD UDP* contiene la longitud del

datagrama UDP (sin incluir el pseudo-encabezado). Para revisar la suma de verificación, el receptor debe extraer estos campos del encabezado IP, ensamblarlos en el formato de pseudo-encabezado y recomputar la suma.

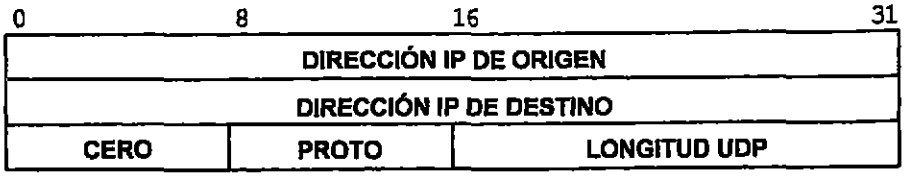


Figura 12.2 Los 12 octetos de un pseudo-encabezado que se utilizan durante el cómputo de la suma de verificación UDP.

12.6 Encapsulación de UDP y estratificación por capas de protocolos

El UDP proporciona nuestro primer ejemplo de un protocolo de transporte. En el modelo de estratificación por capas del capítulo 11, el UDP reside sobre la capa del Protocolo Internet. Conceptualmente, los programas de aplicación accesan el UDP, que utiliza el IP para enviar y recibir datagramas como se muestra en la figura 12.3.

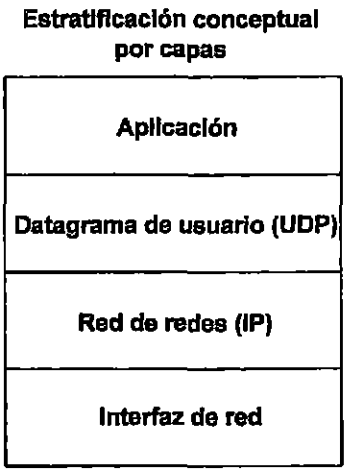


Figura 12.3 Estratificación conceptual por capas de UDP entre programas de aplicación e IP.

Estratificar por capas el UDP por encima del IP significa que un mensaje UDP completo, incluyendo el encabezado UDP y los datos, se encapsula en un datagrama IP mientras viaja a través de una red de redes, tal como se muestra en la figura 12.4.

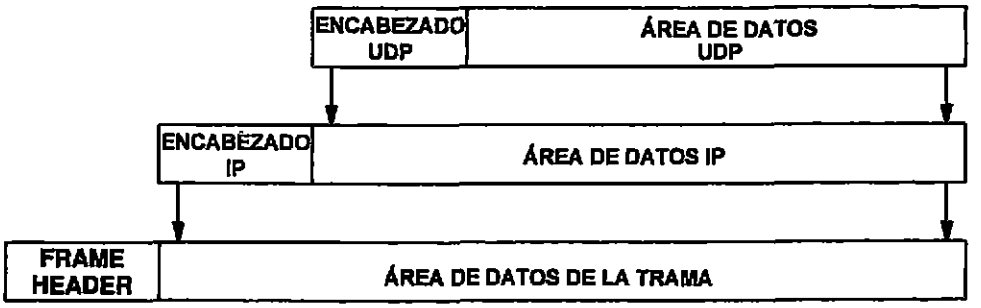


Figura 12.4 Datagrama UDP encapsulado en un datagrama IP para su transmisión a través de una red de redes. El datagrama se encapsula en una trama cada vez que viaja a través de una red.

Para los protocolos que hemos examinado, la encapsulación significa que el UDP adjunta un encabezado a los datos que un usuario envía y lo pasa al IP. La capa IP adjunta un encabezado a lo que recibe del UDP. Y por último, la capa de interfaz de red introduce el datagrama en una trama antes de enviarlo de una máquina a otra. El formato de la trama depende de la tecnología subyacente de red. Por lo general, las tramas de red incluyen un encabezado adicional.

En la entrada, un paquete llega en la capa más baja del software de red y comienza su ascenso a través de capas sucesivamente más altas. Cada capa quita un encabezado antes de pasar el mensaje para que, en el momento en que el nivel más alto pasa los datos al proceso receptor, todos los encabezados se hayan removido. Por lo tanto, el encabezado exterior corresponde a la capa más baja de protocolo y el encabezado interior a la más alta de protocolo. Cuando se considera cómo se insertan y remueven los encabezados, es importante tener en cuenta el principio de la estratificación por capas. En lo particular, observe que este principio se aplica al UDP, así que el datagrama UDP que recibió el IP en la máquina de destino es idéntico al datagrama que el UDP pasó al IP en la máquina de origen. También, los datos que el UDP entrega a un proceso usuario en la máquina receptora serán los mismos que un proceso usuario pase al UDP en la máquina transmisora.

La división de funciones entre varias capas de protocolos es inflexible y clara:

> *La capa IP sólo es responsable de transferir datos entre un par de anfitriones dentro de una red de redes, mientras que la capa UDP solamente es responsable de diferenciar entre varias fuentes o destinos dentro de un anfitrión.*

Por lo tanto, sólo el encabezado IP identifica los anfitriones de origen y destino; sólo la capa UDP identifica los puertos de origen o destino dentro de un anfitrión.

12.7 Estratificación por capas y cómputo UDP de suma de verificación

Los lectores observadores habrán notado una contradicción aparente entre las reglas de la estratificación por capas y el cómputo de la suma de verificación UDP. Recuerde que la suma de verificación UDP incluye un pseudo-encabezado que tiene campos para las direcciones IP de origen y de destino. Se puede argüir que el usuario debe conocer la dirección IP de destino cuando envía un datagrama UDP y que éste la debe pasar a la capa UDP. Por lo tanto, la capa UDP puede obtener la dirección IP de destino sin interactuar con la capa IP. Sin embargo, la dirección IP de origen depende de la ruta que el IP seleccione para el datagrama, debido a que esta dirección identifica la interfaz de red sobre la que se transmite el datagrama. Por lo tanto, el UDP no puede conocer una dirección IP de origen a menos que interactúe con la capa IP.

Asumimos que el software UDP pide a la capa IP que compute la dirección IP de origen y (posiblemente) la de destino, las utiliza para construir un pseudo-encabezado, computa la suma de verificación, descarta el pseudo-encabezado y transfiere a la capa IP el datagrama UDP para su transmisión. Con un enfoque alternativo, que produce una mayor eficiencia, se logra que la capa UDP encapsule el datagrama UDP en un datagrama IP, obtenga del IP la dirección de origen, almacene las direcciones tanto de origen como de destino en los campos apropiados del encabezado del datagrama, compute la suma de verificación UDP y pase el datagrama IP a la capa IP, que sólo necesita llenar los campos restantes del encabezado IP.

¿La fuerte interacción entre el UDP y el IP viola nuestra premisa básica de que la estratificación por capas refleja la separación de funcionalidad? Sí. El UDP está fuertemente integrado al protocolo IP. Es claramente una transigencia de la separación pura, diseñado enteramente por razones prácticas. Deseamos pasar por alto la violación de estratificación por capas, ya que es imposible identificar plenamente un programa de aplicación de destino sin especificar la máquina de destino y porque queremos realizar, de manera eficaz, la transformación de direcciones utilizadas por el UDP y el IP. En uno de los ejercicios se examina este tema desde un punto de vista diferente y se pide al lector que considere si el UDP se debe separar de IP.

12.8 Multiplexado, demultiplexado y puertos de UDP

En el capítulo 11, vimos que el software a través de las capas de una jerarquía de protocolos debe multiplexar y demultiplexar muchos objetos en la capa siguiente. El software UDP proporciona otro ejemplo de multiplexado y demultiplexado. Acepta datagramas UDP de muchos programas de aplicación y los pasa a IP para su transmisión, también acepta datagramas entrantes UDP del IP y los transfiere al programa de aplicación apropiado.

Conceptualmente, todo el multiplexado y el demultiplexado entre el software UDP y los programas de aplicación ocurre a través del mecanismo de puerto. En la práctica, cada programa de aplicación debe negociar con el sistema operativo para obtener un puerto del protocolo y un número de puerto asociado, antes de poder enviar un datagrama UDP.[1] Una vez que se asigna el puerto,

[1] Por ahora, describiremos los puertos en forma abstracta; en el capítulo 20, se proporciona un ejemplo del sistema operativo que antiguamente se empleaba para crear y utilizar puertos.

cualquier datagrama que envíe el programa de aplicación a través de él, tendrá el número de puerto en el campo *PUERTO DE ORIGEN UDP*.

Mientras procesa la entrada, el UDP acepta datagramas entrantes del software IP y los demultiplexa, basándose en el puerto de destino UDP, como se muestra en la figura 12.5.

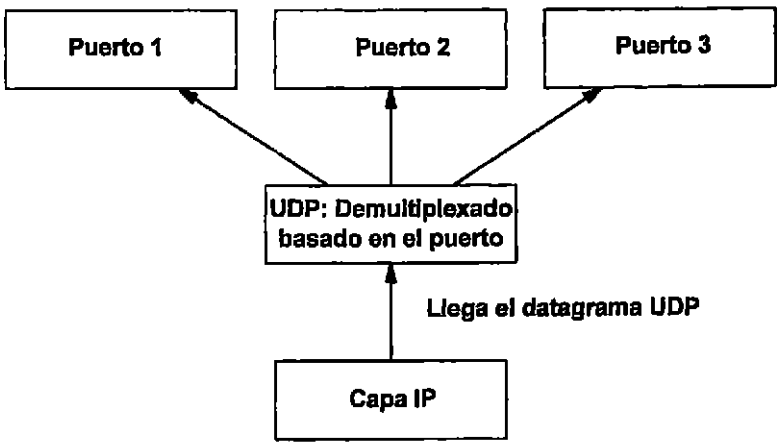


Figura 12.5 Ejemplo del demultiplexado de una capa sobre el IP. El UDP utiliza el número de puerto UDP de destino para seleccionar el puerto apropiado de destino para los datagramas entrantes.

La forma más fácil de pensar en un puerto UDP es en una cola de espera. En la mayor parte de las implantaciones, cuando un programa de aplicación negocia con el sistema operativo la utilización de cierto puerto, el sistema operativo crea una cola de espera interna que puede almacenar los mensajes que lleguen. A menudo, la aplicación puede especificar o modificar el tamaño de la cola de espera. Cuando el UDP recibe un datagrama, verifica si el número de puerto de destino corresponde a uno de los puertos que están en uso. Si no, envía mensaje de error ICMP de *puerto no accesible* y descarta el datagrama. Si encuentra una correspondencia, el UDP pone en cola de espera el nuevo datagrama, en el puerto en que lo pueda accesar un programa de aplicación. Por supuesto, ocurre un error si el puerto se encuentra lleno y el UDP descarta el datagrama entrante.

12.9 Números de puerto UDP reservados y disponibles

¿Cómo se deben asignar los números de puerto de protocolo? El problema es importante ya que dos computadoras necesitan estar de acuerdo en los números de puerto antes de que puedan interoperar. Por ejemplo, cuando la computadora *A* quiere obtener un archivo de la computadora *B*, necesita saber qué puerto utiliza el programa de transferencia de archivos en la computadora *B*. Existen dos enfoques fundamentales para la asignación de puertos. El primero se vale de una autoridad central. Todos se ponen de acuerdo en permitir que una autoridad central asigne los números de puerto

conforme se necesitan y que publique la lista de todas las asignaciones. Entonces, todo el software se diseña de acuerdo con la lista. Este enfoque, a veces, se conoce como *enfoque universal* y las asignaciones de puerto especificadas por la autoridad se conocen como *asignaciones bien conocidas de puerto*.

El segundo enfoque para la asignación de puertos emplea la transformación dinámica. En este enfoque, los puertos no se conocen de manera global. En vez de eso, siempre que un programa necesita un puerto, el software de red le asigna uno. Para conocer la asignación actual de puerto en otra computadora, es necesario enviar una solicitud que pregunte algo así como "¿qué puerto está utilizando el servicio de transferencia de archivos?" La máquina objetivo responde al proporcionar el número de puerto correcto a utilizar.

Los diseñadores del TCP/IP adoptaron un enfoque híbrido que preasigna algunos números de puerto, pero que deja muchos de ellos disponibles para los sitios locales o programas de aplicación. Los números de puerto asignados comienzan con valores bajos y se extienden hacia arriba, dejando disponibles valores de números enteros altos para la asignación dinámica. En la tabla de la figura 12.6, se listan algunos de los números de puerto UDP actualmente asignados. La segunda columna contiene palabras clave asignadas como estándar de Internet y la tercera contiene palabras clave utilizadas en la mayor parte de los sistemas UNIX.

Decimal	Palabra clave	Palabra clave UNIX	Descripción
0	-	-	Reservado
7	ECHO	echo	Eco
9	DISCARD	discard	Descartar
11	USERS	systat	Usuarios Activos
13	DAYTIME	daytime	Hora del día
15	-	netstat	Quién está ahí o NETSTAT
17	*QUOTE*	*qotd*	*Cita del día*
19	CHARGEN	chargen	Generador de caracteres
37	TIME	time	Hora
42	NAMESERVER	name	Servidor de nombres de anfitriones
43	NICNAME	whois	Quién es
53	DOMAIN	nameserver	Servidor de nombres de dominios
67	BOOTPS	bootps	Servidor de protocolo bootstrap
68	BOOTPC	bootpc	Cliente de protocolo bootstrap
69	TFTP	tftp	Transferencia trivial de archivos
111	SUNRPC	sunrpc	RPC de Sun Microsystems
123	NTP	ntp	Protocolo de tiempo de red
161		snmp	monitor de red SNMP
162		snmp-trap	interrupciones SNMP
512		biff	comsat UNIX
513		who	rwho daemon UNIX
514		syslog	conexión de sistema
525		timed	daemon de hora

Figura 12.6 Ejemplo ilustrativo de los puertos UDP actualmente asignados, que muestra la palabra clave estándar y su equivalente UNIX; la lista no es completa. En lo posible, otros protocolos de transporte que ofrecen los mismos servicios utilizan los mismos números de puerto que el UDP.

12.10 Resumen

La mayor parte de los sistemas de computadoras permite que varios programas de aplicaciones se ejecuten al mismo tiempo. Utilizando la jerga de los sistemas operativos, nos referimos a dicho programa en ejecución como un *proceso*. El protocolo de datagrama de usuario, UDP, distingue entre muchos procesos dentro de una máquina al permitir que los transmisores y los receptores agreguen números enteros de 16 bits, llamados números de puerto de protocolo, a cada mensaje UDP. Los números de puerto identifican el origen y el destino. Algunos números de puerto UDP, llamados *bien conocidos*, se asignan y mencionan permanentemente a través de Internet (por ejemplo, el puerto 69 está reservado para que lo utilice el protocolo simple de transferencia de archivos, *TFTP*, que se describe en el capítulo 24). Otros números de puerto están disponibles para que los utilicen programas arbitrarios de aplicación.

El UDP es un protocolo sencillo en el sentido de que no aumenta de manera significativa la semántica del IP. Sólo proporciona a los programas de aplicación la capacidad para comunicarse, mediante el uso del servicio de entrega de paquetes, sin conexión y no confiable. Por lo tanto, los mensajes UDP se pueden perder, duplicar, retrasar o entregar en desorden; el programa de aplicación que utiliza el UDP debe resolver estos problemas. Muchos programas que emplean el UDP no funcionan correctamente en una red de redes debido a que no manejan estas condiciones.

En el esquema de estratificación por capas de protocolo, el UDP reside en la capa de transporte, arriba de la capa del Protocolo Internet y bajo la capa de aplicación. Conceptualmente, la capa de transporte es independiente de la capa Internet, pero en la práctica interactúan estrechamente. La suma de verificación UDP incluye las direcciones IP de origen y destino, lo cual significa que el software UDP debe interactuar con el software IP para encontrar direcciones antes de enviar los datagramas.

PARA CONOCER MÁS

Tanenbaum (1981) hace una comparación tutorial de los modelos de comunicación de datagrama y de circuito virtual. Ball *et. al.* (1979) describe los sistemas basados en mensajes sin tratar el protocolo de mensajes. El protocolo UDP que se describió aquí es un estándar para TCP/IP y lo define Postel (RFC 768).

EJERCICIOS

12.1 Utilice el UDP en su ambiente local. Mida la velocidad promedio de transferencia con mensajes de 256, 512, 1024, 2048, 4096, y 8192 octetos. ¿Puede explicar los resultados? Pista: ¿cuál es el MTU de su red?

12.2 ¿Por qué la suma de verificación UDP está separada de la IP? ¿Objetaría un protocolo que utilizara una sola suma de verificación para todo el datagrama IP, incluyendo el mensaje UDP?

12.3 No utilizar sumas de verificación puede ser peligroso. Explique cómo una sola difusión corrompida de paquetes ARP, realizada por la máquina *P*, puede ocasionar que sea imposible accesar otra máquina, *Q*.

12.4 ¿Se debería incorporar al IP la noción de muchos destinos identificados por puertos de protocolos? ¿Por qué?

12.5 *Registro de Nombres.* Suponga que quiere permitir que pares de programas de aplicación establezcan comunicación con el UDP, pero no les quiere asignar números fijos de puerto UDP. En vez de eso, le gustaría que las correspondencias potenciales se identificaran por medio de una cadena de 64 o menos caracteres. Por lo tanto, un programa en la máquina *A* podría querer comunicarse con el programa de "id especial curiosamente larga" en la máquina *B* (puede asumir que un proceso siempre conoce la dirección IP del anfitrión con el que se quiere comunicar). Mientras tanto, un proceso en la máquina *C* se quiere comunicar "el programa id propio de comer" en la máquina *A*. Demuestre que solamente tiene que asignar un puerto UDP para hacer posible dicha comunicación al diseñar software en cada máquina que permita: (a) que un proceso local escoja una ID no utilizada de puerto UDP sobre la cual comunicarse, (b) que un proceso local registre el nombre de 64 caracteres al que responde, y (c) que un proceso exterior utilice el UDP para establecer comunicación utilizando solamente el nombre de 64 caracteres y la dirección de red de redes de destino.

12.6 Ponga en práctica el software de registro de nombres del ejercicio anterior.

12.7 ¿Cuál es la principal ventaja de utilizar números preasignados de puerto UDP? ¿La principal desventaja?

12.8 ¿Cuál es la principal ventaja de emplear puertos de protocolo en vez de identificadores de proceso para especificar el destino dentro de una máquina?

12.9 El UDP proporciona comunicación no confiable de datagramas debido a que no garantiza la entrega del mensaje. Vislumbre un protocolo confiable de datagramas que utilice terminación de tiempo y acuses de recibo para garantizar la entrega. ¿Qué tanto retraso y trabajo adicional provoca la confiabilidad?

12.10 Envíe datagramas IP a través de una red de área amplia y mida el porcentaje de datagramas perdidos y reordenados. ¿El resultado depende de la hora del día? ¿De la carga de la red?

13

Servicio de transporte de flujo confiable (TCP)

13.1 Introducción

En los capítulos anteriores exploramos el servicio de entrega de paquetes sin conexión y no confiable, que forma la base para toda la comunicación en red de redes, así como el protocolo IP que lo define. En este capítulo, se introduce el segundo servicio más importante y mejor conocido de nivel de red, la entrega de flujo confiable, así como el *Protocolo de Control de Transmisión (TCP)* que lo define. Veremos que el TCP añade una funcionalidad substancial a los protocolos que ya hemos analizado, pero también veremos que su implantación es substancialmente más compleja.

Aunque aquí se presenta el TCP como parte del grupo de protocolos Internet TCP/IP, es un protocolo independiente de propósitos generales que se puede adaptar para utilizarlo con otros sistemas de entrega. Por ejemplo, debido a que el TCP asume muy poco sobre la red subyacente, es posible utilizarlo en una sola red como Ethernet, así como en una red de redes compleja. De hecho, el TCP es tan popular, que uno de los protocolos para sistemas abiertos de la Organización Internacional para la Estandarización, TP-4, se derivó de él.

13.2 Necesidad de la entrega de flujo

En el nivel más bajo, las redes de comunicación por computadora proporcionan una entrega de paquetes no confiable. Los paquetes se pueden perder o destruir cuando los errores de transmisión interfieren con los datos, cuando falla el hardware de red o cuando las redes se sobrecargan demasia-

do. Las redes que rutean dinámicamente los paquetes pueden entregarlos en desorden, con retraso o duplicados. Además, las tecnologías subyacentes de red pueden dictar un tamaño óptimo de paquete o formular otras obligaciones necesarias para lograr velocidades eficientes de transmisión.

En el nivel más alto, los programas de aplicación a menudo necesitan enviar grandes volúmenes de datos de una computadora a otra. Utilizar un sistema de entrega sin conexión y no confiable para las transferencias de gran volumen se vuelve tedioso, molesto y requiere que los programadores incorporen, en cada programa de aplicación, la detección y solución de errores. Debido a que es difícil diseñar, entender o modificar el software que proporciona confiabilidad, muy pocos programadores de aplicaciones tienen los antecedentes técnicos necesarios. Como consecuencia, una meta de la investigación de protocolos de red ha sido encontrar soluciones de propósito general para el problema de proporcionar una entrega de flujo confiable, lo que posibilita a los expertos a construir una sola instancia de software de protocolos de flujo que utilicen todos los programas de aplicación. Tener un solo protocolo de propósito general es útil para aislar los programas de aplicación de los detalles del trabajo con redes y permite la definición de una interfaz uniforme para el servicio de transferencia de flujo.

13.3 Características del servicio de entrega confiable

La interfaz entre los programas de aplicación y el servicio TCP/IP de entrega confiable se puede caracterizar por cinco funciones:

- *Orientación de flujo.* Cuando dos programas de aplicación (procesos de usuario) transfieren grandes volúmenes de datos, pensamos en los datos como un *flujo* de bits, divididos en *octetos* de 8 bits, que informalmente se conocen como *bytes*. El servicio de entrega de flujo en la máquina de destino pasa al receptor exactamente la misma secuencia de octetos que le pasa el transmisor en la máquina de origen.
- *Conexión de circuito virtual.* La transferencia de flujo es análoga a realizar una llamada telefónica. Antes de poder empezar la transferencia, los programas de aplicación, transmisor y receptor interactúan con sus respectivos sistemas operativos, informándose de la necesidad de realizar una transferencia de flujo. Conceptualmente, una aplicación realiza una "llamada" que la otra tiene que aceptar. Los módulos de software de protocolo en los dos sistemas operativos se comunican al enviarse mensajes a través de una red de redes, verificando que la transferencia esté autorizada y que los dos extremos estén listos. Una vez que se establecen todos los detalles, los módulos de protocolo informan a los programas de aplicación que se estableció una *conexión* y que la transferencia puede comenzar. Durante la transferencia, el software de protocolo en las dos máquinas continúa comunicándose para verificar que los datos se reciban correctamente. Si la comunicación no se logra por cualquier motivo (por ejemplo, debido a que falle el hardware de red a lo largo del camino entre las máquinas), ambas máquinas detectarán la falla y la reportarán a los programas apropiados de aplicación. Utilizamos el término *circuito virtual* para describir dichas conexiones porque aunque los programas de aplicación visualizan la conexión como un circuito dedicado de hardware, la confiabilidad que se proporciona depende del servicio de entrega de flujo.
- *Transferencia con memoria intermedia.* Los programas de aplicación envían un flujo de datos a través del circuito virtual pasando repetidamente octetos de datos al software de protocolo. Cuando transfieren datos, cada aplicación utiliza piezas del tamaño que encuentre adecuado, que

pueden ser tan pequeñas como un octeto. En el extremo receptor, el software de protocolo entrega octetos del flujo de datos en el mismo orden en que se enviaron, poniéndolos a disposición del programa de aplicación receptor tan pronto como se reciben y verifican. El software de protocolo puede dividir el flujo en paquetes, independientemente de las piezas que transfiera el programa de aplicación. Para hacer eficiente la transferencia y minimizar el tráfico de red, las implantaciones por lo general recolectan datos suficientes de un flujo para llenar un datagrama razonablemente largo antes de transmitirlo a través de una red de redes. Por lo tanto, inclusive si el programa de aplicación genera el flujo un octeto a la vez, la transferencia a través de una red de redes puede ser sumamente eficiente. De forma similar, si el programa de aplicación genera bloques de datos muy largos, el software de protocolo puede dividir cada bloque en partes más pequeñas para su transmisión.

Para aplicaciones en las que los datos se deben entregar aunque no se llene una memoria intermedia, el servicio de flujo proporciona un mecanismo de *empuje* (*push*) que las aplicaciones utilizan para forzar una transferencia. En el extremo transmisor, un empuje obliga al software de protocolo a transferir todos los datos generados sin tener que esperar a que se llene una memoria intermedia. Cuando llega al extremo receptor, el empuje hace que el TCP ponga los datos a disposición de la aplicación sin demora. Sin embargo, el lector debe notar que la función de empuje sólo garantiza que los datos se transferirán; no proporciona fronteras de registro. Por lo tanto, aun cuando la entrega es forzada, el software de protocolo puede dividir el flujo en formas inesperadas.

- *Flujo no estructurado.* Es importante entender que el servicio de flujo TCP/IP no está obligado a formar flujos estructurados de datos. Por ejemplo, no existe forma para que una aplicación de nómina haga que un servicio de flujo marque fronteras entre los registros de empleado o que identifique el contenido del flujo como datos de nómina. Los programas de aplicación que utilizan el servicio de flujo deben entender el contenido del flujo y ponerse de acuerdo sobre su formato antes de iniciar una conexión.
- *Conexión Full Duplex.* Las conexiones proporcionadas por el servicio de flujo TCP/IP permiten la transferencia concurrente en ambas direcciones. Dichas conexiones se conocen como *full duplex*. Desde el punto de vista de un proceso de aplicación, una conexión full duplex consiste en dos flujos independientes que se mueven en direcciones opuestas, sin ninguna interacción aparente. El servicio de flujo permite que un proceso de aplicación termine el flujo en una dirección mientras los datos continúan moviéndose en la otra dirección, haciendo que la conexión sea *half duplex*. La ventaja de una conexión full duplex es que el software subyacente de protocolo puede enviar en datagramas información de control de flujo al origen, llevando datos en la dirección opuesta. Este procedimiento de carga, transporte y descarga reduce el tráfico en la red.

13.4 Proporcionando confiabilidad

Hemos dicho que el servicio de entrega de flujo confiable garantiza la entrega de los datos enviados de una máquina a otra sin pérdida o duplicación. Surge la pregunta: "¿cómo puede el software de protocolo proporcionar una transferencia confiable si el sistema subyacente de comunicación sólo ofrece una entrega no confiable de paquetes?" La respuesta es complicada, pero la mayor parte de los protocolos confiables utilizan una técnica fundamental conocida como *acuse de recibo positivo con retransmisión*. La técnica requiere que un receptor se comunique con el origen y le en-

vie un mensaje de *acuse de recibo (ACK)* conforme recibe los datos. El transmisor guarda un registro de cada paquete que envía y espera un acuse de recibo antes de enviar el siguiente paquete. El transmisor también arranca un temporizador cuando envía un paquete y lo *retransmite* si dicho temporizador expira antes de que llegue un acuse de recibo.

En la figura 13.1 se muestra cómo transfiere datos el protocolo más sencillo de acuse de recibo positivo.

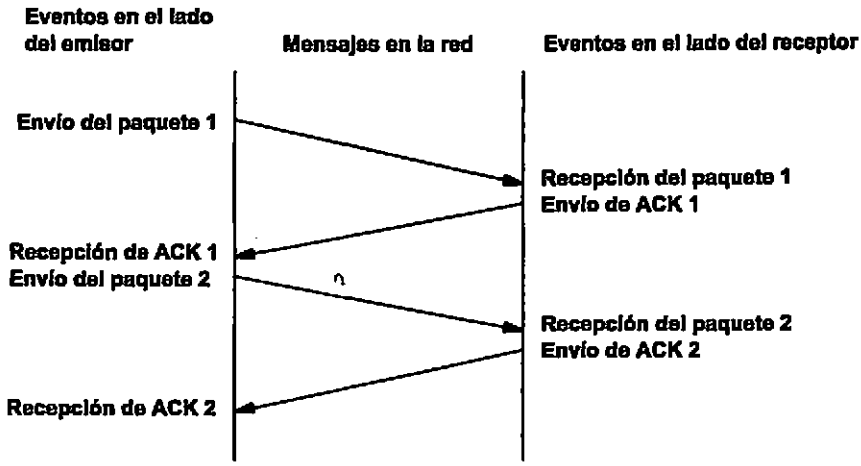


Figura 13.1 Un protocolo que se vale de reconocimientos o acuses de recibo positivos, con retransmisión, en la cual el emisor espera un acuse de recibo para cada paquete enviado. La distancia vertical bajo la figura representa el incremento en el tiempo y las líneas que cruzan en diagonal representan la transmisión de paquetes de red.

En la figura, los eventos en el transmisor y receptor se muestran a la izquierda y derecha, respectivamente. Cada línea diagonal que cruza por el centro muestra la transferencia de un mensaje a través de la red.

En la figura 13.2 se utiliza el mismo diagrama de formato que en la figura 13.1 para mostrar qué sucede cuando se pierde o corrompe un paquete. El transmisor arranca un temporizador después de enviar el paquete. Cuando termina el tiempo, el transmisor asume que el paquete se perdió y lo vuelve a enviar.

El problema final de confiabilidad surge cuando un sistema subyacente de entrega de paquetes los duplica. Los duplicados también pueden surgir cuando las redes tienen grandes retrasos que provocan la retransmisión prematura. La solución de la duplicación requiere acciones cuidadosas ya que tanto los paquetes como los acuses de recibo se pueden duplicar. Por lo general, los protocolos confiables detectan los paquetes duplicados al asignar a cada uno un número de secuencia y al obligar al receptor a recordar qué números de secuencia recibe. Para evitar la confusión causada por acuses de recibo retrasados o duplicados, los protocolos de acuses de recibo positivos envían los números de secuencia dentro de los acuses, para que el receptor pueda asociar correctamen te los acuses de recibo con los paquetes.

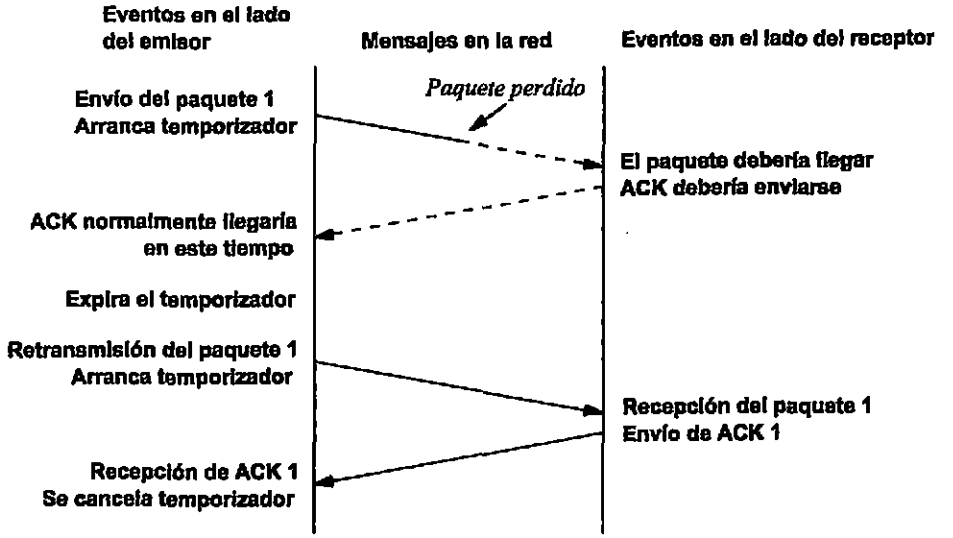


Figura 13.2 Tiempo excedido y retransmisión que ocurre cuando un paquete se pierde. La línea punteada muestra el tiempo que podría ocuparse para la transmisión de un paquete y su acuse de recibo, si no se perdiera el paquete.

13.5 La idea detrás de las ventanas deslizables

Antes de examinar el servicio de flujo TCP, necesitamos explorar un concepto adicional que sirve de base para la transmisión de flujo. Este concepto, conocido como *ventana deslizable*, hace que la transmisión de flujo sea eficiente. Para entender lo que motiva a utilizar ventanas deslizables, recuerde la secuencia de eventos que se muestran en la figura 13.1. A fin de lograr la confiabilidad, el transmisor envía un paquete y espera un acuse de recibo antes de enviar otro. Como se muestra en la figura 13.1, los datos sólo fluyen entre las máquinas en una dirección a la vez, inclusive si la red tiene capacidad para comunicación simultánea en ambas direcciones. La red estará del todo ociosa durante el tiempo en que las máquinas retrasen sus respuestas (por ejemplo, mientras las máquinas computan rutas o sumas de verificación). Si nos imaginamos una red con altos retrasos en la transmisión, el problema es evidente:

> *Un protocolo simple de acuses de recibo positivos ocupa una cantidad sustancial de ancho de banda de red debido a que debe retrasar el envío de un nuevo paquete hasta que reciba un acuse de recibo del paquete anterior.*

La técnica de ventana deslizable es una forma más compleja de acuse de recibo positivo y retransmisión que el sencillo método mencionado antes. Los protocolos de ventana deslizable utilizan el ancho de banda de red de mejor forma, ya que permiten que el transmisor envíe varios paquetes sin esperar un acuse de recibo. La manera más fácil de visualizar la operación de ventana

deslizable es pensar en una secuencia de paquetes que se transmitirán como se muestra en la figura 13.3. El protocolo coloca una *ventana* pequeña y de tamaño fijo en la secuencia, y transmite todos los paquetes que residan dentro de la ventana.

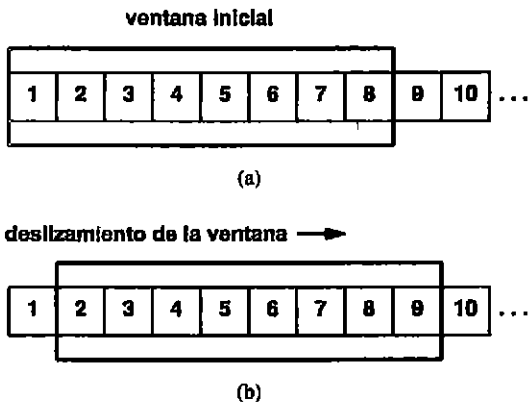


Figura 13.3 (a) Un protocolo de ventana deslizante con ocho paquetes en la ventana, y (b) La ventana que se desliza hacia el paquete *9* puede enviarse cuando se recibe un acuse de recibo del paquete *1*. Unicamente se retransmiten los paquetes sin acuse de recibo.

Decimos que un paquete es *unacknowledged* o sin acuse de recibo[1] si se transmitió pero no se recibió ningún acuse de recibo. Técnicamente, el número de paquetes sin acuse de recibo en un tiempo determinado depende del *tamaño de la ventana* y está limitado a un número pequeño y fijo. Por ejemplo, en un protocolo de ventana deslizable con un tamaño de ventana de 8, se permite al transmisor enviar 8 paquetes antes de recibir un acuse de recibo.

Como se muestra en la figura 13.3, una vez que el transmisor recibe un acuse de recibo para el primer paquete dentro de la ventana, "mueve" la misma y envía el siguiente paquete. La ventana continuará moviéndose en tanto se reciban acuses de recibo.

El desempeño de los protocolos de ventana deslizable depende del tamaño de la ventana y de la velocidad en que la red acepta paquetes. En la figura 13.4, se muestra un ejemplo de la operación de un protocolo de ventana deslizable cuando se envían tres paquetes. Nótese que el transmisor los envía antes de recibir cualquier acuse de recibo.

Con un tamaño de ventana de *1*, un protocolo de ventana deslizable sería idéntico a un protocolo simple de acuse de recibo positivo. Al aumentar el tamaño de la ventana, es posible eliminar completamente el tiempo ocioso de la red. Esto es, en una situación estable, el transmisor puede enviar paquetes tan rápido como la red los pueda transferir. El punto principal es que:

[1] N del T: no confirmado o paquete del cual no se recibió acuse de recibo.

Como un protocolo de ventana deslizable bien establecido mantiene la red completamente saturada de paquetes, con él se obtiene una generación de salida substancialmente más alta que con un protocolo simple de acuse de recibo positivo.

Conceptualmente, un protocolo de ventana deslizable siempre recuerda qué paquetes tienen acuse de recibo y mantiene un temporizador separado para cada paquete sin acuse de recibo. Si se pierde un paquete, el temporizador concluye y el transmisor reenvía el paquete. Cuando el receptor desliza su ventana, mueve hacia atrás todos los paquetes con acuse. En el extremo receptor, el software de protocolo mantiene una ventana análoga, que acepta y acusa como recibidos los paquetes conforme llegan. Por lo tanto, la ventana divide la secuencia de paquetes en tres partes: los paquetes a la izquierda de la ventana se transmitieron, recibieron y acusaron exitosamente; los paquetes a la derecha no se han transmitido; y los paquetes que quedan dentro de la ventana están en proceso de transmisión. El paquete con menor número en la ventana es el primer paquete en la secuencia para el que no se ha hecho un acuse de recibo.

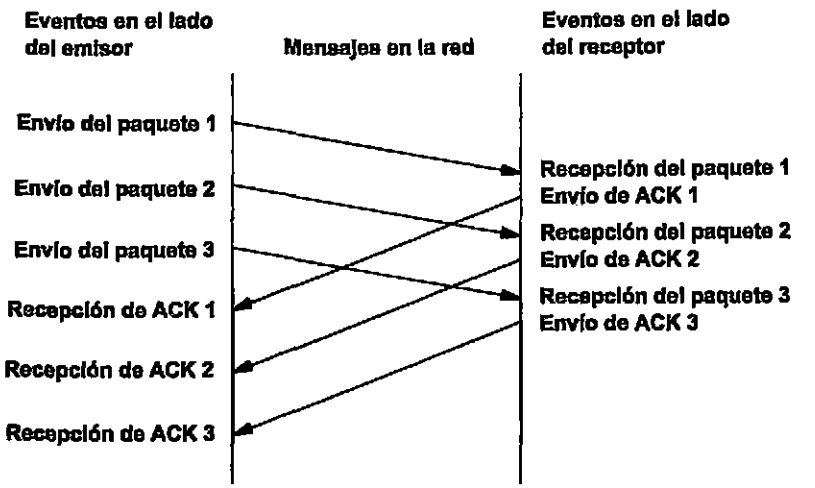


Figura 13.4 Ejemplo de tres paquetes transmitidos mediante un protocolo de ventana deslizante. El concepto clave es que el emisor puede transmitir todos los paquetes de la ventana sin esperar un acuse de recibo.

13.6 Protocolo de control de transmisión

Ya que entendimos el principio de las ventanas deslizables, podemos examinar el servicio de flujo confiable proporcionado por el grupo de protocolos TCP/IP de Internet. El servicio lo define el *Protocolo de Control de Transmisión* o TCP. El servicio de flujo confiable es tan importante que todo el grupo de protocolos se conoce como TCP/IP. Es importante entender que:

El TCP es un protocolo de comunicación, no una pieza de software.

La diferencia entre un protocolo y el software que lo implementa es análoga a la diferencia entre la definición de un lenguaje de programación y un compilador. Al igual que en el mundo de los lenguajes de programación, la distinción entre definición e implantación a veces es imprecisa. Las personas encuentran software TCP mucho más frecuentemente que la especificación de protocolo, así que es natural pensar en una implantación en particular como en el estándar. No obstante, el lector debe tratar de distinguir entre las dos.

¿Qué proporciona el TCP exactamente? El TCP es complejo, por lo que no hay una respuesta sencilla. El protocolo especifica el formato de datos y los acuses de recibo que intercambian dos computadoras para lograr una transferencia confiable, así como los procedimientos que la computadora utiliza para asegurarse de que los datos lleguen de manera correcta. También, especifica cómo el software TCP distingue el correcto entre muchos destinos en una misma máquina, y cómo las máquinas en comunicación resuelven errores como la pérdida o duplicación de paquetes. El protocolo también especifica cómo dos computadoras inician una transferencia de flujo TCP y cómo se ponen de acuerdo cuando se completa.

Asimismo, es importante entender lo que el protocolo no incluye. Aunque la especificación TCP describe cómo utilizan el TCP los programas de aplicación en términos generales, no aclara los detalles de la interfaz entre un programa de aplicación y el TCP. Esto es, la documentación del protocolo sólo analiza las operaciones que el TCP proporciona; no especifica los procedimientos exactos que los programas de aplicación invocan para accesar estas operaciones. La razón para no especificar la interfaz del programa de aplicación es la flexibilidad. En particular, debido a que los programadores por lo general implantan el TCP en el sistema operativo de una computadora, necesitan emplear la interfaz que proporciona el sistema operativo, sea cual sea. Permitir que el implantador tenga flexibilidad hace posible tener una sola especificación para el TCP que pueda utilizarse para diseñar software en una gran variedad de máquinas.

Debido a que TCP asume muy poco sobre el sistema subyacente de comunicación, TCP se puede utilizar con una gran variedad de sistemas de entrega de paquetes, incluyendo el servicio de entrega de datagramas IP. Por ejemplo, el TCP puede implantarse para utilizar líneas de marcación telefónica, una red de área local, una red de fibra óptica de alta velocidad o una red de largo recorrido y baja velocidad. De hecho, la gran variedad de sistemas de entrega que puede utilizar el TCP es una de sus ventajas.

13.7 Puertos, conexiones y puntos extremos

Al igual que el Protocolo de Datagrama de Usuario (UDP) que vimos en el capítulo 12, el TCP reside sobre el IP en el esquema de estratificación por capas de protocolos. En la figura 13.5 se muestra la organización conceptual. El TCP permite que varios programas de aplicación en una máquina se comuniquen de manera concurrente y realiza el demultiplexado del tráfico TCP entrante entre los programas de aplicación. Al igual que el Protocolo de Datagrama de Usuario (UDP), el TCP utiliza números de *puerto de protocolo* para identificar el destino final dentro de una máquina. Cada puerto tiene asignado un número entero pequeño utilizado para identificarlo.[2]

[2] Aun cuando el TCP y el UDP se valen de identificadores enteros de puerto que comienzan en 1 para identificar puertos, no hay confusión entre ellos ya que un datagrama IP entrante identifica el protocolo en uso así como en número de puerto.

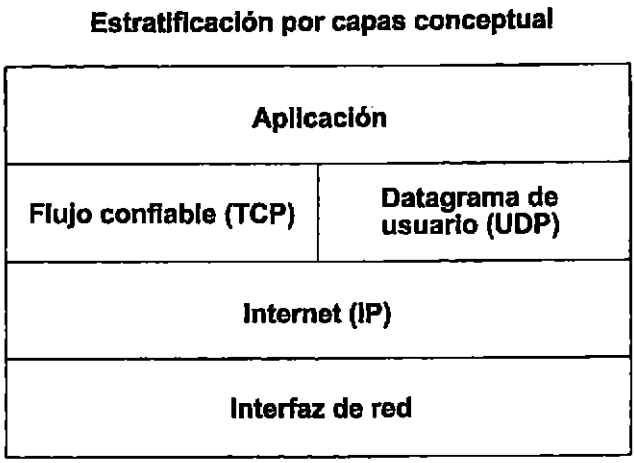


Figura 13.5 La estratificación por capas conceptual del UDP y el TCP sobre el IP. El TCP proporciona un servicio de flujo confiable, mientras que el UDP proporciona un servicio de entrega de datagramas no confiable. Los programas de aplicación emplean ambos.

Cuando tratamos los puertos UDP, dijimos que se pensara de cada puerto como en una cola de salida en la que el software de protocolo coloca los datagramas entrantes. Sin embargo, los puertos TCP son mucho más complejos, ya que un número de puerto no corresponde a un solo objeto. De hecho, el TCP se diseñó según la *abstracción de conexión*, en la que los objetos que se van a identificar son conexiones de circuito virtual, no puertos individuales. Entender que el TCP utiliza la noción de conexiones es crucial, ya que nos ayuda a explicar el significado y la utilización de los números de puerto TCP:

El TCP utiliza la conexión, no el puerto de protocolo, como su abstracción fundamental; las conexiones se identifican por medio de un par de puntos extremos.

¿Qué son exactamente los "puntos extremos" de una conexión? Hemos dicho que una conexión consiste en un circuito virtual entre dos programas de aplicación, por lo que puede ser natural asumir que un programa de aplicación sirve como el "punto extremo" de la conexión. Sin embargo, no es así. El TCP define que *un punto* extremo es un par de números enteros (*anfitrión, puerto*), en donde *anfitrión* es la dirección IP de un anfitrión y *puerto* es un puerto TCP en dicho anfitrión. Por ejemplo, el punto extremo (*128.10.2.3, 25*), se refiere al puerto TCP *25* en la máquina con dirección IP *128.10.2.3*.

Ahora que ya explicamos los puntos extremos, será fácil entender las conexiones. Recuerde que una conexión se define por sus dos puntos extremos. Por lo tanto, si existe una conexión entre la máquina (*18.26.0.36*) en el MIT y la máquina (*128.10.2.3*) en la Universidad de Purdue, la conexión se definiría por los puntos extremos:

(*18.26.0.36, 1069*) y (*128.10.2.3, 25*).

Mientras tanto, otra conexión se puede dar entre la máquina (*128.9.0.32*) en el Instituto de Ciencias de la Información y la misma máquina en Purdue, conexión identificada por sus puntos extremos:

(*128.9.0.32, 1184*) y (*128.10.2.3, 53*).

Hasta ahora, nuestros ejemplos de conexiones han sido directos, ya que los puertos utilizados en todos los puntos extremos han sido únicos. Sin embargo, la abstracción de conexión permite que varias conexiones compartan un punto extremo. Por ejemplo, podemos agregar otra conexión a las dos arriba mencionadas entre la máquina (*128.2.254.139*) en la CMU y la máquina en Purdue:

(*128.2.254.139, 1184*) y (*128.10.2.3, 53*).

Puede parecer extraño que dos conexiones utilicen al mismo tiempo el puerto TCP *53* en la máquina 128.10.2.3, pero no hay ambigüedad. Debido a que el TCP asocia los mensajes entrantes con una conexión en vez de hacerlo con un puerto de protocolo, utiliza ambos puntos extremos para identificar la conexión apropiada. La idea importante que se debe recordar es:

> *Como el TCP identifica una conexión por medio de un par de puntos extremos, varias conexiones en la misma máquina pueden compartir un número de puerto TCP.*

Desde el punto de vista de un programador, la abstracción de comunicación es importante. Significa que un programador puede diseñar un programa que proporcione servicio concurrente a varias conexiones al mismo tiempo, sin necesitar números únicos de puerto local para cada una. Por ejemplo, la mayor parte de los sistemas proporciona acceso concurrente a su servicio de correo electrónico, lo cual permite que varias computadoras les envíen correo electrónico de manera concurrente. Debido a que el programa que acepta correo entrante utiliza el TCP para comunicarse, sólo necesitan emplear un puerto TCP local, aun cuando permita que varias conexiones se realicen en forma concurrente.

13.8 Aperturas pasivas y activas

A diferencia del UDP, el TCP es un protocolo orientado a la conexión, el cual requiere que ambos puntos extremos estén de acuerdo en participar. Esto es, antes de que el tráfico TCP pueda pasar a través de una red de redes, los programas de aplicación en ambos extremos de la conexión deben estar de acuerdo en que desean dicha conexión. Para hacerlo, el programa de aplicación en un extremo realiza una función de *apertura pasiva* al contactar su sistema operativo e indicar que aceptará una conexión entrante. En ese momento, el sistema operativo asigna un número de puerto TCP a su extremo de la conexión. El programa de aplicación en el otro extremo debe contactar a su sistema operativo mediante una solicitud de *apertura activa* para establecer una conexión. Los dos módulos de software TCP se comunican para establecer y llevar a cabo la conexión. Una vez que

se crea ésta, los programas de aplicación pueden comenzar a transferir datos; los módulos de software TCP en cada extremo intercambian mensajes que garantizan la entrega confiable. Regresaremos a los detalles del establecimiento de conexiones después de examinar el formato de mensaje TCP.

13.9 Segmentos, flujos y números de secuencia

El TCP visualiza el flujo de datos como una secuencia de octetos (o bytes) que divide en *segmentos* para su transmisión. Por lo general, cada segmento viaja a través de una red de redes como un solo datagrama IP.

El TCP utiliza un mecanismo especializado de ventana deslizable para solucionar dos problemas importantes: la transmisión eficiente y el control de flujo. Al igual que el protocolo de ventana deslizable descrito anteriormente, el mecanismo de ventana del TCP hace posible enviar varios segmentos antes de que llegue un acuse de recibo. Hacerlo así aumenta la generación total de salida ya que mantiene ocupada a la red. La forma TCP de un protocolo de ventana deslizable también soluciona el problema de *control de flujo* de extremo a extremo, al permitir que el receptor restrinja la transmisión hasta que tenga espacio suficiente en memoria intermedia para incorporar más datos.

El mecanismo TCP de ventana deslizable opera a nivel de octeto, no a nivel de segmento ni de paquete. Los octetos del flujo de datos se numeran de manera secuencial, y el transmisor guarda tres apuntadores asociados con cada conexión. Los apuntadores definen una ventana deslizable, como se muestra en la figura 13.6. El primer apuntador marca el extremo izquierdo de la ventana deslizable, separa los octetos que ya se enviaron y envía el acuse de recibo de los octetos ya enviados. Un segundo apuntador marca el extremo derecho de la ventana deslizable y define el octeto más alto en la secuencia que se puede enviar antes de recibir más acuses de recibo. El tercer apuntador señala la frontera dentro de la ventana que separa los octetos que ya se enviaron de los que todavía no se envían. El software de protocolo envía sin retraso todos los octetos dentro de la ven-

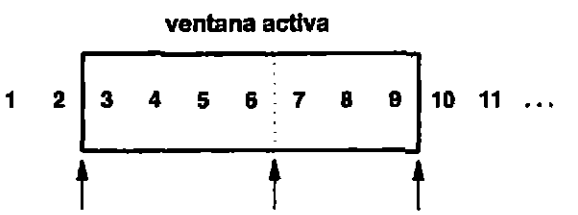


Figura 13.6 Ejemplo de la ventana deslizante del TCP. Los octetos hasta el dos se han enviado y reconocido, los octetos del 3 al 6 han sido enviados pero no reconocidos, los octetos del 7 al 9 no se han enviado pero serán enviados sin retardo y los octetos del 10 en adelante no pueden ser enviados hasta que la ventana se mueva.

tana, por lo que en general, la frontera dentro de la ventana se mueve rápidamente de izquierda a derecha.

Hemos descrito cómo la ventana TCP del transmisor se desliza y hemos mencionado que el receptor debe tener una ventana similar para ensamblar de nuevo el flujo. Sin embargo, es importante entender que, como las conexiones TCP son de tipo full duplex, se llevan a cabo dos transferencias al mismo tiempo en cada conexión, una en cada dirección. Pensamos en las transferencias como en algo totalmente independientes porque, en cualquier momento, los datos pueden fluir a través de la conexión en una o en ambas direcciones. Por lo tanto, el software TCP en cada extremo mantiene dos ventanas por cada conexión (un total de cuatro), una se desliza a lo largo del flujo de datos que se envía, mientras la otra se desliza a lo largo de los datos que se reciben.

13.10 Tamaño variable de ventana y control de flujo

Una diferencia entre el protocolo TCP de ventana deslizable y el protocolo simplificado de ventana deslizable, presentado anteriormente, es que el TCP permite que el tamaño de la ventana varíe. Cada acuse de recibo, que informa cuántos octetos se recibieron, contiene un *aviso de ventana*, que especifica cuántos octetos adicionales de datos está preparado para aceptar el receptor. Pensemos el aviso de ventana como la especificación del tamaño actual de la memoria intermedia del receptor. En respuesta a un aumento en el aviso de ventana, el transmisor aumenta el tamaño de su ventana deslizable y procede al envío de octetos de los que todavía no se tiene un acuse de recibo. En respuesta a una disminución en el aviso de ventana, el transmisor disminuye el tamaño de su ventana y deja de enviar los octetos que se encuentran más allá de la frontera. El software TCP no debe contradecir los anuncios previos, reduciendo la posición aceptable de la ventana, que pasó anteriormente, en flujo de octetos. De hecho, los anuncios más pequeños acompañan a los acuses de recibo, así que el tamaño de la ventana cambia en el momento que se mueve hacia adelante.

La ventaja de utilizar una ventana de tamaño variable es que ésta proporciona control de flujo así como una transferencia confiable. Si la memoria intermedia del receptor se llena, no puede aceptar más paquetes, así que envía un anuncio de ventana más pequeño. En caso extremo, el receptor anuncia un tamaño de ventana igual a cero para detener toda la transmisión. Después, cuando hay memoria intermedia disponible, el receptor anuncia un tamaño de ventana distinto a cero para activar de nuevo el flujo de datos.[3]

Tener un mecanismo para el flujo de datos es esencial en un ambiente de red de redes, en donde las máquinas de varias velocidades y tamaños se comunican a través de redes y ruteadores de varias velocidades y capacidades. En realidad, existen dos problemas independientes de flujo. Primero, los protocolos de red de redes necesitan un control de flujo extremo a extremo entre la fuente y el destino final. Por ejemplo, cuando se comunican una minicomputadora y un gran mainframe, ambos necesitan regular la entrada de datos, o el software de protocolo se sobrecargaría rápidamente. Por lo tanto, el TCP debe implantar el control de flujo extremo a extremo para garantizar una entrega confiable. Segundo, los protocolos de red de redes necesitan un mecanismo de con-

[3] Hay dos excepciones para la transmisión cuando el tamaño de la ventana es cero. Primero, cuando un emisor está autorizado a transmitir un segmento con el bit de urgente activado para informar al receptor que está disponible un dato urgente. Segundo, para evitar un fin de cronometrado potencial si un anuncio diferente de cero se pierde luego de que el tamaño de la ventana llega a cero, el emisor prueba una ventana de tamaño cero periódicamente.

trol de flujo que permita que los sistemas intermedios (por ejemplo, los ruteadores) controlen una fuente que envíe más tráfico del que la máquina puede tolerar.

La sobrecarga de las máquinas intermedias se conoce como *congestionamiento* y los mecanismos que resuelven el problema se conocen como mecanismos de *control de congestionamiento*. El TCP emplea su esquema de ventana deslizable para resolver el problema de control de flujo extremo a extremo; no cuenta con un mecanismo explícito para el control de congestionamientos. Sin embargo, más adelante, veremos que una implantación TCP cuidadosamente programada puede detectar y resolver los congestionamientos, así como una implantación descuidada puede empeorarlos. En particular, aunque un esquema de retransmisión cuidadosamente seleccionado puede ser útil para evitar el congestionamiento, uno mal elegido puede empeorarlo.

13.11 Formato del segmento TCP

La unidad de transferencia entre el software TCP de dos máquinas se conoce como *segmento*. Los segmentos se intercambian para establecer conexiones, transferir datos, enviar acuses de recibo, anunciar los tamaños de ventanas y para cerrar conexiones. Debido a que el TCP utiliza acuses de recibo incorporados, un acuse que viaja de la máquina *A* a la máquina *B* puede viajar en el mismo segmento en el que viajan los datos de la máquina *A* a la máquina *B*, aun cuando el acuse de recibo se refiera a los datos enviados de *B* hacia *A*.[4] En la figura 13.7 se muestra el formato del segmento TCP.

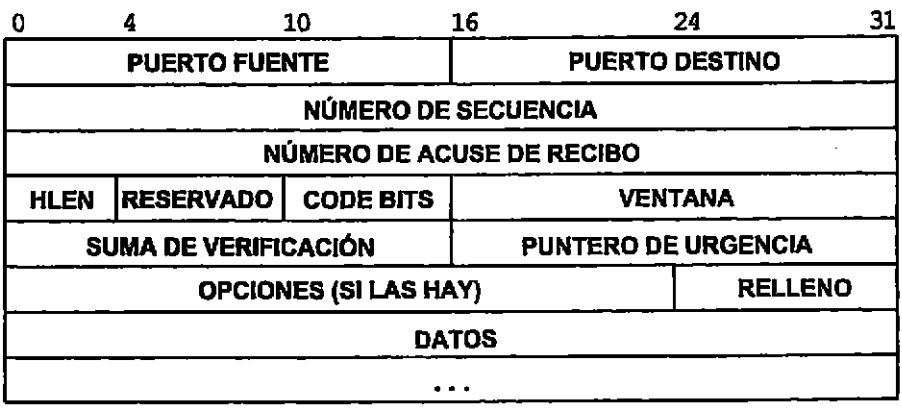


Figura 13.7 Formato de un segmento TCP con un encabezado TCP seguido de datos. Los segmentos se utilizan para establecer conexiones, así como para transportar datos y acuses de recibo.

[4] En la práctica este tipo de incorporación no se presenta con frecuencia ya que la mayor parte de las aplicaciones no envía datos en ambas direcciones de manera simultánea.

Cada segmento se divide en dos partes: encabezado y datos. El encabezado, conocido como *encabezado TCP*, transporta la identificación y la información de control. Los campos *SOURCE PORT* (*PUERTO FUENTE*) y *DESTINATION PORT* (*PUERTO DESTINO*) contienen los números de puerto TCP que identifican a los programas de aplicación en los extremos de la conexión. El campo *SEQUENCE NUMBER* (*NÚMERO DE SECUENCIA*) identifica la posición de los datos del segmento en el flujo de datos del transmisor. El campo *ACKNOWLEDGEMENT NUMBER* (*NÚMERO DE ACUSE DE RECIBO*) identifica el número de octetos que la fuente espera recibir después. Observe que el número de secuencia se refiere al flujo que va en la misma dirección que el segmento, mientras que el número de acuse de recibo se refiere al flujo que va en la dirección opuesta al segmento.

El campo *HLEN*[5] contiene un número entero que especifica la longitud del encabezado del segmento, medida en múltiplos de 32 bits. Es necesario porque el campo *OPTIONS* (*OPCIONES*) varía en su longitud, dependiendo en qué opciones se haya incluido. Así, el tamaño del encabezado TCP varía dependiendo de las opciones seleccionadas. El campo de 6 bits marcado como *RESERVED* (*RESERVADO*), está reservado para usarse en el futuro.

Algunos segmentos sólo llevan un acuse de recibo y otros solamente llevan datos. Otros llevan solicitudes para establecer o cerrar una conexión. El software TCP utiliza el campo de 6 bits, etiquetado como *CODE BITS*, para determinar el propósito y contenido del segmento. Los seis bits indican cómo interpretar otros campos en el encabezado, de acuerdo con la tabla en la figura 13.8.

Bit (de izquierda a derecha)	Significado si el bit está puesto a 1
URG	El campo de puntero de urgente es válido
ACK	El campo de acuse de recibo es válido
PSH	Este segmento solicita una operación push
RST	Iniciación de la conexión
SYN	Sincronizar números de secuencia
FIN	El emisor ha llegado al final de su flujo de octetos

Figura 13.8 Bits del campo CODE en el encabezado TCP.

El software TCP informa sobre cuántos datos está dispuesto a aceptar cada vez que envía un segmento, al especificar su tamaño de memoria intermedia en el campo *WINDOW*. El campo contiene un número entero sin signo de 16 bits en el orden de octetos estándar de red. Los anuncios de ventana proporcionan otro ejemplo de acuse de recibo de carga, transporte y descarga ya que acompañan a todos los segmentos, tanto a los que llevan datos, como a los que sólo llevan un acuse de recibo.

[5] La especificación indica que el campo *HLEN* es el desplazamiento del área de datos dentro del segmento.

13.12 Datos fuera de banda

Aunque el TCP es un protocolo orientado al flujo, algunas veces es importante que el programa en un extremo de la conexión envíe datos *fuera de banda*, sin esperar a que el programa en el otro extremo de la conexión consuma los octetos que ya están en flujo. Por ejemplo, cuando se utiliza el TCP para una sesión de acceso remoto, el usuario puede decidir si envía una secuencia de teclado que *interrumpa* o *aborte* el programa en el otro extremo. Dichas señales se necesitan aun más cuando un programa en la máquina remota no opera de manera correcta. Las señales se deben enviar sin esperar a que el programa lea los octetos que ya están en el flujo TCP (o no sería posible interrumpir programas que dejen de leer la entrada).

Para incorporar la señalización fuera de banda, el TCP permite que el transmisor especifique los datos como *urgentes*, dando a entender que se debe notificar su llegada al programa receptor tan pronto como sea posible, sin importar su posición en el flujo. El protocolo especifica que, cuando se encuentra con datos urgentes, el TCP receptor debe notificar al programa de aplicación, que esté asociado con la conexión, que entre en "modalidad urgente". Después de asimilar todos los datos urgentes, el TCP indica al programa de aplicación que regrese a su operación normal.

Por supuesto, los detalles exactos de cómo el TCP informa al programa de aplicación sobre datos urgentes dependen del sistema operativo de la máquina. El mecanismo utilizado para marcar los datos urgentes cuando se transmiten en un segmento consiste en un bit de código URG y en un campo *URGENT POINTER* (*PUNTERO DE URGENCIA*). Cuando se activa el bit URG, el indicador urgente especifica la posición dentro del segmento en la que terminan los datos urgentes.

13.13 Opción de tamaño máximo de segmento

No todos los segmentos que se envían a través de una conexión serán del mismo tamaño. Sin embargo, ambos extremos necesitan acordar el tamaño máximo de los segmentos que transferirán. El software TCP utiliza el campo *OPTIONS* para negociar con el software TCP en el otro extremo de la conexión; una de las opciones permite que el software TCP especifique el *tamaño máximo de segmento (MSS)* que está dispuesto a recibir. Por ejemplo, cuando un sistema incorporado que solamente tiene unos cuantos cientos de octetos de memoria intermedia se conecta con una gran supercomputadora, puede negociar un MSS que restrinja los segmentos para que quepan en la memoria intermedia. Para las computadoras conectadas por redes de área local de alta velocidad es especialmente importante escoger un tamaño máximo de segmento que llene los paquetes o no harán un buen uso del ancho de banda. Por lo tanto, si los dos puntos extremos residen en la misma red física, el TCP por lo general computará un tamaño máximo de segmento de tal forma que los datagramas IP resultantes correspondan con la MTU de la red. Si los puntos extremos no residen en la misma red física, pueden intentar descubrir la MTU mínima a lo largo del camino entre ellos o pueden escoger un tamaño máximo de segmento de *536* (tamaño máximo asignado por omisión de un datagrama IP, *576*, menos el tamaño estándar de los encabezados IP y TCP).

En un ambiente general de red de redes, escoger un tamaño máximo de segmento apropiado puede ser difícil, ya que el desempeño puede ser bajo tanto por tamaños de segmento demasiado grandes, como por tamaños muy pequeños. Por una parte, cuando el tamaño del segmento es pequeño, la utilización de la red permanece baja. Para entender por qué, recuerde que los segmentos

TCP viajan encapsulados dentro de datagramas IP, que a su vez están encapsulados en tramas de red física. Por lo tanto, cada segmento tiene al menos 40 octetos de encabezados TCP e IP, además de los datos. Así pues, los datagramas que sólo llevan un octeto de datos utilizan como máximo 1/41 del ancho de banda de la red subyacente para los datos de usuario; en la práctica, las brechas mínimas entre paquetes y el hardware de red que ponen bits en tramas hacen que el rango sea aún más pequeño.

Por otro lado, los tamaños de segmento muy grandes también pueden producir un bajo desempeño. Los grandes segmentos resultan en grandes datagramas IP. Cuando dichos datagramas viajan a través de una red con una MTU pequeña, el IP debe fragmentarlos. A diferencia de un segmento TCP, un fragmento no se puede confirmar o retransmitir en forma independiente; todos los fragmentos deben llegar o de lo contrario se tendrá que retransmitir todo el datagrama. Debido a que la probabilidad de perder un fragmento no es de cero, aumentar el tamaño de segmento por arriba del umbral de fragmentación, disminuye la probabilidad de que lleguen los datagramas, lo que disminuye la producción de salida.

En teoría, el tamaño óptimo de segmento, *S*, ocurre cuando los datagramas IP que llevan los segmentos son tan grandes como sea posible sin requerir fragmentación en ninguna parte a lo largo del camino entre la fuente y el destino. En la práctica, encontrar *S* es difícil por muchas razones. Primero, la mayor parte de las implantaciones de TCP no incluye un mecanismo para hacerlo. Segundo, debido a que los ruteadores en una red de redes pueden cambiar las rutas en forma dinámica, el camino que siguen los datagramas entre un par de computadoras en comunicación puede cambiar también de manera dinámica, así como también puede cambiar el tamaño en que se tienen que fragmentar los datagramas. Tercero, el tamaño óptimo depende de los encabezados de protocolos de nivel más bajo (por ejemplo, el tamaño del segmento se debe reducir para incorporar opciones IP). La investigación sobre el problema de encontrar un tamaño óptimo de segmento continúa.

13.14 Cómputo de suma de verificación TCP

El campo *CHECKSUM* (*VERIFICACIÓN DE SUMA*) en el encabezado TCP contiene una suma de verificación de números enteros y 16 bits que se utiliza para verificar la integridad de los datos así como del encabezado TCP. Para computar la suma de verificación, el software TCP en la máquina transmisora sigue un procedimiento igual al descrito en el capítulo 12 para UDP. Coloca un *pseudo-encabezado* en el segmento, agrega suficientes bits en cero para lograr que el segmento sea un múltiplo de 16 bits y calcula la suma de 16 bits sobre todo el resultado. El TCP no cuenta el pseudo-encabezado ni los caracteres de relleno en la longitud del segmento, ni tampoco los transmite. También, asume que el campo de suma de verificación por sí mismo es de cero, para propósitos de la suma. Como en el caso de otras sumas de verificación, el TCP utiliza aritmética de 16 bits y toma el complemento a uno del complemento a uno de la suma. En la localidad receptora, el software TCP realiza el mismo cómputo para verificar que el segmento llega intacto.

El propósito de utilizar un pseudo-encabezado es exactamente el mismo que en el UDP. Permite que el receptor verifique que el segmento llegó a su destino correcto, que incluye tanto una dirección IP de anfitrión como un número de puerto de protocolo. Tanto la dirección IP de origen como la de destino son importantes para el TCP, ya que debe utilizarlas para identificar una conexión a la que pertenece el segmento. Así, cada vez que llega un datagrama que transporta un seg-

mento TCP, el IP debe pasar al TCP las direcciones IP de origen y destino, así como el segmento mismo. En la figura 13.9, se muestra el formato del pseudo-encabezado empleado en el cómputo de la suma de verificación.

0	8	16	31
DIRECCIÓN IP DE LA FUENTE			
DIRECCIÓN IP DEL DESTINO			
CERO	PROTOCOLO	LONGITUD TCP	

Figura 13.9 Formato del pseudo-encabezado utilizado en el cálculo de la suma de verificación del TCP. En la localidad receptora, esta información se extrae del datagrama IP que transportaba el segmento.

El TCP transmisor asigna al campo *PROTOCOL* (*PROTOCOLO*) el valor que utilizará el sistema subyacente de entrega en su campo de tipo de protocolo. Para los datagramas IP que transporten TCP, el valor es 6. El campo *TCP LENGHT* (*LONGITUD TCP*) especifica la longitud total del segmento TCP, incluyendo el encabezado TCP. En el extremo receptor, la información utilizada en el pseudo-encabezado se extrae del datagrama IP que transportó el segmento y se incluye en el cómputo de la suma para verificar que el segmento llegó intacto al destino correcto.

13.15 Acuses de recibo y retransmisión

Como el TCP envía los datos en segmentos de longitud variable, y debido a que los segmentos retransmitidos pueden incluir más datos que los originales, los acuses de recibo no pueden remitirse fácilmente a los datagramas o segmentos. De hecho, se remiten a una posición en el flujo, utilizando los números de secuencia de flujo. El receptor recolecta octetos de datos de los segmentos entrantes y reconstruye una copia exacta del flujo que se envía. Como los segmentos viajan en datagramas IP, se pueden perder o llegar en desorden; el receptor utiliza los números de secuencia para reordenar los segmentos. En cualquier momento, el receptor tendrá cero o más octetos reconstruidos contiguamente desde el comienzo del flujo, pero puede tener piezas adicionales del flujo de datagramas que hayan llegado en desorden. El receptor siempre acusa recibo del prefijo contiguo más largo del flujo que se recibió correctamente. Cada acuse de recibo especifica un valor de secuencia mayor en una unidad, con respecto al octeto de la posición más alta en el prefijo contiguo que recibió. Por lo tanto, el transmisor recibe una retroalimentación continua del receptor conforme progresa el flujo. Podemos resumir esta idea importante de la siguiente manera:

Un acuse de recibo TCP especifica el número de secuencia del siguiente octeto que el receptor espera recibir.

Al esquema TCP de acuse de recibo se le llama *acumulativo* porque reporta cuánto se ha acumulado del flujo. Los acuses de recibo acumulativos tienen ventajas y desventajas. Una ventaja es que los acuses de recibo son fáciles de generar y no son ambiguos. Otra es que los acuses de recibo perdidos no necesariamente forzarán la retransmisión. Una gran desventaja es que el receptor no obtiene información sobre todas las transmisiones exitosas, sino únicamente sobre una sola posición en el flujo que se recibió.

Para entender por qué la falta de información sobre todas las transmisiones exitosas hace que los acuses de recibo acumulativos sean menos eficientes, piense en una ventana que abarca *5000* octetos comenzando en la posición *101* en el flujo, y suponga que el transmisor envió todos los datos en la ventana al transmitir cinco segmentos. Suponga también que se pierde el primer segmento y que todos los demás llegan intactos. Conforme llega cada segmento, el receptor envía un acuse de recibo, pero todos los acuses especifican el octeto *101*, que es el octeto contiguo siguiente más alto que espera recibir. No hay forma para que el receptor indique al transmisor que llegó la mayor parte de los datos para la ventana actual.

Cuando ocurre una terminación de tiempo en el extremo transmisor, éste debe escoger entre dos esquemas potencialmente ineficaces. Puede retransmitir un segmento o retransmitir los cinco. En este caso, retransmitir los cinco segmentos no es eficaz. Cuando llega el primer segmento, el receptor tendrá todos los datos en la ventana, y en el acuse de recibo aparecerá *5101*. Si el transmisor sigue el estándar aceptado y retransmite sólo el primer segmento para el que no hay acuse, debe esperar a obtener el acuse de recibo antes de decidir qué y cuántos datos enviar. Por lo tanto, regresa a un protocolo simple de acuse de recibo positivo y puede perder las ventajas de tener una gran ventana.

13.16 Tiempo límite y retransmisión

Una de las ideas más importantes y complejas del TCP es parte de la forma en que maneja la terminación de tiempo (*time out*) y la retransmisión. Al igual que otros protocolos confiables, el TCP espera que el destino envíe acuses de recibo siempre que recibe exitosamente nuevos octetos del flujo de datos. Cada vez que envía un segmento, el TCP arranca un temporizador y espera un acuse de recibo. Si se termina el tiempo antes de que se acusen de recibidos los datos en el segmento, el TCP asume que dicho segmento se perdió o corrompió y lo retransmite.

Para entender por qué el algoritmo TCP de retransmisión difiere del algoritmo utilizado en muchos protocolos de red, necesitamos recordar que el TCP está diseñado para emplearse en un ambiente de red de redes. En una red de redes, un segmento que viaja entre dos máquinas puede atravesar una sola red de poco retraso (por ejemplo, una LAN de alta velocidad) o puede viajar a través de varias redes intermedias y de varios ruteadores. Por lo tanto, es imposible saber con anticipación qué tan rápido regresarán los acuses de recibo al origen. Además, el retraso en cada ruteador depende del tráfico, por lo que el tiempo total necesario para que un segmento viaje al destino y para que un acuse de recibo regrese al origen varía dramáticamente de un ejemplo a otro. En la figura 13.10, en la que se muestran medidas de tiempos de viaje redondo a través de la red Internet global para 100 paquetes consecutivos, se ilustra el problema. El software TCP debe incorporar las amplias diferencias en el tiempo necesario para llegar a varios destinos, así como los cambios en el tiempo necesario para llegar a cierto destino conforme varía la carga de tráfico.

El TCP maneja los retrasos variables en la red de redes al utilizar un *algoritmo adaptable de retransmisión*. En esencia, el TCP monitorea el desempeño de cada conexión y deduce valores razonables para la terminación de tiempo. Conforme cambia el desempeño de una conexión, el TCP revisa su valor de terminación de tiempo (por ejemplo, se adapta al cambio).

Para recolectar los datos necesarios para un algoritmo adaptable, el TCP registra la hora en la que se envía cada segmento y la hora en la que se recibe un acuse de recibo para los datos en el segmento. Considerando las dos horas, el TCP computa el tiempo transcurrido, conocido como *tiempo ejemplo de viaje redondo* o *ejemplo de viaje redondo*. Siempre que obtiene un nuevo ejemplo de viaje redondo, el TCP ajusta su noción del tiempo de viaje redondo promedio para la conexión. Por lo general, el software TCP almacena el tiempo estimado de viaje redondo, RTT (*round trip time*), como promedio calculado y utiliza nuevos ejemplos de viaje redondo para cambiar lentamente dicho promedio. Por ejemplo, para computar un nuevo cálculo de promedio, una técnica antigua para promediar se valía de un factor constante de cálculo, α, donde $0 \leq \alpha < 1$, para calcular el promedio anterior contra el último ejemplo de viaje redondo:

$$RTT = (\alpha * Old_RTT) + ((1-\alpha) * New_Round_Trip_Sample)$$

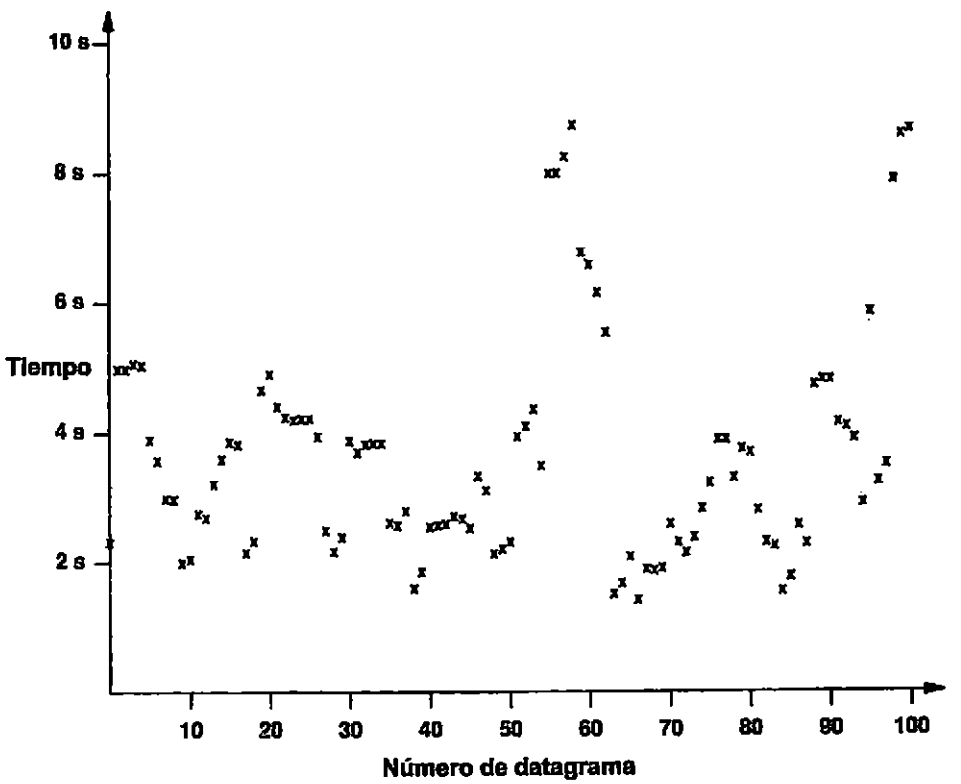


Figura 13.10 Gráfico de tiempos de viaje redondo medidos para 100 datagramas IP sucesivos. Aun cuando la mayor parte de Internet opera ahora con retardos mucho menores, los retardos varían aún en tiempo.

Escoger un valor cercano a *1* para α, hace que el promedio calculado sea inalterable ante los cambios mínimos de tiempo (por ejemplo, un solo segmento que encuentra un gran retraso). Escoger un valor cercano a *0* para α, hace que el promedio calculado responda con rapidez a los cambios en el retraso.

Cuando envía un paquete, el TCP computa un valor de terminación de tiempo como una función de la estimación actual para viaje redondo. Las implantaciones antiguas del TCP se valían de un factor constante de cálculo, ß (ß > 1), y la terminación de tiempo era mayor que la estimación actual de viaje redondo:

$$\text{Terminación de tiempo} = ß * RTT$$

Escoger un valor para ß puede ser difícil. Por una parte, para detectar con rapidez la pérdida de paquetes, el valor de terminación de tiempo debe acercarse al tiempo actual de viaje redondo (por ejemplo, ß debe acercarse a *1*). La rápida detección de pérdida de paquetes mejora la producción de salida porque el TCP no esperará un tiempo innecesariamente largo para retransmitir. Por otra parte, si ß = *1*, el TCP se vuelve muy ansioso —cualquier retraso pequeño causará una retransmisión innecesaria, que desperdiciará el ancho de banda de la red. La especificación original recomendaba establecer ß = *2;* pero trabajos más recientes, descritos abajo, han producido mejores técnicas para el ajuste de la terminación de tiempo.

Podemos resumir las ideas hasta aquí presentadas de la siguiente manera:

> *Para manejar los retrasos variables que se encuentran en un ambiente de red de redes, el TCP utiliza un algoritmo adaptable de retransmisión que monitorea los retrasos en cada conexión y ajusta de acuerdo a ellos su parámetro de terminación de tiempo.*

13.17 Medición precisa de muestras de viaje redondo

En teoría, la medición de una muestra de viaje redondo es trivial —consiste en substraer la hora a la que se envía el segmento de la hora a la que llega el acuse de recibo. Sin embargo, surgen complicaciones debido a que el TCP se vale de un esquema de acuses de recibo acumulativos en el que un acuse se refiere a los datos recibidos y no al caso de un datagrama específico que transporta datos. Considere una retransmisión. El TCP forma un segmento, lo coloca en un datagrama y lo envía, el tiempo termina y el TCP vuelve a enviar el segmento en un segundo datagrama. Como ambos datagramas llevan exactamente los mismos datos, el receptor no tiene forma de saber si un acuse de recibo corresponde al datagrama original o al retransmitido. Este fenómeno se conoce como *ambigüedad de acuse de recibo* (*acknowledgement ambiguity*), y se dice que los acuses de recibo TCP son *ambiguos*.

¿Debe el TCP asumir que los acuses de recibo pertenecen a la primera transmisión (por ejemplo, al original) o a la última (por ejemplo, la transmisión más reciente)? De forma sorprendente, ninguno de los dos casos funciona. La asociación de los acuses de recibo con la transmisión original puede causar que el tiempo estimado de viaje redondo aumente sin medida en los casos en

los que una red de redes pierda datagramas.[6] Si llega un acuse de recibo después de una o más retransmisiones, el TCP medirá la muestra de viaje redondo de la transmisión original y computará un RTT nuevo utilizando la muestra excesivamente larga. Por lo tanto, el RTT crecerá ligeramente. En la siguiente ocasión que el TCP envíe un segmento, el RTT más largo resultará en terminaciones de tiempo ligeramente más grandes, por lo que si llega un acuse de recibo después de una o más retransmisiones, el siguiente tiempo de muestra de viaje redondo será aún más largo, y así sucesivamente.

La asociación de un acuse de recibo con la retransmisión más reciente también puede fallar. Considere lo que sucede cuando el retraso extremo a extremo aumenta repentinamente. Cuando el TCP envía un segmento, utiliza la estimación anterior de viaje redondo para computar una terminación de tiempo, que ahora es demasiado pequeña. El segmento llega y comienza el acuse de recibo, pero el aumento en el retraso significa que el tiempo termina antes de que llegue el acuse y el TCP retransmite el segmento. Poco después de que el TCP hace la retransmisión, llega el primer acuse de recibo y se asocia con la retransmisión. La muestra de viaje redondo será mucho más pequeño y resultará en una ligera disminución del tiempo estimado de viaje redondo, RTT. Por desgracia, disminuir la estimación de tiempo de viaje redondo garantiza que el TCP ajustará una terminación de tiempo demasiado pequeña para el siguiente segmento. Por último, la estimación del tiempo de viaje redondo se puede estabilizar en un valor, T, que sea de tal manera que el tiempo correcto de viaje redondo resulte ligeramente mayor que algunos múltiplos de T. Se ha observado que las implantaciones del TCP que asocian los acuses de recibo con la retransmisión más reciente llegan a un estado estable con el RTT ligeramente menor que la mitad del valor correcto (por ejemplo, el TCP envía cada segmento exactamente dos veces aunque no ocurra ninguna pérdida).

13.18 Algoritmo de Karn y anulación del temporizador

Si tanto la transmisión original como la más reciente fallan en proporcionar tiempos de viaje redondo, ¿qué debe hacer el TCP? La respuesta aceptada es sencilla: el TCP no debe actualizar la estimación de viaje redondo para los segmentos retransmitidos. Esta idea, conocida como *algoritmo de Karn*, evita el problema de todos los acuses de recibo ambiguos únicamente al ajustar la estimación de viaje redondo para acuses de recibo no ambiguos (acuses relacionados con segmentos que sólo se transmitieron una vez).

Por supuesto, una implantación simplista del algoritmo de Karn, que solamente ignore los tiempos para los segmentos retransmitidos, también puede conducir a fallas. Considere lo que sucede si el TCP envía un segmento después de un aumento significativo en el retraso. El TCP computa una terminación de tiempo mediante la estimación existente de viaje redondo. La terminación de tiempo será demasiado pequeña para el nuevo retardo y forzará la retransmisión. Si el TCP ignora los acuses de recibo para los segmentos retransmitidos, nunca actualizará la estimación y el ciclo continuará.

Para resolver dichas fallas, el algoritmo de Karn necesita que el transmisor combine las terminaciones de tiempo de transmisión con una estrategia de *anulación del temporizador* (timer backoff). La técnica de anulación computa una terminación de tiempo inicial por medio de una fórmu-

[6] La estimación sólo puede tener una longitud arbitrariamente grande si todos los segmentos se pierden al menos na vez.

la como la que se mostró anteriormente. Sin embargo, si se termina el tiempo y se provoca una retransmisión, el TCP aumenta el valor de terminación de tiempo. De hecho, cada vez que debe retransmitir un segmento, el TCP aumenta el valor de terminación de tiempo (para evitar que se vuelvan demasiado largos, la mayor parte de las implantaciones limitan los aumentos a una frontera mayor que es más larga que el retraso a lo largo de cualquier camino en la red de redes).

Las implantaciones utilizan varias técnicas para computar la anulación. La mayor parte escoge un factor multiplicativo, γ, y ajustan el nuevo valor a:

$$\text{new_timeout} = \gamma * \text{timeout}$$

Por lo general, γ es *2*. (Se ha argüido que los valores de γ menores a 2 provocan inestabilidades.) Otras implantaciones utilizan una tabla de factores multiplicativos, lo que permite la anulación arbitraria en cada paso.[7]

El algoritmo de Karn combina la técnica de anulación con la estimación de viaje redondo para solucionar el problema de no incrementar las estimaciones de viaje redondo:

> *Algoritmo de Karn: Cuando se compute la estimación de viaje redondo, ignorar los ejemplos que correspondan a los segmentos retransmitidos, pero utilizar una estrategia de anulación, y mantener el valor de terminación de tiempo de un paquete retransmitido para los paquetes subsecuentes, hasta que se obtenga un ejemplo válido.*

Hablando en forma general, cuando una red de redes no se comporta adecuadamente, el algoritmo de Karn separa el cómputo del valor de terminación de tiempo de la estimación actual de viaje redondo. Utiliza la estimación antes mencionada para computar un valor inicial de terminación de tiempo, pero luego anula la terminación en cada retransmisión hasta que pueda transferir un segmento con éxito. Cuando envía segmentos subsecuentes, mantiene el valor de terminación de tiempo que resulta de la anulación. Por último, cuando llega un acuse de recibo correspondiente a un segmento que no requirió retransmisión, el TCP recalcula la estimación de viaje redondo y restablece la terminación de tiempo. La experiencia muestra que el algoritmo de Karn funciona bien, inclusive en las redes que tienen alta pérdida de paquetes.[8]

13.19 Respuesta a una variación alta en el retraso

La investigación sobre la estimación de viajes redondos ha mostrado que los cómputos descritos con anterioridad no se adaptan a un rango amplio de variación en el retraso. La teoría de poner en cola de espera sugiere que las variaciones en el tiempo de viaje redondo, σ, varían proporcionalmente a $1/(1-L)$, donde L es la carga actual de red, $0 \leq L \leq 1$. Si una red de redes está corriendo a 50% de su capacidad, esperamos que el retraso de viaje redondo varíe por un factor de $\pm 2\sigma$, o *4*. Cuando la carga llega al 80%, esperamos una variación de *10*. El estándar TCP original especificaba la téc-

[7] El sistema Berkeley de UNIX es el sistema más notable de los que utiliza una tabla de factores, pero los valores activos en la tabla son equivalentes a usar $\gamma = 2$.

[8] Phil Karn es un radioaficionado entusiasta que ha desarrollado este algoritmo para permitir la comunicación TCP a través de conexiones de paquetes de radio de pérdidas altas.

nica para la estimación del tiempo de viaje redondo que describimos anteriormente. La utilización de esta técnica y la limitación de ß al valor sugerido de 2 significa que la estimación de viaje redondo se puede adaptar a cargas de hasta 30%.

La especificación de 1989 para el TCP necesita que las implantaciones estimen tanto el tiempo promedio de viaje redondo como la variación, y que utilicen la variación estimada en vez de la constante ß. Como resultado, las nuevas implantaciones del TCP se pueden adaptar a un rango más amplio de variación en el retraso y generar sustancialmente una salida más alta. Por fortuna, las aproximaciones requieren muy poca computación; se pueden derivar programas muy eficientes de las siguientes ecuaciones simples:

$$\text{DIFF} = \text{SAMPLE} - \text{Old_RTT}$$

$$\text{Smoothed_RTT} = \text{Old_RTT} + \delta * \text{DIFF}$$

$$\text{DEV} = \text{Old_DEV} + \rho(|\text{DIFF}| - \text{Old_DEV})$$

$$\text{Timeout} = \text{Smoothed_RTT} + \eta * \text{DEV}$$

donde *DEV* es la desviación estimada deseada, δ es una fracción entre *0* y *1* que controla qué tan rápidamente afecta el nuevo ejemplo al promedio calculado, ρ es una fracción entre *0* y *1* que controla qué tan rápidamente afecta el nuevo ejemplo la desviación estimada deseada, y η es un factor que controla qué tanto afecta la desviación a la terminación de tiempo del viaje redondo. Para hacer el cómputo en forma eficiente, el TCP selecciona δ y ρ para que cada una sea un inverso de una potencia de *2*, escala el cómputo por 2^n para lograr *n* apropiadamente, y utiliza aritmética de números enteros. La investigación sugiere que los valores de $\delta = 1/2^3$, $\rho = 1/2^2$, y $n = 3$ funcionarán bien. El valor original para η en 4.3BSD de UNIX era *2*; se cambió a *4* en 4.4BSD de UNIX.

13.20 Respuesta al congestionamiento

Parecería como si el software TCP se pudiera diseñar considerando la interacción entre dos puntos extremos de una conexión y los retrasos en la comunicación entre ellos. Sin embargo, en la práctica, el TCP también debe reaccionar al *congestionamiento* en la red de redes. El congestionamiento es una condición de retraso severo causada por una sobrecarga de datagramas en uno o más puntos de conmutación (por ejemplo, en ruteadores). Cuando ocurre un congestionamiento, los retrasos aumentan y los ruteadores comienzan a colocar en colas de salida a los datagramas hasta poderlos rutear. Debemos recordar que cada ruteador tiene una capacidad finita de almacenamiento y que los datagramas compiten por dicho almacenamiento (por ejemplo, en una red de redes basada en datagramas, no existe una prelocalización de recursos para conexiones TCP individuales). En el peor de los casos, el número total de datagramas entrantes a un ruteador congestionado, crece hasta que el ruteador alcanza su capacidad máxima y comienza a descartar datagramas.

Los puntos extremos por lo general no conocen los detalles sobre dónde ha ocurrido un congestionamiento o por qué. Para ellos, el congestionamiento tan sólo significa un mayor retraso. Por desgracia, la mayor parte de los protocolos de transporte utiliza la terminación de tiempo y la retransmisión, por lo que éstos responden a un aumento en el retraso retransmitiendo datagramas.

Las retransmisiones empeoran el congestionamiento en vez de solucionarlo. Si no se revisa, el incremento en el tráfico producirá mayor retraso, conduciendo a mayor tráfico, y así sucesivamente, hasta que la red no pueda utilizarse. La condición se conoce como *colapso por congestionamiento.*

Para evitar el colapso por congestionamiento, el TCP debe reducir la velocidad de transmisión cuando ocurre un congestionamiento. Los ruteadores verifican la longitud de sus colas de salida y utilizan técnicas como la solicitud de disminución ICMP para informar a los anfitriones que ha ocurrido un congestionamiento,[9] pero los protocolos de transporte como el TCP pueden ayudar a evitar el congestionamiento al reducir automáticamente la velocidad de transmisión siempre que ocurra un retraso. Por supuesto, los algoritmos para evitar los congestionamientos se deben diseñar con cuidado, ya que aun bajo condiciones normales de operación una red de redes tendrá amplias variaciones en los retrasos de viajes redondos.

Para evitar el congestionamiento, el estándar TCP ahora recomienda la utilización de dos técnicas: el *arranque lento* y la *disminución multiplicativa*. Estas técnicas están relacionadas y se pueden implantar con facilidad. Dijimos que para cada conexión, el TCP debe recordar el tamaño de la ventana del receptor (por ejemplo, el tamaño de la memoria intermedia, indicado en los acuses de recibo). Para controlar el congestionamiento, el TCP mantiene un segundo límite, llamado *límite de ventana de congestionamiento* o *ventana de congestionamiento*. En cualquier momento, el TCP actúa como si el tamaño de la ventana fuera:

Allowed_window = min(receiver_advertisement, congestion_window)

En un estado constante de una conexión no congestionada, la ventana de congestionamiento es del mismo tamaño que la ventana del receptor. La reducción de la ventana de congestionamiento reduce el tráfico que el TCP inyectará a la conexión. Para estimar el tamaño de la ventana de congestionamiento, el TCP asume que la mayor parte de la pérdida de datagramas viene del congestionamiento y se vale de la siguiente estrategia:

> *Prevención del Congestionamiento por Disminución Multiplicativa: Cuando se pierda un segmento, reducir a la mitad la ventana de congestionamiento (hasta un mínimo de un segmento). Para los segmentos que permanezcan en la ventana permitida, anular exponencialmente el temporizador para la retransmisión.*

Debido a que el TCP reduce a la mitad la ventana de congestionamiento por *cada* pérdida, disminuye la ventana exponencialmente si la pérdida continúa. En otras palabras, si el congestionamiento continúa, el TCP reduce exponencialmente el volumen de tráfico, así como la velocidad de retransmisión. Si la pérdida continúa, el TCP finalmente limita la transmisión a un solo datagrama y continúa duplicando los valores de terminación de tiempo antes de retransmitir. La idea es proporcionar una reducción rápida y significativa del tráfico, a fin de permitir que otros ruteadores tengan suficiente tiempo para deshacerse de los datagramas que ya tienen en sus colas de espera.

¿Cómo se puede recuperar el TCP cuando termina el congestionamiento? Usted puede sospechar que el TCP debe revertir la disminución multiplicativa y duplicar la ventana de congestionamiento cuando el tráfico comienza a fluir de nuevo. Sin embargo, hacerlo así produciría un sistema

---

[9] En una red congestionada, la longitud de las colas crece exponencialmente para un tiempo significativo.

inestable que oscilaría mucho entre poco tráfico y congestionamiento. Por el contrario, el TCP emplea una técnica, llamada *arranque lento*[10] para aumentar la transmisión:

> *Recuperación de arranque lento (aditiva): siempre que se arranque el tráfico en una nueva conexión o se aumente el tráfico después de un periodo de congestionamiento, activar la ventana de congestionamiento con el tamaño de un solo segmento y aumentarla un segmento cada vez que llegue un acuse de recibo.*

Los arranques lentos evitan saturar la red de redes con tráfico adicional, justamente después de que se libere un congestionamiento o cuando comienzan repentinamente nuevas conexiones.

El término *arranque lento* puede no estar bien aplicado porque bajo condiciones ideales, el arranque no es muy lento. El TCP inicia la ventana de congestionamiento con *1*, envía un segmento inicial y espera. Cuando llega el acuse de recibo, aumenta la ventana de congestionamiento a *2*, envía dos segmentos y espera. Cuando llegan los dos acuses de recibo, cada uno aumenta la ventana de congestionamiento en *1*, por lo que el TCP puede enviar *4* segmentos. Los acuses de recibo por estos *4* segmentos incrementarán a *8* la ventana de congestionamiento. Cuando ocurren cuatro viajes redondos, el TCP puede enviar *16* segmentos, que a veces es lo suficiente para llegar al límite de la ventana del receptor. Inclusive para ventanas muy largas, únicamente toma $\log_2 N$ viajes redondos antes de que el TCP pueda enviar N segmentos.

Para evitar el aumento demasiado rápido del tamaño de la ventana y no causar congestionamiento adicional, el TCP agrega una restricción más. Una vez que la ventana de congestionamiento llega a la mitad de su tamaño original, antes del congestionamiento, el TCP entra en una fase de *prevención de congestionamiento* y hace más lenta la velocidad de incremento. Durante la prevención de congestionamiento, aumenta el tamaño de la ventana por *1* solamente si, para todos los segmentos en la ventana, se tiene acuses de recibo.

Juntos, el incremento de arranque lento, la disminución multiplicativa, la prevención de congestionamiento, la medida de variación, y la anulación exponencial del temporizador mejoran notablemente el desempeño del TCP, sin agregar ningún trabajo computacional significativo del software de protocolo. Las versiones que utilizan estas técnicas han mejorado el desempeño de versiones anteriores en factores de *2* a *10*.

13.21 Establecimiento de una conexión TCP

Para establecer una conexión, el TCP utiliza un saludo (*handshake*) de tres etapas. En el caso más sencillo, este intercambio procede como se muestra en la figura 13.11.

El primer segmento del saludo se puede identificar porque tiene activo el bit SYN[11] en el campo de código. El segundo mensaje tiene tanto el bit SYN como el bit ACK activos, indicando tanto el acuse de recibo del primer segmento SYN como el hecho de que se continúa con el intercambio. El mensaje final del saludo es sólo un acuse de recibo y nada más se utiliza para informar al destino que ambos extremos están de acuerdo en establecer una conexión.

---

[10] El término *arranque lento* se atribuye a John Nagle; la técnica se conoció originalmente como *arranque suave*.

[11] SYN es una expresión que se emplea como abreviatura de *synchronization*: se pronuncia "sin".

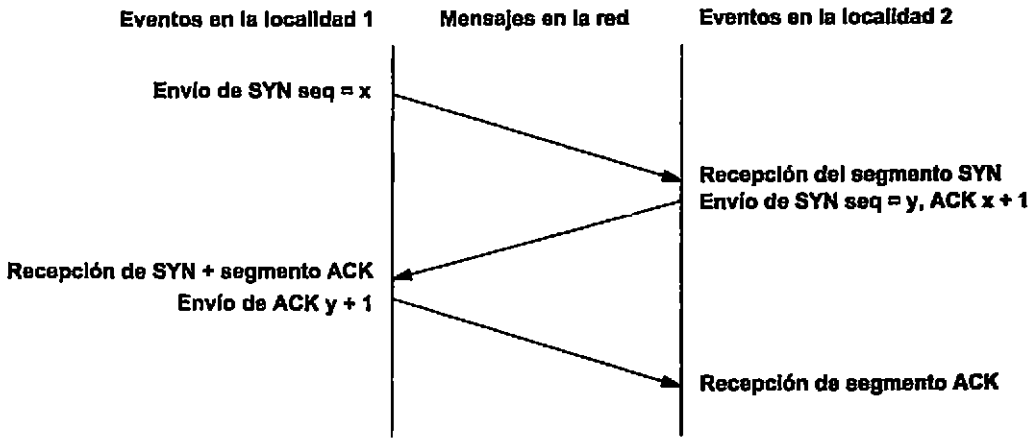


Figura 13.11 Secuencia de mensajes en el saludo de tres etapas. En la representación, el tiempo transcurre hacia la parte inferior de la página; las líneas diagonales representan segmentos enviados entre localidades. Los segmentos SYN transportan información sobre el número de secuencia inicial.

Por lo general, el software TCP en una máquina espera de forma pasiva el intercambio de señales y el software TCP en otra máquina lo inicia. Sin embargo, el saludo (handshake) está cuidadosamente diseñado para funcionar aun cuando ambas máquinas intenten iniciar una conexión al mismo tiempo. Por lo tanto, se puede establecer una conexión desde cualquier extremo o desde ambos al mismo tiempo. Una vez que se establece la conexión, los datos pueden fluir en ambas direcciones por igual. No existe un maestro ni un esclavo.

El saludo de tres etapas es necesario y suficiente para la sincronización correcta entre los dos extremos de la conexión. Para entender por qué, recuerde que el TCP se construye sobre un servicio de entrega no confiable de paquetes, así que los mensajes pueden perderse, retrasarse, duplicarse o entregarse en desorden. Por lo tanto, el protocolo debe utilizar un mecanismo de terminación de tiempo y retransmitir las solicitudes perdidas. Sucederán algunos problemas si las solicitudes originales y retransmitidas llegan mientras se establece la conexión o si las solicitudes retransmitidas se retrasan hasta que se establezca, utilice y termine una conexión. Un saludo de tres etapas (más la regla de que el TCP ignore solicitudes adicionales de conexión después de que se establezca la misma), resuelve estos problemas.

13.22 Números de secuencia inicial

El saludo (*handshake*) de tres etapas realiza dos funciones importantes. Garantiza que ambos lados estén listos para transferir datos (y que tengan conocimiento de que ambos están listos) y permite, a ambas partes, acordar un número de secuencia inicial. Los números de secuencia son enviados y reconocidos durante el saludo. Cada máquina debe seleccionar un número de secuencia inicial en forma aleatoria que se utilizará para identificar octetos en el flujo que se está enviando. Los núme-

ros de secuencia no pueden comenzar siempre con el mismo valor. En particular, el TCP no puede seleccionar una secuencia *1* cada vez que crea una conexión (en uno de los ejercicios se examinan los problemas que se pueden originar si se hace de esta manera). Por supuesto, es importante que ambas partes acuerden un número inicial, así como el número de octetos empleados en un acuse de recibo de acuerdo a los utilizados en el segmento de datos.

Para entender cómo pueden acordar las máquinas un número de secuencia para dos flujos después de tres mensajes solamente, recordemos que cada segmento contiene un campo de número de secuencia y un campo de acuse de recibo. La máquina *A*, que inicia un saludo, transfiere un número de secuencia inicial, *x*, en el campo de secuencia del primer segmento SYN como parte del saludo de tres etapas. La segunda máquina, *B*, recibe el SYN, registra el número de secuencia y responde enviando su número de secuencia inicial en el campo de secuencia así como un reconocimiento que especifica el octeto $x+1$ esperado por *B*. En el mensaje final del saludo, *A* envía un "acuse de recibo" de la recepción del mensaje de *B* de todos los octetos a través de *y*. En todos los casos, los acuses de recibo siguen la convención de utilizar el número del *próximo* octeto esperado.

Hemos descrito cómo el TCP normalmente transporta el saludo de tres etapas intercambiando segmentos que contienen una cantidad mínima de información. Debido al diseño del protocolo es posible enviar datos junto con los números de secuencia iniciales en los segmentos de saludo. En cada caso el software TCP debe manejar los datos hasta que se complete el saludo. Una vez que la conexión se ha establecido, el software TCP puede liberar los datos manejados y entregarlos rápidamente al programa de aplicación. El lector deberá referirse a las especificaciones del protocolo para obtener más detalles.

13.23 Terminación de una conexión TCP

Dos programas que utilizan el TCP para comunicarse pueden terminar la conversación cortésmente valiéndose de la operación *close*. De manera interna, el TCP utiliza una modificación del saludo de tres etapas para cerrar conexiones. Recordemos que las conexiones TCP son de tipo full duplex y que hemos visto que éstas contienen dos transferencias de flujo independientes, una en cada dirección. Cuando un programa de aplicación informa al TCP que ya no tiene más datos para enviar, éste cerrará la conexión *en una dirección*. Para cerrar la mitad de una conexión, el emisor TCP termina de transmitir los datos restantes, espera la recepción de un acuse de recibo y, entonces, envía un segmento con el bit FIN activado. El receptor TCP reconoce el segmento FIN e informa al programa de aplicación en su extremo que no tiene más datos disponibles (por ejemplo, mediante el mecanismo de fin de archivo de sistema operativo).

Una vez que la conexión se ha cerrado en una dirección dada, TCP rechaza más datos en esta dirección. Mientras tanto, los datos pueden continuar fluyendo en la dirección opuesta hasta que el emisor se cierra. Por supuesto, los acuses de recibo continúan fluyendo hacia el emisor incluso después de que la conexión se ha cerrado. Cuando ambas direcciones se han cerrado, el software TCP en cada punto extremo borra sus registros de la conexión.

Los detalles del cierre de una conexión son más sutiles de lo que se ha sugerido anteriormente porque el TCP utiliza un saludo de tres etapas modificado para cerrar una conexión. La figura 13.12 ilustra el procedimiento.

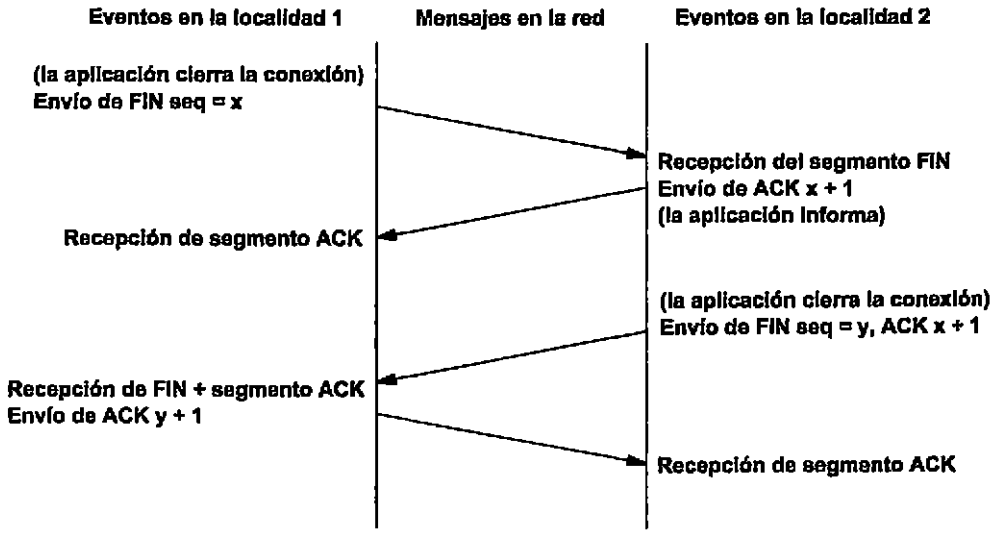


Figura 13.12 Modificación del saludo de tres etapas utilizado para cerrar conexiones. La localidad que recibe el primer segmento FIN lo reconoce de inmediato y, después, lo retarda antes de enviar el segundo segmento FIN.

La diferencia entre el saludo de tres etapas empleado para establecer e interrumpir conexiones se presenta luego de que una máquina recibe el segmento FIN inicial. En lugar de generar un segundo segmento FIN inmediatamente, el TCP envía un acuse de recibo y luego informa a la aplicación de la solicitud de interrupción. Informar al programa de aplicación de la solicitud y obtener una respuesta, puede tomar un tiempo considerable (por ejemplo, si comprende la interacción humana). El acuse de recibo evita la retransmisión del segmento inicial FIN durante la espera. Por último, cuando el programa de aplicación instruye al TCP para que interrumpa la conexión completamente, el TCP envía el segundo segmento FIN y la localidad original responde con el tercer mensaje, un ACK.

13.24 Restablecimiento de una conexión TCP

Normalmente, un programa de aplicación se vale de la operación de cierre para interrumpir una conexión cuando termina de utilizarla. Así, el cierre de conexión es considerado como una parte normal de su uso, análogo al cierre de archivos. Algunas veces se presentan condiciones anormales que obligan a un programa de aplicación o al software de red a interrumpir una conexión. El TCP proporciona una capacidad de iniciación (*reset*) para estas desconexiones anormales.

Para iniciar una conexión, un lado inicia la interrupción enviando un segmento con el bit RST activado en el campo *CODE*. El otro lado responde a un segmento de iniciación inmediatamente interrumpiendo la conexión. El TCP también informa al programa de aplicación que se ha

presentado una iniciación. Una iniciación es una interrupción instantánea, lo cual significa que la transferencia en ambas direcciones se interrumpe de manera inmediata y se liberan recursos como los búfers.

13.25 Máquina de estado TCP

Como en la mayor parte de los protocolos, la operación del TCP se puede explicar mejor mediante un modelo teórico, llamado *máquina de estado finito*. La figura 13.13 muestra la máquina de estado finito TCP, en ella los círculos representan estados y las flechas representan transiciones entre éstos. El nombre en cada transición muestra qué recibe el TCP para generar la transición y qué envía como respuesta. Por ejemplo, el software TCP en cada extremo comienza en un estado *CLOSED* (*CERRADO*). El programa de aplicación debe emitir un comando *passive open* (apertura pasiva) (para esperar una conexión desde otra máquina) o un comando *active open* (apertura activa) (para iniciar una conexión). El comando active open obliga a que se dé una transición del estado *CLOSED* al estado *SYN SENT*. Cuando el TCP continúa con la transición, emite un segmento SYN. Cuando el otro extremo devuelve un segmento que contiene un SYN, más un ACK, el TCP cambia al estado *ESTABLISHED* (*ESTABLECIDO*) y comienza la transferencia de datos.

El estado *TIMED WAIT* (*ESPERA CRONOMETRADA*) revela cómo el maneja el TCP algunos de los problemas que se presentan con la entrega no confiable. El TCP conserva una noción de *máximo tiempo de vida del segmento*, el tiempo máximo en que un segmento puede mantenerse activo en una red de redes. Para evitar tener segmentos de una conexión previa interfiriendo con los actuales, el TCP cambia el estado *TIMED WAIT* después de cerrar una conexión. Se mantiene en este estado dos veces el máximo tiempo de vida del segmento antes de borrar sus registros de la conexión. Si algún segmento duplicado logra llegar a la conexión durante el intervalo de exceso de tiempo, el TCP lo rechazará. Sin embargo, para manejar casos cuyo el último acuse de recibo fue perdido, reconocerá los segmentos válidos y reiniciará el temporizador. Dado que el temporizador permite al TCP distinguir entre las conexiones anteriores de las nuevas, se evita que el TCP responda con un *RST* (iniciación) si el otro extremo retransmite una solicitud *FIN*.

13.26 Forzando la entrega de datos

Hemos dicho que el TCP es libre de dividir el flujo de datos en segmentos para su transmisión sin considerar el tamaño de transferencia que utiliza el programa de aplicación. La mayor ventaja de permitir que el TCP elija la forma de dividir es la eficiencia que se obtiene. Puede acumular suficientes octetos en una memoria intermedia para hacer los segmentos razonablemente largos, reduciendo las sobrecargas altas que se presentan cuando los segmentos contienen sólo unos cuantos octetos de datos.

Aun cuando el procesamiento en memoria intermedia mejora el desempeño de la red, puede interferir con algunas aplicaciones. Consideremos el uso de una conexión de TCP para transferir caracteres de una terminal interactiva a una máquina remota. El usuario espera una respuesta instantánea para cada pulsación de tecla. Si el emisor TCP pone en memoria intermedia los datos, la

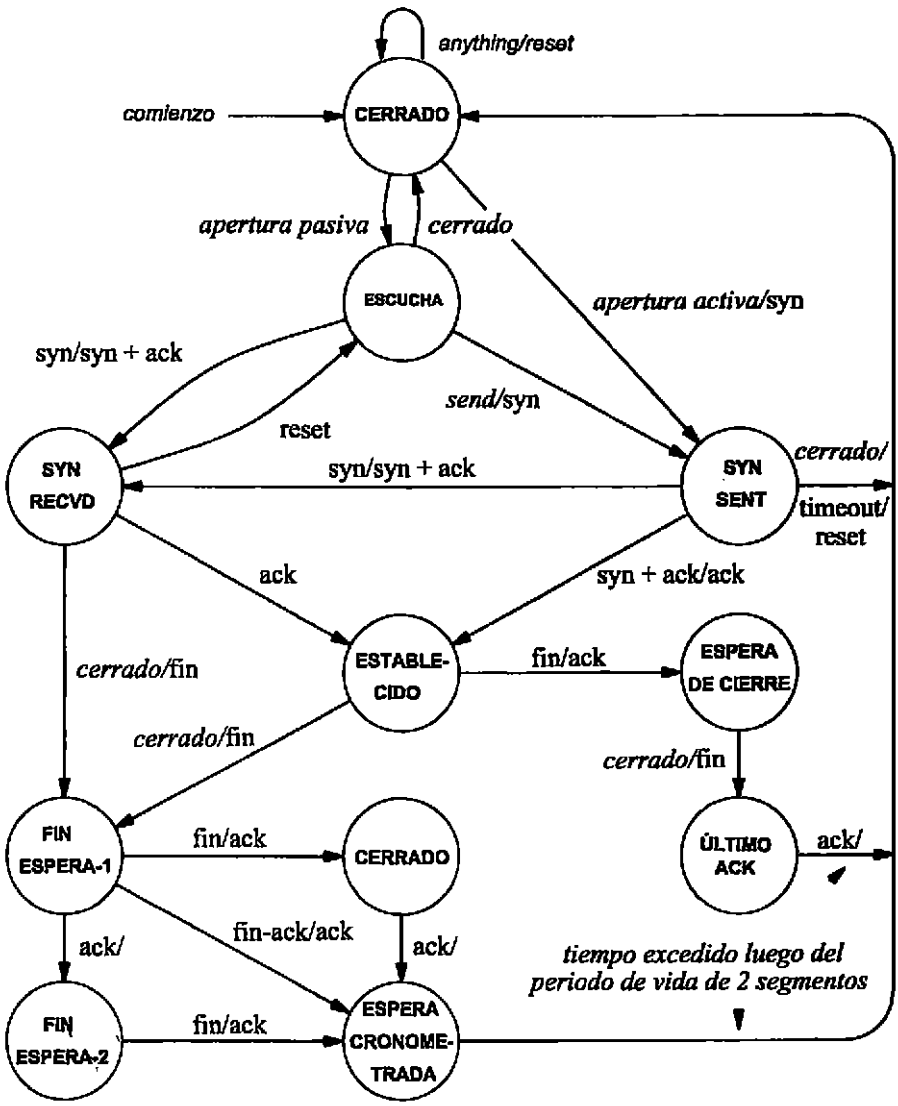


Figura 13.13 Máquina de estado finito TCP. Cada punto final comienza en el estado *cerrado*. Los nombres de las transiciones muestran la entrada que ocasiona la transición, seguida por la salida, si la hay.

respuesta podría retrasarse, posiblemente por cientos de pulsaciones de teclas. De la misma forma, debido a que el receptor TCP puede poner en memoria intermedia los datos antes de que estén disponibles para el programa de aplicación en su extremo, forzar al emisor a transmitir los datos puede no ser suficiente para garantizar la entrega.

Para adaptarse a los usos interactivos, el TCP proporciona la operación *push* (*empujar*), que un programa de aplicación puede utilizar para forzar la entrega de octetos actuales en el flujo de transmisión sin esperar a que se les almacene en memoria intermedia. La operación empujar hace más que forzar al TCP a enviar un segmento. También solicita al TCP que active el bit *PSH* en el segmento de campo de código, así los datos se entregarán al programa de aplicación en el extremo de recepción. Entonces, cuando se envían datos desde una terminal interactiva, la aplicación utiliza la función empujar luego de cada pulsación de tecla. De la misma forma, los programas de aplicación pueden forzar la salida para que sea enviada y desplegada en el indicador de la terminal llamando a la función empujar luego de escribir un carácter o una línea.

13.27 Números reservados de puerto TCP

Como el UDP, el TCP combina la asignación dinámica y estática de puertos mediante un conjunto de *asignación de puertos bien conocidos* para programas llamados con frecuencia (por ejemplo, el correo electrónico), pero la salida de la mayor parte de los números de puerto disponibles para el sistema operativo se asigna conforme los programas lo necesitan. Aun cuando el estándar original reservaba los números de puerto menores a *256* para utilizarlos como puertos bien conocidos, ahora se han asignado números superiores a 1,024. La figura 13.14 lista algunos de los puertos TCP asignados en la actualidad. Habría que puntualizar que, aunque los números de puerto TCP y UDP son independientes, los diseñadores han decidido utilizar el mismo número entero de puerto para cualquier servicio accesible desde UDP y TCP. Por ejemplo, un servidor de nombre de dominio puede acesarse con el TCP o el UDP. En ambos protocolos, el puerto número 53 ha sido reservado para los servidores en el sistema de nombre de dominio.

13.28 Desempeño del TCP

Como hemos visto, el TCP es un protocolo complejo que maneja las comunicaciones sobre una amplia variedad de tecnologías de red subyacentes. Mucha gente asume que, como el TCP aborda tareas mucho más complejas que otros protocolos de transporte, el código debe ser incómodo e ineficaz. Sorpresivamente, en general, lo que hemos analizado no parece entorpecer el desempeño del TCP. Experimentos realizados en Berkeley han mostrado que el mismo TCP que opera en forma eficiente en la red global de Internet puede proporcionar un desempeño sostenido a 8 Mbps con datos de usuario entre dos estaciones de trabajo en una red Ethernet a 10 Mbps.[12] Los investigadores de Cray Research, Inc. han demostrado que el desempeño del TCP se acerca a un gigabit por segundo.

[12] Encabezado Ethernet, IP, y TCP y el espacio requerido de "inter-packet", representan en amplitud de banda remanente.

Decimal	Clave	Clave UNIX	Descripción
0			Reservado
1	TCPMUX	-	Multiplexor TCP
5	RJE	-	Introducción de función remota
7	ECHO	echo	Echo
9	DISCARD	discard	Abandonar
11	USERS	systat	Usuarios activos
13	DAYTIME	daytime	Fecha, hora
15	-	netstat	Programa de estado de red
17	QUOTE	qotd	Cita del día
19	CHARGEN	chargen	Generador de caracteres
20	FTP-DATA	ftp-data	Protocolo de transferencia de archivos (datos)
21	FTP	ftp	Protocolo de transferencia de archivos
23	TELNET	telnet	Conexión de Terminal
25	SMTP	smtp	Protocolo de transporte de correo sencillo
37	TIME	time	Hora
42	NAMESERVER	name	Nombre del anfitrión servidor
43	NICNAME	whois	¿Quién está ahí?
53	DOMAIN	nameserver	Servidor de nombre de dominio
77	-	rje	Cualquier servicio RJE privado
79	FINGER	finger	Finger
93	DCP	-	Protocolo de control de dispositivo
95	SUPDUP	supdup	Protocolo SUPDUP
101	HOSTNAME	hostnames	Servidor de nombre de anfitrión NIC
102	ISO-TSAP	iso-tsap	ISO-TSAP
103	X400	x400	Servicio de correo X.400
104	X400-SND	x400-snd	Envío de correo X.400
111	SUNRPC	sunrpc	Llamada a procedimiento remoto de SUN
113	AUTH	auth	Servicio de autenticación
117	UUCP-PATH	uucp-path	Servicio de trayecto UUCP
119	NNTP	nntp	Protocolo de transferencia de noticias USENET
129	PWDGEN	-	Protocolo generador de clave de acceso
139	NETBIOS-SSN	-	Servicio de sesión NETBIOS
160-223	Reservado		

Figura 13.14 Ejemplos de números de puerto TCP asignados actualmente. Para posibles extensiones, protocolos como el UDP utilizan los mismos números.

13.29 Síndrome de ventana tonta y paquetes pequeños

Los investigadores que participaron en el desarrollo del TCP observaron un serio problema de desempeño que puede presentarse cuando las aplicaciones del emisor y el receptor operan a velocidades diferentes. Para entender el problema, recordemos que el TCP almacena en memoria intermedia los datos de entrada y consideremos lo que sucedería si la aplicación de un receptor elige leer los datos de entrada un octeto a la vez. Cuando una conexión se establece por primera vez, el receptor TCP asigna un búfer de K octetos y utiliza el campo *WINDOW*, en los segmentos de acuse de recibo, para anunciar el tamaño disponible del búfer al emisor. Si la aplicación del emisor gene-

ra datos rápidamente, el emisor TCP transmitirá segmentos con datos para toda la ventana. Finalmente, el emisor recibirá un acuse de recibo que especifique que toda la ventana está llena y que no queda espacio adicional en el búfer del receptor.

Cuando la aplicación de recepción lee un octeto de datos desde la memoria intermedia llena, queda disponible un espacio de un octeto. Hemos dicho que cuando un espacio queda disponible en el búfer, el TCP genera un acuse de recibo que utiliza el campo *WINDOW*, en la máquina de recepción, para informar al emisor. En el ejemplo, el receptor anunciará una ventana de un octeto. Cuando tenga conocimiento del espacio disponible, el emisor TCP responderá con la transmisión de un segmento que contenga un octeto de datos.

Aun cuando el anuncio de la ventana de un solo octeto trabaja de manera correcta conservando llena la memoria intermedia del receptor, el resultado es una serie de segmentos de datos pequeños. El emisor TCP debe componer un segmento que contenga un octeto de datos, colocar el segmento en un datagrama IP y transmitir el resultado. Cuando la aplicación de recepción lea otro octeto, el TCP generará otro acuse de recibo, lo cual ocasionará que el emisor transmita otro segmento que contenga un octeto de datos. La interacción resultante puede llegar a estabilizarse en un estado en el cual el TCP envíe un segmento separado para cada octeto de datos.

La transferencia de segmentos pequeños ocupa ancho de banda de la red innecesariamente e introduce una sobrecarga computacional. La transmisión de pequeños segmentos ocupa un ancho de banda pues cada datagrama transfiere sólo un octeto de datos; la cantidad de encabezados para los datos será muy extensa. La sobrecarga computacional se origina debido a que el TCP, tanto en el emisor como en el receptor, debe procesar cada segmento. El software TCP del emisor tiene que asignar espacio de memoria intermedia, formar un encabezado de segmento y calcular una suma de verificación para el segmento. De la misma forma, el software IP en la máquina emisora debe encapsular el segmento en un datagrama, calcular la suma de verificación del encabezado, rutear el datagrama y transferirlo hacia la interfaz de red apropiada. En la máquina de recepción, el IP debe verificar la suma de verificación del encabezado y transferir el segmento hacia el TCP. El TCP tiene que verificar la suma de verificación del segmento, examinar el número de secuencia, extraer el dato y colocarlo en una memoria intermedia.

Aun cuando hemos descrito lo que pueden provocar los segmentos pequeños cuando un receptor anuncia una ventana pequeña disponible, un emisor puede también ocasionar que cada segmento contenga una pequeña cantidad de datos. Por ejemplo, imagine una implantación del TCP que envía datos agresivamente cada vez que están disponibles y considere qué sucedería si una aplicación del emisor generara un octeto de datos por vez. Luego de que la aplicación generara un octeto de datos, el TCP crearía y transmitiría un segmento. El TCP puede también enviar un segmento pequeño si una aplicación genera datos en bloques de tamaños fijos de B octetos y el emisor TCP extrae datos de la memoria intermedia en bloques del tamaño del segmento máximo M, donde M es diferente de B, dado que el último bloque en una memoria intermedia puede ser pequeño.

Este problema se conoce como *síndrome de ventana tonta* (*silly window syndrome* o *SWS* por sus siglas en inglés) y se convirtió en una plaga en las primeras implantaciones del TCP. En resumen:

> *Las primeras implantaciones del TCP presentaron un problema conocido como* síndrome de ventana tonta *en el cual cada acuse de recibo anunciaba una pe-*

queña cantidad de espacio disponible y cada segmento transportaba una pequeña cantidad de datos.

13.30 Prevención del síndrome de ventana tonta

Las especificaciones del TCP incluyen ahora la heurística necesaria para prevenir el síndrome de ventana tonta. La heurística utilizada en una máquina emisora evita la transmisión de cantidades pequeñas de datos en cada segmento. Otra heurística empleada en la máquina receptora evita la emisión de incrementos pequeños en los anuncios de ventana que pueden activar paquetes de datos pequeños. Aun cuando las heurísticas juntas trabajan bien, tanto el receptor como el emisor evitan el síndrome de ventana tonta ayudando a asegurar un buen desempeño en caso de que uno de los extremos falle en la correcta implantación del procedimiento para evitar las ventanas tontas.

En la práctica el software TCP debe contener el código necesario para evitar el síndrome de ventana tonta, tanto en el emisor como en el receptor. Para entender por qué, recordemos que la conexión del TCP es de tipo full duplex los datos pueden fluir en ambas direcciones. Así, una implementación del TCP incluye tanto el código para enviar datos como el código para recibirlos.

13.30.1 Prevención de la ventana tonta en el lado del receptor

La heurística que utiliza un receptor para evitar las ventanas tontas es fácil de entender. En general, un receptor mantiene un registro interno de la ventana disponible en el momento, pero retarda los anuncios para incrementar el tamaño de la ventana del emisor hasta que la ventana pueda avanzar una cantidad significativa. La definición de "significativa" depende del tamaño de la memoria intermedia del receptor y del tamaño del segmento máximo. El TCP lo define como el mínimo de la mitad de la memoria intermedia del receptor o el número de octetos de datos en un segmento de tamaño máximo.

El procedimiento para evitar las ventanas tontas en el lado del receptor evita el anuncio de ventanas pequeñas en caso de que una aplicación de recepción extraiga octetos de datos lentamente. Por ejemplo, cuando la memoria intermedia de un receptor se llena por completo, envía un acuse de recibo que contiene un anuncio de ventana en 0. Conforme la aplicación del receptor extrae octetos de la memoria intermedia, el receptor TCP calcula nuevamente el espacio disponible en la memoria intermedia. En lugar de enviar el anuncio de una ventana inmediatamente, el receptor espera hasta que se logre un espacio disponible equivalente a la mitad del tamaño de la memoria intermedia o equivalente al segmento de tamaño máximo. Así, el emisor siempre recibirá incrementos extensos en la ventana actual, permitiendo la transferencia de segmentos grandes. La heurística puede resumirse en la siguiente forma:

Procedimiento para evitar ventanas tontas en el lado del receptor: antes de enviar el anuncio de una ventana actualizada, luego de anunciar una ventana igual a 0, esperar hasta que se obtenga un espacio disponible que sea equivalente a por lo menos el 50% del tamaño total de la memoria intermedia o igual al segmento de tamaño máximo.

13.30.2 Acuses de recibo retardados

Se han tomado dos enfoques para implantar la prevención de las ventanas tontas en el lado del receptor. En el primer método, el TCP acusa de recibido cada segmento que llega pero no anuncia un incremento en sus ventanas hasta que la ventana alcanza el límite especificado por la heurística para la prevención de las ventanas tontas. En el segundo método, el TCP retarda el envío de un acuse de recibo cuando la prevención de las ventanas tontas especifica que la ventana no es lo suficientemente grande como para anunciarse. Los estándares recomiendan retrasar los acuses de recibo.

El retraso de los acuses de recibo tiene ventajas y desventajas. La mayor ventaja reside en que el retardo de los acuses de recibo puede reducir el tráfico y, por lo tanto, mejorar el desempeño. Por ejemplo, si llegan datos adicionales durante el periodo de retardo, un solo acuse de recibo reconocerá a todos los datos recibidos. Si la aplicación de recepción genera una respuesta inmediata después de la llegada de los datos (por ejemplo, un eco de caracteres en una sesión remota en línea), un pequeño retardo puede permitir que el acuse de recibo incorpore un segmento de datos. Sin embargo, el TCP no puede cambiar esta ventana hasta que la aplicación de recepción extraiga los datos desde la memoria intermedia. En los casos en que la aplicación de recepción lee los datos conforme éstos llegan, un corto retardo permite al TCP enviar un solo segmento de acuse de recibo de los datos y anunciar una actualización de ventana. Dentro del acuse de recibo retardado, el TCP reconocerá la llegada de datos inmediatamente y después enviará un acuse de recibo adicional para actualizar el tamaño de la ventana.

La desventaja del retardo en los acuses de recibo debería ser clara. Un aspecto importante es que, si un receptor retarda los acuses de recibo por mucho tiempo, el emisor TCP retransmitirá el segmento. Las retransmisiones innecesarias reducen el desempeño debido a que desperdician ancho de banda de la red. Además, las retransmisiones ocasionan una sobrecarga en las máquinas de emisión y recepción. Además, el TCP utiliza la llegada de los acuses de recibo para estimar los tiempos de viaje redondo; el retardo en los acuses de recibo puede provocar una estimación confusa y hacer que los tiempos de retransmisión sean demasiado largos.

Para evitar problemas potenciales, el estándar TCP define un límite para el tiempo de los retardos de acuse de recibo. Las implantaciones no pueden retrasar un acuse de recibo por más de 500 milisegundos. Además, para garantizar que el TCP reciba un número suficiente de estimaciones de viaje redondo, el estándar recomienda que un receptor debe acusar de recibido por lo menos cada segmento de datos diferente.

13.30.3 Prevención de la ventana tonta del lado del emisor

La heurística que un emisor TCP utiliza para evitar las ventanas tontas es sorprendente y elegante. Recordemos que el objetivo es evitar el envío de segmentos pequeños. También no olvidemos que una aplicación de emisión puede generar datos en bloques arbitrariamente pequeños (por ejemplo, de un octeto). Así, para lograr este objetivo, un emisor TCP debe permitir a la aplicación del emisor hacer múltiples llamadas a *write* y debe reunir los datos transferidos con cada llamada antes de transmitirlos a un solo segmento largo. Es decir que un emisor TCP debe retardar el envío de un segmento hasta que pueda acumular una cantidad razonable de datos. Esta técnica se conoce con el nombre de *clumping* (*agrupamiento*).

La cuestión es la siguiente, ¿qué tanto debe esperar el TCP antes de transmitir datos? Por un lado, si el TCP espera demasiado la aplicación tendrá retardos demasiado largos. Algo muy importante es que el TCP no puede saber si tiene que esperar pues no puede saber si la aplicación generará más datos en un futuro cercano. Por otro lado, si el TCP no espera lo suficiente, los segmentos serán pequeños y el desempeño se reducirá.

Los primeros protocolos diseñados para el TCP enfrentaron el mismo problema y utilizaron técnicas para agrupar datos en paquetes grandes. Por ejemplo, para obtener una transferencia eficaz a través de una red, los protocolos de terminal remota originales retrasaban la transmisión de cada pulsación de tecla por unos pocos cientos de milisegundos para determinar si el usuario continuaba presionando la tecla. Como el TCP está diseñado con propósitos generales, puede usarse para un conjunto diverso de aplicaciones. Los caracteres pueden viajar a través de una conexión TCP dado que un usuario pulsa una tecla o dado que un programa transfiere archivos. Un retraso fijo no es óptimo para todas las aplicaciones.

Como en los algoritmos del TCP utilizados para la retransmisión y el algoritmo de comienzo lento empleado para evitar el congestionamiento, la técnica que un emisor TCP utiliza para evitar el envío de paquetes pequeños es flexible —el retraso depende del desempeño actual de la red de redes. Como en el comienzo lento, la prevención de la ventana tonta en el lado del emisor se conoce como *self clocking* pues no se calculan retardos. Por el contrario, el TCP utiliza la llegada de un acuse de recibo para disparar la transmisión de paquetes adicionales. Esta heurística se puede resumir de la siguiente forma:

> *Procedimiento para evitar la ventana tonta del lado del emisor: cuando una aplicación de emisión genera datos adicionales para enviarse sobre una conexión por la que se han transmitido datos anteriormente, pero de los cuales no hay un acuse de recibo, debe colocarse los datos nuevos en la memoria intermedia de salida como se hace normalmente, pero no enviar segmentos adicionales hasta que se reúnan suficientes datos para llenar un segmento de tamaño máximo. Si todavía se está a la espera cuando llega un acuse de recibo, debe enviarse todos los datos que se han acumulado en la memoria intermedia. Aplíquese la regla incluso cuando el usuario solicita la operación de* empujar.

Si una aplicación genera datos un octeto por vez, el TCP enviará el primer octeto inmediatamente. Sin embargo, hasta que llegue el ACK, el TCP acumulará octetos adicionales en su memoria intermedia. Así, si la aplicación es razonablemente rápida, comparada con la red (por ejemplo, en una transferencia de archivo), los segmentos sucesivos contendrán muchos octetos. Si la aplicación es lenta en comparación con la red (por ejemplo, un usuario que pulsa un teclado), se enviarán segmentos pequeños sin retardos largos.

Conocida como *algoritmo Nagle,* en honor a quien la inventó, esta técnica es especialmente elegante pues requiere de una pequeña carga computacional. Un anfitrión no necesita conservar temporizadores separados para cada conexión ni hacer que el anfitrión examine un reloj cuando una aplicación genera datos. Algo muy importante, a través de esta técnica se logra la adaptación a combinaciones arbitrarias de retardos de red, tamaños de segmento máximo y velocidad de aplicaciones, lo cual no disminuye el desempeño en casos convencionales.

Para entender por qué el desempeño se conserva para las comunicaciones convencionales, observemos que las aplicaciones optimizadas para altos desempeños no generan datos un octeto a

la vez (de hacerlo así se incurriría en sobrecargas innecesarias del sistema operativo). De hecho, cada aplicación escribe grandes bloques de datos con cada llamada. Así, la memoria intermedia de salida del TCP tiene suficientes datos para al menos un segmento de tamaño máximo. Además, como la aplicación produce datos con mayor rapidez comparado a como el TCP los puede transferir, la memoria intermedia del emisor se mantiene casi llena y el TCP no tiene retardos de transmisión. Como resultado , el TCP continúa enviando segmentos en la medida en que la red de redes lo puede tolerar mientras la aplicación continúe llenando la memoria intermedia. En resumen:

> *El TCP ahora requiere que el emisor y el receptor implanten heurísticas que eviten el síndrome de ventana tonta. Un receptor evita anunciar ventanas pequeñas y un emisor utiliza un esquema flexible para retardar la transmisión y, así, agrupar datos dentro de segmentos largos.*

13.31 Resumen

El Protocolo de Control de Transmisión (TCP por sus siglas en inglés) define un servicio clave proporcionado para una red de redes llamado entrega de flujo confiable. El TCP proporciona una conexión tipo full duplex entre dos máquinas, lo que les permite intercambiar grandes volúmenes de datos de manera eficaz.

Dado que utiliza un protocolo de ventana deslizable, el TCP puede hacer eficiente el uso de la red. Como se hacen pocas suposiciones sobre el sistema de entrega subyacente, el TCP es lo suficientemente flexible como para operar sobre una gran variedad de sistemas de entrega. Ya que proporciona un control de flujo, el TCP permite que el sistema cuente con una amplia variedad de velocidades para la comunicación.

La unidad básica de transferencia utilizada por el TCP es un segmento. Los segmentos se emplean para transferir datos o información de control (por ejemplo, para permitir que el software TCP en dos máquinas establezca o interrumpa la comunicación). El formato de los segmentos permite a una máquina incorporar un acuse de recibo para datos que fluyen en una dirección, incluyéndolos en el encabezado del segmento de datos que fluyen en el sentido opuesto.

El TCP implanta el control de flujo estableciendo, en el anuncio del receptor, la cantidad de datos que está dispuesto a aceptar. También soporta mensajes fuera de banda utilizando una capacidad de datos urgentes y forzando la entrega por medio de un mecanismo de empuje.

El estándar TCP actual especifica un retroceso exponencial para los temporizadores de retransmisión y algoritmos de prevención de congestionamiento como el de arranque lento, disminución multiplicativa e incremento aditivo. Además, el TCP se vale de procedimientos heurísticos para evitar la transferencia de paquetes pequeños.

PARA CONOCER MÁS

El estándar para TCP puede encontrarse en Postel (RFC 793); Braden (RFC 1122) contiene una actualización que aclara varios puntos. Clark (RFC 813) describe la administración de ventanas TCP,

Clark (RFC 816) describe las fallas en el aislamiento y la recuperación y Postel (RFC 879) reporta el tamaño de segmento máximo de TCP. Nagle (RFC 896) comenta la congestión en las redes TCP/IP y explica el efecto del cronometrado autónomo en el procedimiento para evitar las ventanas tontas. Karn y Partridge (1987) analizan la estimación del tiempo de viaje redondo y presentan el algoritmo de Karn. Jacobson (1988) presenta el algoritmo de control de congestión que ahora es una parte necesaria del estándar. Tomlinson (1975) considera el saludo de tres etapas con mayor detalle. Mills (RFC 889) reporta mediciones de retardos de viaje redondo de Internet. Jain (1986) describe el control de congestión, basado en el temporizador, en el ambiente de una ventana tonta. Borman (abril 1989) resume experimentos con el TCP de alta velocidad en computadoras Cray.

EJERCICIOS

13.1 El TCP utiliza un campo finito para contener números de secuencia de flujo. Estudie las especificaciones del protocolo para que descubra cómo éste permite a un flujo de longitud arbitraria pasar de una máquina a otra.

13.2 Las notas de texto de una de las opciones del TCP permiten a un receptor especificar el tamaño de segmento máximo que está dispuesto a aceptar. ¿Por qué el TCP soporta una opción para especificar el tamaño de segmento máximo cuando también tiene un mecanismo de anuncio de ventana?

13.3 ¿Bajo qué condiciones de retardo, ancho de banda, carga y pérdida de paquetes el TCP retransmitirá volúmenes de datos significativos innecesariamente?

13.4 Los acuses de recibo del TCP perdidos no necesariamente obligan a una retransmisión. Explique por qué.

13.5 Experimente con máquinas locales para determinar como el TCP maneja la reiniciación de una máquina. Establezca una conexión (por ejemplo, un enlace remoto) y déjela inactiva. Espere a que la máquina de destino interrumpa y reinicialice, y entonces fuerce a la máquina local a enviar un segmento TCP (por ejemplo, tecleando caracteres hacia la conexión remota).

13.6 Imagine una implantación de TCP que descarte segmentos que llegan fuera de orden, incluso si éstos caen en la ventana actual. Esto es, la versión imaginada sólo aceptará segmentos que extiendan el flujo de octetos que ya ha recibido. ¿Trabajará? ¿Cómo se compara con una implantación TCP estándar?

13.7 Considere el cálculo de una suma de verificación TCP. Asuma que, aun cuando el campo de suma de verificación en el segmento *no* ha sido puesto en 0, el resultado del cómputo de la suma de verificación es 0. ¿Qué se puede concluir de ello?

13.8 ¿Cuáles son los argumentos a favor y en contra del cierre automático de una conexión inactiva?

13.9 Si dos programas de aplicación utilizan el TCP para enviar datos, pero sólo envían un carácter por segmento (por ejemplo, utilizando la operación PUSH), ¿cuál es el máximo porcentaje del ancho de banda de la red que se tendrá para los datos?

13.10 Supongamos que una implantación del TCP emplea un número de secuencia inicial *1* cuando se hace una conexión. Explique cómo un sistema que ha sido interrumpido y reiniciado puede confundir un sistema remoto en el supuesto de que la conexión anterior se mantiene abierta.

13.11 Observe el algoritmo de estimación de tiempo de viaje redondo sugerido, en la especificación del protocolo ISO TP-4, y compárelo con el algoritmo TCP analizado en este capítulo. ¿Cuál preferiría utilizar?

13.12 Averigüe cómo deben resolver las implantaciones del TCP el *problema de la superposición de segmentos.* El problema se presenta porque el receptor debe recibir sólo una copia de todos los octetos desde el flujo de datos, incluso si el emisor transmite dos segmentos que parcialmente se sobreponen uno sobre otro (por ejemplo, el primer segmento transfiere los octetos 100 a 200 y un segundo transporta los octetos 150 a 250).

13.13 Siga la trayectoria de las transiciones de la máquina de estado finito TCP para las localidades que ejecutan una apertura pasiva y activa, asimismo siga los pasos a través del saludo de tres etapas.

13.14 Lea la especificación TCP para encontrar las condiciones exactas bajo las que el TCP puede hacer la transición de *FIN WAIT-1* hacia *TIME WAIT.*

13.15 Siga las transiciones de estado del TCP para dos máquinas que acuerdan cerrar una conexión cortésmente.

13.16 Supongamos que el TCP está enviando segmentos mediante un tamaño de ventana máximo (64 Gigaoctetos) en un canal que tiene un ancho de banda infinito y un tiempo de viaje redondo promedio de 20 milisegundos. ¿Cuál es el máximo desempeño? ¿Cómo cambiará el desempeño si el tiempo de viaje redondo se incrementa a 40 milisegundos (mientras que el ancho de banda se mantiene infinito)?

13.17 ¿Podría usted derivar una ecuación que exprese el desempeño máximo posible del TCP como una función del ancho de banda de la red, el retardo de la red y el tiempo para procesar un segmento y generar un acuse de recibo? Sugerencia: considere el ejercicio anterior.

13.18 Describa las circunstancias (anormales) por las que el extremo de una conexión puede quedar indefinidamente en un estado *FIN WAIT-2.* Sugerencia: piense en pérdidas de datagramas y en caídas de sistemas.

14

Ruteo: núcleos, pares y algoritmos (GGP)

14.1 Introducción

En los capítulos anteriores nos concentramos en los servicios a nivel de red que ofrece el TCP/IP y los detalles de los protocolos en los anfitriones y ruteadores que proporcionan este servicio. En los análisis anteriores asumimos que los ruteadores siempre tenían rutas correctas y observamos que los ruteadores podían solicitar directamente a los anfitriones, a los que estaban conectados, que cambiaran las rutas mediante el mecanismo de redireccionamiento ICMP.

En este capítulo, se consideran dos preguntas generales: "¿qué valores debe contener una tabla de ruteo?" y "¿cómo se obtienen tales valores?" Para responder la primera pregunta consideraremos la relación entre la arquitectura de red de redes y el ruteo. En particular, analizaremos la estructura de las redes de redes construidas alrededor de una columna vertebral y también compuestas por varias redes pares (*peer networks*), asimismo consideraremos las consecuencias de esto para el ruteo. Si bien varios de nuestros ejemplos provienen de la red global Internet, las ideas se aplican de igual forma a las pequeñas redes de redes corporativas. Para responder a la segunda pregunta, consideraremos los dos tipos básicos de algoritmos de difusión de rutas y veremos cómo cada uno proporciona información de ruteo de manera automática.

Comenzaremos por analizar el ruteo en general. En las secciones posteriores nos concentraremos en la arquitectura de red de redes y describiremos los tipos de protocolos de ruteo utilizados para intercambiar información de ruteo. En los capítulos 15 y 16 se amplía el estudio del ruteo. En estos capítulos se explora los protocolos propiedad de dos grupos administrativos independientes, utilizados para intercambiar información, y los protocolos que un solo grupo utiliza entre todos sus ruteadores.

14.2 Origen de las tablas de ruteo

Recordemos del capítulo 3 que los ruteadores IP proporcionan interconexiones activas entre las redes. Cada ruteador está conectado a dos o más redes físicas y envía datagramas IP entre éstas, acepta datagramas que llegan por medio de una interfaz de red y los rutea hacia otra interfaz. Excepto para los destinos conectados directamente a la red, los anfitriones pasan todo el tráfico IP hacia los ruteadores, los cuales envían los datagramas hacia su destino final. Un datagrama viaja de un ruteador a otro hasta encontrar un ruteador que se encuentre conectado directamente a la misma red en la que se ubica su destino final. Así, el sistema de ruteo forma la arquitectura básica de una red de redes y maneja todo el tráfico, excepto en el caso de las entregas directas de un anfitrión a otro.

En el capítulo 8 se describió el algoritmo de ruteo IP que los anfitriones y los ruteadores siguen para enviar datagramas y se mostró cómo utilizan los algoritmos una tabla para tomar decisiones de ruteo. Cada introducción de información en la tabla de ruteo especifica la porción de red de una dirección de destino y establece la dirección de la siguiente máquina a lo largo de una ruta utilizada para alcanzar la red. Como en el caso de los anfitriones, los ruteadores entregan directamente datagramas a su destino en la red a la que el ruteador está conectado.

Aun cuando hemos visto las bases del envío de datagramas, no hemos dicho cómo obtienen los anfitriones o los ruteadores la información para sus tablas de ruteo. Este problema tiene dos aspectos: *¿qué* valores deben colocarse en las tablas y *cómo* obtienen los ruteadores estos valores? Ambas elecciones dependen de la complejidad de la arquitectura y del tamaño de la red, así como de las políticas administrativas.

En general el establecimiento de rutas comprende procesos de iniciación y actualización. Cada ruteador debe establecer un conjunto inicial de rutas cuando es iniciado y debe actualizar las tablas cuando las rutas cambian (por ejemplo, cuando una interfaz de red falla). La iniciación depende del sistema operativo. En algunos sistemas el ruteador lee una tabla de ruteo inicial desde un almacenamiento secundario en el proceso de iniciación, haciéndola residente en la memoria principal. En otros casos, el sistema operativo comienza con una tabla vacía que debe llenarse ejecutando comandos explícitos (por ejemplo, comandos que se encuentran en un archivo de comandos de iniciación). Finalmente, algunos sistemas operativos comienzan por deducir un conjunto inicial de rutas del conjunto de direcciones para la red local a la que la máquina está conectada, y se ponen en contacto con las máquinas vecinas para solicitar rutas adicionales.

Una vez que se ha construido una tabla de ruteo inicial, un ruteador debe adaptarse a los cambios en las rutas. En los casos de cambios pequeños y lentos en la red de redes, los administradores pueden establecer y modificar rutas a mano. Sin embargo, para el caso de ambientes extensos que cambian rápidamente, la actualización manual es imposible. En este caso, son necesarios métodos automatizados.

Antes de poder entender los protocolos de actualización automática de las tablas de ruteo que se utilizan en los ruteadores IP, necesitamos revisar varias ideas subyacentes. Esto se hará en las siguientes secciones, en las que proporcionaremos las bases conceptuales necesarias para el ruteo. En secciones posteriores, trataremos la arquitectura de red de redes y los protocolos de ruteo empleados para intercambiar información de ruteo.

14.3 Ruteo con información parcial

La diferencia principal entre los ruteadores y los anfitriones comunes es que los anfitriones por lo general saben poco acerca de la estructura de la red de redes a la que están conectados. Los anfitriones no tienen un conocimiento completo de todas las direcciones de destino o todas las redes de destino posibles. De hecho, muchos anfitriones tienen sólo dos rutas en su tabla de ruteo: una para la red local y otra por omisión hacia un ruteador cercano. El ruteador envía todos los datagramas no locales hacia el ruteador local para su entrega. El punto a considerar es el siguiente:

> *Un anfitrión puede rutear datagramas exitosamente aun cuando sólo cuente con información de ruteo parcial ya que puede basarse en un ruteador.*

¿Un ruteador también puede rutear datagramas sólo con información parcial? Sí, pero únicamente bajo ciertas circunstancias. Para entender esto, imagine a una red de redes como a un país atravesado por carreteras polvorientas que cuentan con señales de direccionamiento en las intersecciones. Imagine, por otra parte, que usted no tiene mapas, no puede preguntar nada porque no puede hablar el idioma local, tampoco tiene idea de posibles puntos de referencia visibles, pero usted necesita viajar hacia una villa llamada *Sussex*. Comienza su jornada siguiendo la única carretera que sale de la población y poco a poco va viendo las señales de direccionamiento. En la primera señal encuentra el siguiente letrero:

Norfolk hacia la izquierda; Hammond hacia la derecha; para cualquier otra siga en línea recta.[1]

Como el destino que usted busca no está nombrado explícitamente, tendrá que continuar en línea recta. En la jerga del ruteo se dice que está siguiendo *una ruta por omisión*. Luego de varios señalamientos más, finalmente usted encuentra uno en el que puede leer:

Essex hacia la izquierda; Sussex hacia la derecha; para cualquier otra siga en línea recta.

Usted da vuelta hacia la derecha siguiendo varios señalamientos más y llega hasta una carretera que desemboca en Sussex.

Nuestro viaje imaginario es análogo a la travesía de un datagrama en la red de redes y los señalamientos en la carretera son semejantes a las tablas de ruteo en los ruteadores a lo largo del camino. Sin un mapa u otra ayuda de dirección, completar el viaje dependerá completamente de los señalamientos de la carretera —como el ruteo de un datagrama en una red de redes depende de las tablas de ruteo. Está claro que es posible completar el recorrido aun cuando cada señalamiento en la carretera contenga sólo información parcial.

Una pregunta central nos permitirá introducir algunas precisiones. Como viajero, podría preguntarse: "¿cómo puedo estar seguro de que, al seguir los señalamientos, llegaré a mi destino final?" También, podría preguntarse "¿cómo puedo estar seguro de que al seguir los señalamientos llegaré hasta mi destino, por la ruta más corta?" Esta pregunta podría parecer especialmente molesta si usted pasa frente a varios señalamientos sin encontrar su destino mencionado de manera explícita. Por supuesto, las respuestas dependerán de la topología del sistema de carreteras y del contenido de

[1] Afortunadamente, los señalamientos están impresos en un lenguaje que usted puede comprender.

los señalamientos, pero la idea fundamental es que, tomada en conjunto, la información de los señalamientos debe ser consistente y completa. Considerando esto de otra manera, vemos que no es necesario que en cada intersección haya un señalamiento para todo destino. Los señalamientos pueden listar trayectorias por omisión, señalar explícitamente hacia todos los puntos a lo largo de una trayectoria corta, y los cambios también estar marcados hacia las trayectorias cortas que conducen a todos los destinos. Unos cuantos ejemplos nos permitirán entender algunas de las formas en las que se puede lograr la consistencia.

En un extremo, consideremos una topología simple de estrella formada por carreteras, en la que cada ciudad tiene exactamente una carretera que conduce hasta ella y todas estas carreteras convergen en un punto central. Para garantizar la consistencia, el señalamiento en la intersección central deberá contener información acerca de todos los destinos posibles. En el otro extremo, imagine un conjunto indeterminado de carreteras con señalamientos en todas las intersecciones y las cuales listan todos los destinos posibles. Para garantizar la consistencia, debe ser cierto que, en cualquier intersección, si el señalamiento para el destino D apunta hacia la carretera R, ninguna otra carretera además de R conduce hacia una trayectoria más corta que conduzca a D.

Ninguno de estos extremos arquitectónicos trabaja bien para un sistema de ruteo de red de redes. Por un lado, la intersección central producirá fallas pues ninguna máquina es lo suficientemente rápida como para servir de interruptor central a través del cual pase todo el tráfico. Por otro lado, tener información sobre todos los destinos posibles en todos los ruteadores no sería algo práctico ya que se requeriría difundir grandes volúmenes de información cada vez que se diera un cambio, o cada vez que un administrador necesitara verificar la consistencia. Luego pues, buscamos una solución que permita a los grupos manejar ruteadores locales autónomos añadiendo interconexiones de redes nuevas y rutas sin cambiar ruteadores distantes.

Para ayudar a comprender la arquitectura descrita más adelante, consideremos una tercera topología en la que la mitad de una ciudad se encuentra en la parte oriental y la otra mitad en la occidental. Supongamos que un solo puente cruza el río que separa al Este del Oeste. Imaginemos que las personas que viven en la parte Este no simpatizan con las que viven en la parte Oeste, de tal manera que están deseosas de permitir que los señalamientos de las carreteras indiquen los destinos del Este y no los del Oeste. Supongamos que la gente que vive en el Oeste hace lo mismo en su lado. El ruteo será consistente si todos los señalamientos de las carreteras en el Este señalan hacia los destinos del lado Este explícitamente y apuntan hacia el puente como una ruta por omisión; en tanto que todos los señalamientos de la carretera en el Oeste señalen hacia los destinos del Oeste de manera explícita y apunten hacia el puente como una ruta por omisión.

14.4 Arquitectura y núcleos de Internet originales

La mayor parte del conocimiento sobre el ruteo y los protocolos de difusión de rutas se han derivado de la experiencia con la red global de Internet. Cuando el TCP/IP fue desarrollado por primera vez, las localidades de investigación participantes estaban conectadas a ARPANET, la cual servía como columna vertebral de la red de Internet. Durante los experimentos iniciales, cada localidad administraba tablas de ruteo e instalaba rutas hacia otros destinos a mano. Como Internet comenzaba a crecer, se hizo evidente que el mantenimiento manual de rutas no era práctico; fueron necesarios pues mecanismos automatizados.

Los diseñadores de Internet seleccionaron una arquitectura de ruteo consistente en pequeños conjuntos centrales de ruteadores que contaban con información completa sobre todos los destinos posibles y un gran conjunto de ruteadores externos que contaba con información parcial. En términos de nuestra analogía es como si se designara a un pequeño conjunto de intersecciones localizadas centralmente para tener señalamientos que listaran todos los destinos y se permitiera a las intersecciones exteriores listar únicamente a los destinos locales. A lo largo de la ruta por omisión, en cada punto de intersección exterior hacia una de las intersecciones centrales, los viajeros finalmente encontrarían su destino. La ventaja de usar información parcial en los ruteadores exteriores es que permite a los administradores locales manejar cambios estructurales locales sin afectar otras partes de Internet. La desventaja es que esto introduce la posibilidad de inconsistencias. En el peor de los casos, un error en un ruteador externo puede hacer que los ruteadores distantes sean inaccesibles.

Podemos resumir estas ideas de la siguiente forma:

> *La tabla de ruteo en un ruteador dado contiene información parcial relacionada con destinos posibles. El ruteo que emplea información parcial permite que las localidades tengan autonomía para hacer cambios locales de ruteo, pero introduce la posibilidad de que se den inconsistencias, con las que algunos destinos podrían volverse inaccesibles para algunas fuentes.*

Las inconsistencias entre las tablas de ruteo por lo general son errores en los algoritmos que computan las tablas de ruteo, información incorrecta proporcionada a estos algoritmos, o errores originados cuando se transmiten los resultados hacia otros ruteadores. Los diseñadores de protocolos buscan la forma de limitar el impacto de los errores con el propósito de hacer que todas las rutas sean consistentes en todo momento. Si las rutas se hacen inconsistentes por alguna razón, el protocolo de ruteo debe ser lo suficientemente poderoso para detectar y corregir los errores con rapidez.

14.5 Ruteadores de núcleo

En términos generales, los primeros ruteadores de Internet podrían dividirse en dos grupos, un pequeño conjunto de *ruteadores de núcleo*, controlados por el Internet Network Operations Center (INOC) y un conjunto extenso de *ruteadores no-núcleo*,[2] controlados por grupos individuales. El sistema de núcleo estaba diseñado para proporcionar rutas autorizadas consistentes y confiables para todos los destinos posibles; era el pegamento que sostenía unido a Internet y hacia posible la interconexión universal. Por desgracia, cada localidad asignada a una dirección de Internet debía arreglarse para anunciar su dirección hacia el sistema de núcleo. Las rutas de núcleo estaban comunicadas entre ellas, de esta forma podían garantizar que la información que compartían fuera consistente. Dado que una autoridad central monitoreaba y controlaba los ruteadores de núcleo, éstos eran altamente confiables.

Para entender el sistema de ruteadores de núcleo es necesario recordar que Internet evolucionó a partir de una red de área amplia que ya estaba instalada, ARPANET. Cuando comenzaron los experimentos de Internet, los diseñadores la construyeron a través de ARPANET como una red de

[2] También se han aplicado las expresiones *stub router* y *nonrouting router* a los ruteadores que conectan redes de área local con ARPANET.

columna vertebral principal. Por ello, gran parte de la motivación del sistema de ruteo de núcleo proviene del deseo de conectar redes locales con ARPANET. La figura 14.1 ilustra esta idea.

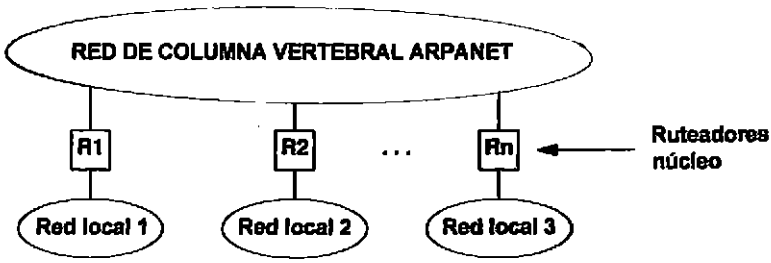


Figura 14.1 El sistema de ruteo de núcleo visto como un conjunto de ruteadores que conectan redes de área local con ARPANET. Los anfitriones en la red local pasan todo el tráfico no local hacia la ruta de núcleo cercana.

Para entender por qué esta arquitectura no conduce en sí a un ruteo con información parcial, supongamos que una extensa red de redes consiste completamente en redes de área local, cada una de ellas conectada a una columna vertebral de la red a través de un ruteador. También imaginemos que algunos de estos ruteadores dependen de rutas por omisión. Luego consideremos la trayectoria de flujo de un datagrama. En la localidad fuente, el ruteador local verifica si hay una ruta explícita hacia el destino y, si no es así, envía el datagrama hacia la trayectoria especificada en su ruta por omisión. Todos los datagramas, para los que el ruteador no tiene una ruta siguen la misma ruta por omisión hacia su destino final. El siguiente ruteador, a lo largo de la trayectoria, desvía datagramas para los que tiene una ruta explícita y envía el resto hacia la ruta por omisión. Para lograr una consistencia global completa, la cadena de rutas por omisión debe alcanzar cualquier ruta en un sitio gigantesco como lo muestra la figura 14.2. Es por ello que la arquitectura requiere que todas las localidades coordinen sus rutas por omisión. Además, dependiendo de las rutas por omisión éstas

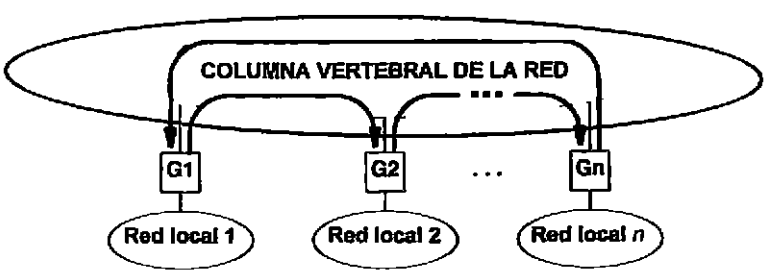


Figura 14.2 Conjunto de ruteadores conectados a una columna vertebral de la red que muestra rutas por omisión. El ruteo es ineficiente, aun cuando sea consistente.

pueden ser ineficientes aun cuando sean consistentes. Como se muestra en la figura 14.2, un datagrama, en el peor de los casos, pasará a través de n ruteadores conforme viaje de la fuente al destino en lugar de ir directamente a través de columna vertebral de la red.

Para evitar la ineficiencia que ocasionan las rutas por omisión, los diseñadores de Internet arreglaron todos los ruteadores de núcleo para intercambiar información de ruteo, de manera que cada uno tenga información completa acerca de las rutas óptimas hacia todos los destinos posibles. Debido a que cada ruteador de núcleo conoce las rutas hacia todos los destinos posibles, no es necesaria una ruta por omisión. Si la dirección de destino en un datagrama no aparece en la tabla de ruteo de un ruteador de núcleo, el ruteador generará un mensaje de destino inalcanzable, ICMP, y eliminará el datagrama. En esencia, el diseño de núcleo evita la ineficiencia al eliminar las rutas por omisión.

La figura 14.3 describe las bases conceptuales de una arquitectura de ruteo de núcleo. La figura muestra un sistema de núcleo central consistente en uno o más ruteadores de núcleo y un conjunto de ruteadores exteriores en sitios locales. Los ruteadores exteriores toman información relacionada con los destinos locales y utilizan una ruta por omisión por la que envían datagramas destinados por otras localidades hacia el núcleo.

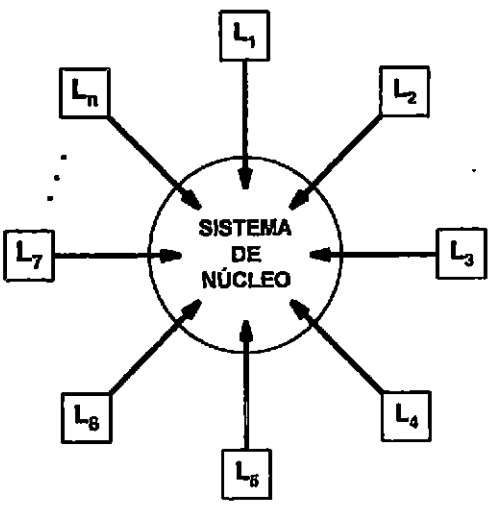


Figura 14.3 Arquitectura de ruteo de un sistema de ruteo simple que muestra las rutas por omisión. Los ruteadores núcleo no utilizan rutas por omisión; los ruteadores externos, señalados como L_i, tienen cada uno una ruta por omisión que apunta hacia el núcleo.

Aun con la simple ilustración de la arquitectura de núcleo mostrada en la figura 14.3 es fácil entender que resulta impráctica por tres razones. En primer lugar, Internet crecería como una sola red de columna vertebral de gran alcance administrada centralmente. La topología se haría compleja y los protocolos necesarios para mantener la consistencia entre ruteadores de núcleo también se harían más complejos. Segundo, no todas las localidades pueden tener un ruteador de núcleo conectado hacia una red de columna vertebral, de manera que resultaría necesaria una estructura adicional de ruteo y protocolos. Tercero, como todos los ruteadores de núcleo interactúan para asegurar la consistencia de

la información de ruteo, la arquitectura de núcleo no podría extenderse a gran escala. Regresaremos a este problema en el capítulo 15 luego de examinar el protocolo que se vale del sistema de núcleo para intercambiar información de ruteo.

14.6 Más allá de la arquitectura de núcleo, hasta las columnas vertebrales pares

La introducción de la columna vertebral de la red NSFNET dentro de Internet añadió una nueva complejidad a la estructura de ruteo. Desde el punto de vista del sistema de núcleo, la conexión de NSFNET no era diferente a la conexión de cualquier otra localidad. NSFNET estaba conectada a la columna vertebral de la red ARPANET a través de un solo ruteador en Pittsburg. El núcleo tenía rutas explícitas hacia todos los destinos en NSFNET. Los ruteadores dentro de NSFNET conocían los destinos locales y utilizaban una ruta por omisión para enviar todo el tráfico que no fuera de NSFNET hacia el núcleo por medio del ruteador de Pittsburg.

Conforme NSFNET crecía, hasta convertirse en la mayor parte de Internet, quedó claro que la arquitectura de ruteo de núcleo no sería suficiente. El cambio conceptual más importante se dio cuando múltiples conexiones se añadieron entre las columnas vertebrales de las redes ARPANET y NSFNET. Diremos que las dos comenzaron a ser *columnas vertebrales de parejas* o simplemente *pares*. La figura 14.4 ilustra el resultado de la topología de pares.

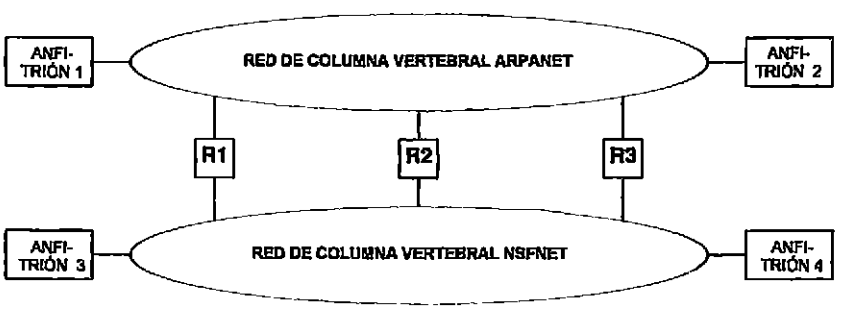


Figura 14.4 Ejemplo de redes pares interconectadas a través de varios ruteadores. El diagrama ilustra la arquitectura de Internet en 1989.

Para comprender las dificultades del ruteo entre redes con columnas vertebrales pares, consideremos las rutas del anfitrión *3* al anfitrión *2* en la figura 14.4. Supongamos, por el momento, que la figura muestra la orientación geográfica. Así pues, el anfitrión *3* está en la Costa Oeste conectado a la red de columna vertebral NSFNET en tanto que el anfitrión *2* está en la Costa Este conectado a la columna vertebral ARPANET. Cuando establecen rutas entre los anfitriones *3* y *2*, los administradores deben decidir si: a) rutear el tráfico desde el anfitrión *3* por medio del ruteador *R1* en la Costa

Oeste y luego a través de la columna vertebral de la red ARPANET; b) rutear el tráfico desde el anfitrión *3* a través de la columna vertebral de la red NSFNET mediante el ruteador *R2* en el Medio Oeste y luego atravesar la columna vertebral de la red ARPANET hasta el anfitrión *2*; o bien c) rutear el tráfico a través de la columna vertebral de la red NSFNET, del ruteador *R3* de la Costa Este, hasta el anfitrión *2*. Es posible incluso un circuito de ruta más: el tráfico puede fluir desde el anfitrión *3* a través del ruteador de la Costa Oeste, a través de la columna vertebral de la red ARPANET hasta el ruteador del Medio Oeste, ir de regreso hacia la columna vertebral de la red NSFNET hacia el ruteador de la Costa este y finalmente, a través de la columna vertebral de la red ARPANET, hasta el anfitrión *2*. Cada ruta puede o no ser accesible, dependiendo de las políticas en uso para la red y de la capacidad de los ruteadores y de las columnas vertebrales de las redes.

Para la mayor parte de las configuraciones de las columnas vertebrales pares (o de pareja), el tráfico entre un par de anfitriones cercanos geográficamente puede seguir una ruta corta independientemente de las rutas seleccionadas para el tráfico a través del país. Por ejemplo, el tráfico del anfitrión *3* hacia el *1* puede fluir a través del ruteador de la Costa Este porque minimiza las distancias entre ambas columnas vertebrales de las redes.

Todas estas afirmaciones suenan bastante simples, pero su implantación es muy compleja por dos razones. En primer lugar, aun cuando los estándares de los algoritmos de ruteo de IP utilizan la porción de red de una dirección IP para seleccionar una ruta, el ruteo óptimo en una arquitectura de columna vertebral de pares requiere de ruteos individuales para anfitriones individuales. En el ejemplo anterior, la tabla de ruteo en el anfitrión *3* necesita diferentes rutas para el anfitrión *1* y *2*, aun cuando los anfitriones *1* y *2* estén conectados a la columna vertebral de la red ARPANET. En segundo lugar, los administradores de ambas columnas vertebrales de las redes deben acordar el establecimiento de rutas consistentes entre todas las rutas o pueden desarrollarse *ciclos cerrados de ruteo* (un ciclo cerrado de ruteo se da cuando las rutas de un conjunto de ruteadores forman un ciclo).

Es importante distinguir la topología de una red de la arquitectura de ruteo. Es posible, por ejemplo, tener un solo sistema de núcleo que abarque varias redes de columna vertebral. Las máquinas núcleo pueden programarse para ocultar los detalles arquitectónicos subyacentes y para computar las rutas más cortas entre ellas. No es posible, sin embargo, dividir el sistema de núcleo en subconjuntos en los que cada uno conserve información parcial sin perder funcionalidad. La figura 14.5 ilustra el problema.

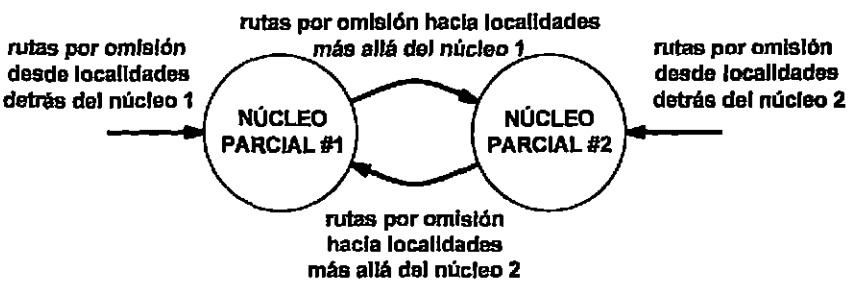


Figura 14.5 Método de partición de una arquitectura de ruteo de núcleo en dos conjuntos de ruteadores que conservan información parcial y utilizan rutas por omisión. Una arquitectura como esta puede producir ciclos cerrados de ruteo para datagramas que tienen un destino ilegal (no existente).

Como se muestra en la figura, los ruteadores externos cuentan con rutas por omisión hacia un lado del núcleo dividido. Cada lado de la partición tiene información sobre destinos en el lado del mundo en que está ubicada y una ruta por omisión para el otro lado del mundo. En una arquitectura como ésta, cualquier datagrama enviado a una dirección ilegal entrará en un ciclo entre las dos particiones en un ciclo cerrado de ruteo hasta que el contador de tiempo de vida llegue a cero.

Lo anterior puede resumirse como sigue:

Una arquitectura de ruteo de núcleo requiere de un conjunto centralizado de servidores de ruteo como depósito de información acerca de todos los destinos posibles en una red de redes. El sistema de núcleos trabaja mejor en redes de redes que cuentan con una sola columna vertebral de red administrada centralmente. La expansión de la topología hacia múltiples columnas vertebrales de redes hace el ruteo más complejo; el método de la división de la arquitectura de núcleo, en la que todos los ruteadores utilizan rutas por omisión, introduce la posibilidad de que se desarrollen ciclos cerrados de ruteo.

14.7 Difusión automática de ruta

Hemos dicho que el sistema original de núcleo de Internet evitaba las rutas por omisión porque éstas difunden información completa sobre todos los destinos posibles hacia todos los ruteadores núcleo. Muchas compañías de red de redes utilizan ahora esquemas similares —los ruteadores de las compañías corren programas que comunican información de ruteo. En la siguiente sección, trataremos dos tipos básicos de algoritmos que computan y difunden información de ruteo y utilizaremos el protocolo de ruteo de núcleo original para ilustrar uno de estos algoritmos. En una sección posterior, describiremos un protocolo que emplea el otro tipo de algoritmo.

Podría parecer que el mecanismo de difusión automática de rutas no es necesario, especialmente en redes de redes pequeñas. Sin embargo, las redes de redes no son estáticas. Las conexiones fallan y tardan en ser reemplazadas. Las redes se pueden sobrecargar en un momento dado y subutilizarse después. El propósito del mecanismo de difusión de ruteo no es únicamente encontrar un conjunto de rutas, sino actualizar continuamente la información. Las personas simplemente no pueden responder a los cambios con la rapidez suficiente; se deben utilizar programas de computadora. Así, cuando pensamos en la difusión de rutas es importante considerar el comportamiento dinámico de los protocolos y los algoritmos.

14.8 Ruteo por vector-distancia (Bellman-Ford)

El término *vector-distancia*[3] indica a una clase de algoritmos de ruteo utilizada para difundir información de ruteo. La idea detrás de los algoritmos vector-distancia es muy sencilla. El ruteador esta-

[3] Los nombres *Ford Fulkerson* y *Bellman-Ford* son sinónimos de *vector-distancia*; éstos se tomaron de los nombres de los investigadores que hicieron pública la idea.

blece una lista de todas las rutas conocidas en una tabla. Cuando arranca, un ruteador inicia esta tabla de ruteo para que contenga una entrada de información por cada red conectada directamente. Cada introducción en la red identifica una red de destino y establece una distancia hacia la red, por lo general medida en saltos (más adelante se definirá esto con mayor precisión). Por ejemplo, la figura 14.6 muestra el contenido inicial de la tabla en un ruteador conectado a dos redes.

Destino	Distancia	Ruta
Red 1	0	directa
Red 2	0	directa

Figura 14.6 Tabla de ruteo inicial vector-distancia con una entrada de información para cada red conectada directamente. Cada entrada de la información contiene la dirección IP de una red y un número entero relacionado con la distancia hacia esa red.

Periódicamente, cada ruteador envía una copia de su tabla de ruteo a cualquier otro ruteador que pueda alcanzar de manera directa. Cuando llega un reporte al ruteador *K* desde el ruteador *J*, *K* examina el conjunto de destinos reportados y la distancia de cada uno. Si *J* conoce una ruta más corta para alcanzar un destino o si *J* lista un destino que *K* no tiene en su tabla, o bien si *K* rutea actualmente hacia un destino a través de *J* y la distancia de *J* hacia el destino ha cambiado, *K* actualiza esta información en su tabla. Por ejemplo, la figura 14.7 muestra una tabla existente en un ruteador *K* y un mensaje actualizado desde otro ruteador *J*.

Destino	Distancia	Ruta
Red 1	0	directa
Red 2	0	directa
Red 4	8	Ruteador L
Red 17	5	Ruteador M
Red 24	6	Ruteador J
Red 30	2	Ruteador Q
Red 42	2	Ruteador J

(a)

Destino	Distancia
Red 1	2
Red 4	3
Red 17	6
Red 21	4
Red 24	5
Red 30	10
Red 42	3

(b)

Figura 14.7 (a) Tabla de ruteo existente para un ruteador *K*, y (b) un mensaje entrante de actualización desde el ruteador *J*. Las entradas de información marcadas se utilizarán para actualizar entradas de información existentes o añadir nuevas entradas a la tabla de *K*.

Obsérvese que si *J* reporta una distancia *N*, el dato actualizado en *K* tendrá la distancia *N+1* (la distancia para alcanzar el destino desde *J*, más la distancia para alcanzar *J*). Por supuesto, la tabla de ruteo completa contiene una tercera columna que especifica una ruta. La entrada inicial de datos se marca con el valor *entrega directa* (*direct delivery*). Cuando el ruteador *K* añade o actualiza una entrada de datos en respuesta al mensaje que proviene del ruteador *J*, asigna al ruteador *J* como la ruta para tal dato.

El término *vector-distancia* proviene del hecho de que la información se envía en mensajes periódicos. Un mensaje contiene una lista de pares (*V,D*), donde *V* identifica el destino (llamado *vector*) y *D* es la distancia hacia el destino. Nótese que el algoritmo vector-distancia reporta las rutas en primera persona (pensemos que un ruteador anuncia: "puedo alcanzar el destino *V* que está a la distancia *D*"). En este tipo de diseño, todos los ruteadores deben participar en el intercambio de información de vector-distancia para que las rutas sean eficientes y consistentes.

Aun cuando los algoritmos vector-distancia son fáciles de implementar, tienen desventajas. En un ambiente completamente estático, los algoritmos vector-distancia difunden rutas hacia todos los destinos. Cuando las rutas cambian rápidamente, sin embargo, los cómputos podrían no ser estables. Cuando una ruta cambia (por ejemplo, si aparece una nueva conexión o si una conexión vieja falla), la información se propaga lentamente de un ruteador a otro. Esto significa que algunos ruteadores pueden tener información de ruteo incorrecta.

Por ahora, examinaremos un protocolo que utiliza el algoritmo vector-distancia sin tratar todos sus puntos débiles. En el capítulo 16, se completa el análisis con otro protocolo vector-distancia, los problemas que pueden aparecer y la heurística utilizada para resolver los inconvenientes más serios.

14.9 Protocolo pasarela-a-pasarela (GGP)

Los ruteadores núcleo iniciales utilizaban un protocolo de vector-distancia conocido como *Gateway-to-Gateway Protocol*[4] (*Protocolo pasarela-a-pasarela* o *GGP*, por sus siglas en inglés) para intercambiar información de ruteo. Aunque el GGP no es una parte clave del conjunto de protocolos TCP/IP proporciona un ejemplo concreto de ruteo vector-distancia. El GGP fue diseñado para viajar en datagramas IP de la misma forma que los datagramas UDP o los segmentos TCP. Cada mensaje GGP tiene un encabezado de formato fijo que identifica el tipo de mensaje y el formato de los campos restantes. Dado que sólo los ruteadores núcleo participan en el GGP y que éstos son controlados por INOC, otros ruteadores no pueden interferir en el intercambio.

El sistema de núcleo original fue diseñado para permitir que nuevos ruteadores núcleo se añadieran sin modificar los ruteadores existentes. Cuando se añadía un nuevo ruteador al sistema de núcleo, éste era asignado a uno o más núcleos *vecinos* con los que se comunicaba. Los vecinos, miembros del núcleo, difundían la información de ruteo entre los demás. Así, el nuevo ruteador sólo necesitaba informar a sus vecinos sobre las redes que podía alcanzar; éstos actualizaban las tablas de ruteo y difundían la nueva información.

[4] Recordemos que a través de los vendedores se adoptó el término *ruteador*, originalmente los científicos utilizaron el término *pasarela (gateway) IP*.

El GGP es un verdadero protocolo de vector-distancia. La información de intercambio de rutas en el GGP consiste en un conjunto de pares (*N,D*), donde *N* es una dirección de red IP y *D* una distancia medida en saltos. Puede decirse que un ruteador que utiliza el GGP *anuncia* las redes que puede alcanzar y el costo para alcanzarlas.

El GGP mide las distancias en *saltos de ruteador*, donde un ruteador se define en cero saltos si está conectado directamente a la red, un salto para redes que están conectadas a través de otro ruteador y así sucesivamente. De esta manera, el *número de saltos* o el *conteo de saltos*, a lo largo de una trayectoria de una fuente dada a un destino, se refiere al número de ruteadores que el datagrama encontrará a lo largo de su recorrido. Debería ser obvio que utilizar un contador de saltos, para calcular las trayectorias más cortas, no siempre produce resultados deseables. Por ejemplo, una trayectoria con un conteo de saltos que atraviesa tres redes LAN podría ser notablemente más rápida que una trayectoria con un conteo de dos saltos que atraviesa dos líneas seriales lentas. Muchos ruteadores emplean artificialmente números altos de conteo de saltos para las trayectorias que cruzan redes lentas.

14.10 Formatos de los mensajes GGP

Hay cuatro tipos de mensajes GGP, cada uno con su formato propio. El primer octeto contiene un código que identifica el *tipo* de mensaje. La figura 14.8 muestra el formato de un tipo de mensaje GGP, el mensaje que el ruteador intercambia para aprender acerca de las rutas. Recordemos que la información consiste en pares de redes IP y valores de distancia. Para mantener los mensajes cortos, las redes se agrupan por distancia y el mensaje consiste en una secuencia de conjuntos, donde cada conjunto contiene un valor de distancia seguido por una lista de todas las redes asociadas a esa distancia.

El valor *12* en el campo *TYPE* especifica que este mensaje es un mensaje de *actualización de ruteo*, lo cual lo distingue de otro tipo de mensajes GGP. Los 16 bits de *SEQUENCE NUMBER* (NÚMERO DE SECUENCIA) se utilizan para validar un mensaje GGP; tanto el emisor como el receptor deben acordar una secuencia de números antes de que el receptor acepte el mensaje. El campo *UPDATE* (ACTUALIZACIÓN) es un valor binario que especifica si el emisor necesita una actualización del receptor. Dado que el GGP agrupa las redes por distancia, el campo llamado *NUM DISTANCES* (NÚMERO DE DISTANCIAS) especifica cuántos grupos de distancias se presentan en esa actualización.

La última parte de un mensaje de actualización de ruteo del GGP contiene conjuntos de redes agrupadas por distancia. Cada conjunto comienza con dos campos de ocho bits que especifican un valor de distancia y un contador de redes en esa distancia. Si el contador especifica *n* redes a una distancia dada, exactamente *n* direcciones IP de red deben aparecer antes del encabezado del siguiente conjunto. Para conservar espacio, sólo se incluye la porción de red de la dirección IP, así, los números de red pueden ser de 1, 2 o 3 octetos de longitud. El receptor debe revisar los primeros bits del identificador de red para determinar su longitud.

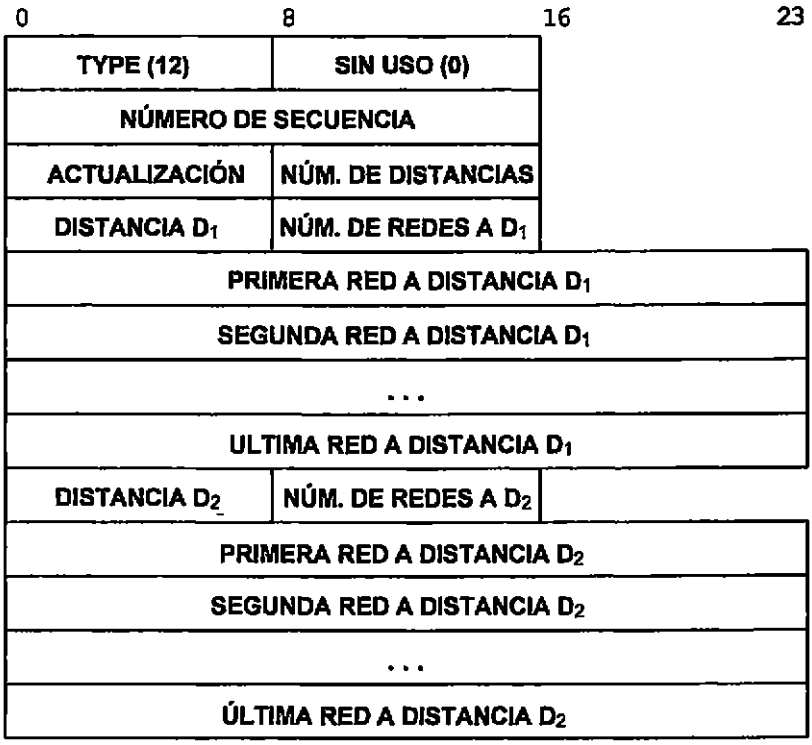


Figura 14.8 Formato de un mensaje de actualización GGP. Un ruteador envía un mensaje para anunciar qué redes de destino puede alcanzar. Los números de red contienen 1, 2 o 3 octetos, dependiendo de si la red es de clase *A*, *B* o *C*.

Cuando un ruteador recibe un mensaje de actualización de ruteo GGP, envía un mensaje de *acuse de recibo* GGP de regreso al emisor, utilizando un acuse de recibo positivo si la actualización de ruteo es aceptada y uno negativo si se detecta un error. La figura 14.9 ilustra el formato de un acuse de recibo GGP.

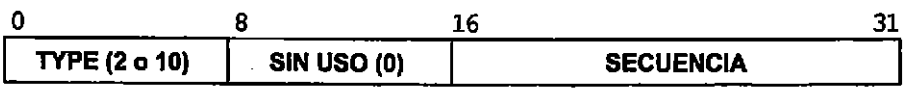


Figura 14.9 Formato de mensaje de acuse de recibo GGP. El valor type *2* identifica el mensaje como un acuse de recibo positivo, mientras que type *10* identifica el mensaje como un acuse de recibo negativo.

Para el caso de un acuse de recibo positivo, el campo *SEQUENCE (SECUENCIA)* especifica un número de secuencia que el receptor está reconociendo. En el caso de un acuse de recibo negativo, el campo *SEQUENCE* conserva el último número de secuencia que el receptor recibió correctamente.

Además de los mensajes de actualización de ruteo, el protocolo GGP incluye mensajes que permiten a un ruteador probar si otros están respondiendo. Un ruteador envía un mensaje de *solicitud de eco* hacia un vecino, el cual solicita que el recipiente responda y envíe de regreso un mensaje *eco de respuesta*. La figura 14.10 muestra el formato de los mensajes de eco.

0	8 … 31
TYPE (0 u 8)	SIN USO (0)

Figura 14.10 Formato de una solicitud de eco o mensaje de réplica GGP. El valor type 8 identifica el mensaje como una solicitud de eco, mientras que el valor type 0 identifica el mensaje como una réplica de eco.

14.11 Ruteo enlace-estado (SPF)

La principal desventaja de los algoritmos vector-distancia es que no se extienden bien. Junto con el problema de la lenta respuesta de cambio mencionado al principio, el algoritmo requiere de intercambios de mensajes largos. Dado que la actualización de los mensajes de ruteo contienen una entrada de información para cada red posible, el tamaño de los mensajes es proporcional al número total de redes en una red de redes. Además, debido a que un protocolo vector-distancia requiere de la participación de todos los ruteadores, el volumen de información a intercambiar puede ser enorme.

La principal alternativa a los algoritmos de vector-distancia es una clase de algoritmos conocidos como *enlace-estado* (*link-state*), *Shortest Path First* (*Primero la ruta más corta*) o *SPF*.[5] Los algoritmos SPF requieren que cada ruteador participante tenga información de la topología completa. La forma más fácil de pensar la información de la topología es imaginando que todos los ruteadores tienen un mapa que muestra a todos los otros ruteadores y las redes a las que están conectados. En términos abstractos, los ruteadores corresponden a los nodos o vértices en un grafo y las redes que conectan a los ruteadores corresponden a los arcos. Hay un arco (enlace) entre dos nodos si y sólo si los correspondientes ruteadores pueden comunicarse directamente.

En lugar de enviar un mensaje que contenga una lista de destinos, un ruteador que participa en un algoritmo SPF desempeña dos tareas. En primer lugar, prueba activamente el estado de todos los ruteadores vecinos. En términos de un grafo, dos ruteadores son vecinos si comparten un enlace; en términos de una red, dos vecinos están conectados a una red común. En segundo lugar, difunde periódicamente la información del estado del enlace hacia los otros ruteadores.

[5] El nombre "primera ruta más corta" es un nombre equivocado y poco afortunado, puesto que la mayor parte de los ruteadores seleccionan las rutas más cortas. Sin embargo, parece que la expresión ha logrado una amplia aceptación.

Para probar el estado de un vecino conectado directamente, un ruteador intercambia de manera periódica mensajes cortos que interrogan si el vecino está activo y conectado. Si el vecino responde, se dice que el enlace entre ellos está levantado ("up"). De otra forma se dice que el enlace está caído ("down"). (En la práctica, para prevenir las oscilaciones entre los estados up y down, varios protocolos utilizan una regla de *n, k fuera* (*k-out-of-n*) para probar la actividad, esto significa que el enlace se mantiene "up" hasta que un porcentaje significativo de solicitudes no tenga réplica, entonces se conserva "down" hasta que un porcentaje significativo de mensajes reciba réplicas.)

Para informar a todos los otros ruteadores, cada ruteador difunde periódicamente un mensaje que lista el estado de cada uno de estos enlaces. El mensaje de estado no especifica rutas —sólo reporta si es posible la comunicación entre pares de ruteadores. El software de protocolo de ruteadores está configurado para entregar una copia de cada mensaje de estado de enlace hacia todos los ruteadores participantes (si la red subyacente no soporta la difusión, la distribución se realiza entregando copias individuales del mensaje punto a punto).

Cada vez que llega un mensaje de estado de enlace, un ruteador utiliza la información para actualizar su mapa de la red de redes haciendo que los enlaces queden "up" o "down". Cada vez que cambia el estado de un enlace, el ruteador computa de nuevo las rutas aplicando el conocido *algoritmo de Dijkstra de la ruta más corta* (*Dijkstra shortest path algorithm*) al grafo resultante. El algoritmo de Dijkstra calcula la trayectoria más corta hacia todos los destinos desde una sola fuente.

Una de las mayores ventajas de los algoritmos SPF es que cada ruteador computa trayectorias independientemente, utilizando la misma información de estado original; esto no depende de un cálculo en máquinas intermedias. Debido a que los mensajes de estado de enlace se difunden sin cambio, es fácil resolver problemas de depuración. Como los ruteadores realizan el cálculo de rutas de manera local, la convergencia está garantizada. Por último, dado que los mensajes de estado de enlace sólo transportan información sobre conexiones directas desde un solo ruteador, su tamaño no depende del número de redes en la red de redes. Así, los algoritmos SPF se extienden mejor que los algoritmos de vector-distancia.

14.12 Protocolos SPF

Junto con lòs protocolos propietarios ofrecidos por los vendedores, sólo unos cuantos protocolos SPF están actualmente en uso en Internet. Uno de los primeros ejemplos de la llegada de SPF proviene de ARPANET, la cual utilizó internamente un algoritmo SPF por casi 10 años. En el otro extremo, en el capítulo 16 se analiza un protocolo SPF de propósito general actualmente en uso en Internet.

Hacia 1988 el sistema de núcleo de Internet cambió de las computadoras iniciales LSI-11 de Digital Equipment Corporation, que corrían el GGP, al procesador *Butterfly* de Balt, Beranek and Newman Computer Corporation que utiliza un algoritmo SPF "primera ruta más corta" (*Shortest Path First*). El protocolo exacto, conocido como *SPREAD*, no está documentado en la información RFC.

14.13 Resumen

Para garantizar que todas las redes se mantengan accesibles con una alta confiabilidad, una red de redes debe proporcionar un ruteo consistente globalmente. Los anfitriones y la mayor parte de los ruteadores contiene sólo información parcial de ruteo; dependen de las rutas por omisión para enviar datagramas a lugares distantes. La red global de Internet resuelve el problema del ruteo mediante el uso de una arquitectura de ruteador núcleo en la que un conjunto de ruteadores núcleo contiene información completa sobre todas las redes. Los ruteadores en el sistema original de núcleo de Internet intercambiaban periódicamente información de ruteo, esto significa que, si un solo ruteador núcleo aprendía algo acerca de una ruta, todos los ruteadores núcleo lo aprendían también. Para prevenir ciclos cerrados de ruteo, los ruteadores núcleo tenían prohibido utilizar rutas por omisión.

Un sistema de núcleo con una administración única y central trabaja bien para una arquitectura de red de redes construida con una sola columna vertebral de red. Sin embargo, cuando una red se compone de varias columnas vertebrales pares administradas por separado, interconectando lugares diversos, la arquitectura de núcleo no es suficiente.

Cuando los ruteadores intercambian información de ruteo por lo general utilizan uno de dos algoritmos básicos: vector-distancia o SPF. Examinamos los detalles del GGP, el protocolo vector-distancia originalmente utilizado para difundir información de actualización de ruteo a través del núcleo. Cada rutina de actualización GGP puede considerarse como un comunicado que lista un conjunto de redes, junto con el costo de ruteo para alcanzar tales redes.

La mayor desventaja de los algoritmos de vector-distancia es que realizan una distribución del cálculo de la ruta más corta que puede no ser convergente si el estado de las conexiones de red cambia a menudo. Otra desventaja es que los mensajes de actualización de ruteo crecen en extensión conforme se incrementa el número de redes.

PARA CONOCER MÁS

La definición del sistema de ruteador núcleo y del protocolo GGP en este capítulo proviene de Hinden y Sheltzer (RFC 823). Braden y Postel (RFC 1009) tratan más especificaciones sobre ruteadores Internet. Almquist (RFC 1716) resume estudios recientes. Braun (RFC 1093) y Rekhter (RFC 1092) analizan el ruteo en la red de columna vertebral NSFNET. Clark (RFC 1102) y Braun (RFC 1104) tratan las políticas basadas en el ruteo. En los siguientes dos capítulos, se presentan los protocolos utilizados para difundir la información de ruteo entre localidades separadas y dentro de una sola localidad.

EJERCICIOS

14.1 Supongamos que un ruteador descubre que, para rutear un datagrama, debe regresarlo por la misma interfaz de red por la que llegó el datagrama. ¿Qué haría? ¿Por qué?

14.2 Luego de leer el documento RFC 823 y el RFC 1009, explique qué hace un ruteador núcleo de Internet en la situación descrita en la pregunta anterior.

14.3 ¿Cómo utilizan los ruteadores en un sistema de núcleo las rutas por omisión para enviar todos los datagramas ilegales hacia una máquina específica?

14.4 Imagine un grupo de estudiantes experimentando con un ruteador que conecta una red de área local con Internet. Quieren anunciar su red al sistema de ruteo de núcleo, pero si accidentalmente anuncian rutas de longitud cero para redes arbitrarias, el tráfico real de Internet puede desviarse de su ruta. ¿Cómo puede el núcleo protegerse de datos ilegales mientras acepta actualizaciones desde ruteadores no confiables?

14.5 ¿Qué mensajes ICMP genera un ruteador?

14.6 ¿Cómo determina un ruteador núcleo original de Internet si un determinado vecino está "up" o "down"? Sugerencias: consulte el documento RFC 823.

14.7 Suponga que dos ruteadores núcleo anuncian el mismo costo, k, para alcanzar una red dada, N. Describa las circunstancias bajo las cuales el recorrido a través de una de ellas puede requerir un número total de saltos menor que el ruteo a través de la otra.

14.8 ¿Cómo sabe un ruteador si un datagrama entrante transporta un mensaje GGP?

14.9 Considere cuidadosamente la actualización del vector-distancia mostrado en la figura 14.7. Dé tres razones por las que el ruteador actualizará esta tabla con los tres aspectos mostrados.

15

Ruteo: sistemas autónomos (EGP)

15.1 Introducción

En el capítulo anterior se introdujo la idea de la propagación de ruta y se examinó un protocolo que los ruteadores utilizan para intercambiar información de ruteo. En este capítulo se amplía el conocimiento sobre la arquitectura de ruteo en una red de redes. Se analiza el concepto de sistemas autónomos y se muestra el protocolo que un grupo de redes y ruteadores operando bajo una autoridad administrativa para difundir información sobre la accesibilidad de redes hacia otros grupos.

15.2 Agregar complejidad al modelo arquitectónico

El sistema original de ruteo de núcleo se desarrolló cuando Internet contaba con una sola columna vertebral (*backbone*). En consecuencia, parte de la motivación de una arquitectura de núcleo fue proporcionar conexiones entre redes de área local y la columna vertebral (ver figura 14.1). Para una red de redes con una sola columna vertebral, más un conjunto de redes de área local conectadas, no es necesaria una estructura adicional. Cada ruteador conoce la única red local a la que está conectado y aprende acerca de todas las otras redes comunicándose a través de la columna vertebral con otros ruteadores. Por desgracia, tener a todos los ruteadores participando de manera directa en un protocolo de actualización de ruteo no es suficiente, salvo para redes de redes triviales. En primer lugar, aun cuando cada localidad conectada a la red de redes tenga sólo una red local, una arquitectura de núcleo es inadecuada ya que ésta no puede crecer para adaptarse a un número arbitrario de localidades. En segundo lugar, la mayor parte de las localidades tiene múltiples

redes de área local y ruteadores interconectados. Como un ruteador núcleo se conecta a una sola red en cada localidad, el núcleo sólo tiene conocimiento acerca de una red en la localidad. En tercer lugar, una red de redes extensa interconecta conjuntos de redes administradas por grupos independientes. Una arquitectura de ruteo debe proporcionar la vía para que cada grupo controle de manera independiente el ruteo y el acceso. Luego de examinar las consecuencias de cada una de estas ideas, aprenderemos cómo un sólo mecanismo de protocolo permite la construcción de una red de redes que abarca varias localidades y permite que éstas conserven su autonomía.

15.3 Una idea fundamental: saltos adicionales (hops)

Hasta aquí hemos analizado la arquitectura de una red de redes con una o más columnas vertebrales conectada por un sistema de ruteo de núcleo. Hemos considerado un sistema de núcleo como un mecanismo de ruteo central al que los ruteadores no-núcleo envían datagramas para su entrega. También hemos dicho que es imposible expandir de manera arbitraria una sola columna vertebral. Tener menos ruteadores núcleo que redes en una red de redes significa que tendremos que modificar nuestra visión de la arquitectura de núcleo o el ruteo será deficiente. Para entender por qué, consideremos el ejemplo de la figura 15.1

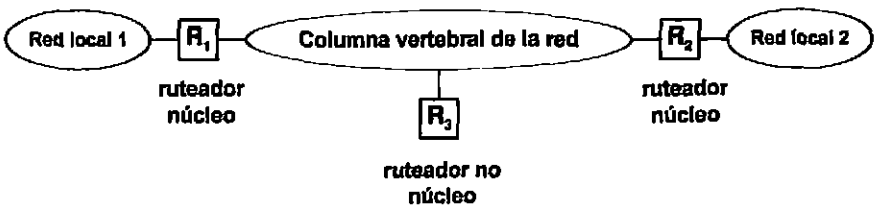


Figura 15.1 Problema del salto extra. Los ruteadores que no pertenecen al núcleo, conectados a la columna vertebral de la red, deben aprender rutas desde los ruteadores núcleo para tener un ruteo óptimo.

En la figura, los ruteadores núcleo R_1 y R_2 conectan las redes de área local *1* y *2* respectivamente. Dado que intercambian información de ruteo, ambos ruteadores saben cómo alcanzar ambas redes. Supongamos que el ruteador R_3, no-núcleo, considera al núcleo como un sistema de entrega y selecciona a uno de los ruteadores núcleo, digamos R_1, para entregar todos los datagramas destinados a las redes con las que no tiene contacto. R_3 envía datagramas para la red *2* a través de la columna vertebral de la red, para esto selecciona al ruteador núcleo R_1 el cual debe enviarlos de regreso, a través de la columna vertebral al ruteador R_2. La ruta óptima, por supuesto, requeriría que R_3 enviara los datagramas destinados a la red 2 directamente hacia R_2. Obsérvese que la selección de ruteador núcleo no da resultados diferentes. Sólo los destinos que se encuentran más allá del ruteador seleccionado tienen rutas óptimas; todas las trayectorias que atraviesan otros ruteadores de la columna vertebral requieren de un salto extra. Nótese también que el ruteador núcleo no puede utilizar mensajes de redireccionamiento ICMP para informar a R_3 que tiene una ruta incorrecta porque los mensajes de

redireccionamiento ICMP sólo pueden enviarse hacia la fuente original y no hacia ruteadores intermedios.

La anomalía de ruteo ilustrada en la figura 15.1 se conoce como *problema del salto extra* (*extra hop*). Para resolverlo será necesario que modifiquemos nuestra visión de la arquitectura de núcleo:

> *Tratar un sistema de núcleo como un ruteador central introduce un salto extra en la mayor parte del tráfico. Se necesita un mecanismo que permita, a los ruteadores que no pertenecen al núcleo, aprender rutas desde los ruteadores núcleo de manera que puedan seleccionar rutas óptimas en la columna vertebral de la red.*

Permitir que las localidades tengan múltiples redes y ruteadores significa que el núcleo no está conectado a todas las redes de manera directa, de tal forma que es necesario un mecanismo adicional para permitir que el sistema de núcleo aprenda esto. Consideremos, por ejemplo, el conjunto de redes y ruteadores mostrados en la figura 15.2. Podemos imaginar una conexión como ésta en una compañía o en un campus universitario, donde cada red corresponde a un solo edificio o a un solo departamento.

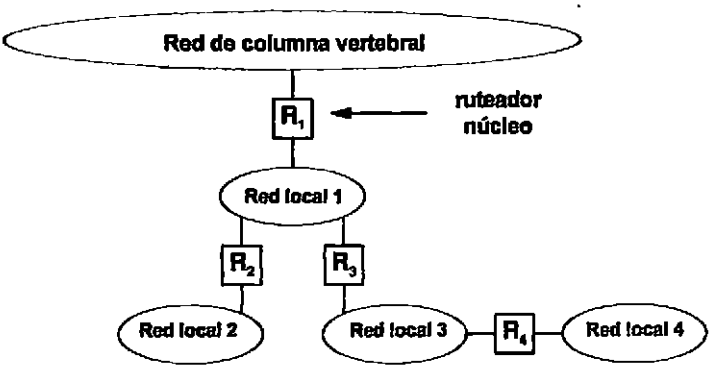


Figura 15.2 Ejemplo de múltiples redes y ruteadores con una sola conexión de columna vertebral de red. Se necesita un mecanismo para transferir la información de accesibilidad de las redes locales adicionales hacia el sistema de núcleo.

Supongamos que la localidad tiene instalada sólo la red local *4* y que ha obtenido una dirección de red de redes para ésta. También imaginemos que los ruteadores R_2, R_3 y R_4 tienen rutas para la cuatro redes locales, así como rutas por omisión que pasan el tráfico externo hacia el ruteador núcleo R_1. Los anfitriones directamente conectados a la red local *4* pueden comunicarse con otras redes y cualquier máquina puede rutear paquetes hacia el exterior para otras localidades de la columna vertebral de la red. Sin embargo, debido a que el ruteador R_1 está conectado sólo a la red local *1*, no puede tener conocimiento acerca de la red local *4*. Decimos que, desde el punto de vista del sistema de núcleo, la red local *4* está *oculta* detrás de la red local *1*. El punto importante aquí es que:

Como las localidades individuales pueden tener una estructura de complejidad arbitraria, un sistema de núcleo no se conecta directamente hacia todas las redes. Es necesario un mecanismo que permita a los ruteadores no-núcleo, informar al núcleo sobre redes ocultas

Recordemos que, además de proveer al núcleo con información sobre redes ocultas, necesitamos un mecanismo que permita a los ruteadores no-núcleo obtener información de ruteo desde el núcleo. Idealmente, un solo mecanismo debería resolver ambos problemas. La construcción de este mecanismo puede ser una tarea compleja y sutil. Los puntos a cuidar, en relación con esto, son la responsabilidad y la capacidad. ¿Exactamente en qué radica la responsabilidad de informar al núcleo? Si decidimos que uno de los ruteadores debe informar al núcleo, ¿cuál de ellos será capaz de hacerlo? Veamos de nuevo el ejemplo. El ruteador R_4 es el ruteador más cercano asociado a la red local *4*, pero éste se encuentra a dos saltos del ruteador núcleo más cercano. Así, R_4 debe depender del ruteador R_3 para rutear paquetes hacia la red *4*. El punto es que R_4 no puede garantizar la accesibilidad de la red local *4* por sí mismo. El ruteador R_3 se encuentra a un salto del núcleo y puede garantizar el paso de paquetes, pero no está directamente conectado hacia la red local *4*. Así, parece incorrecto otorgar a R_3 la responsabilidad de la red *4*. Resolver este dilema requerirá que introduzcamos un nuevo concepto. Las siguientes secciones tratan acerca de este concepto y de un protocolo construido en torno a él.

15.4 Concepto de los sistemas autónomos

El rompecabezas sobre el cual los ruteadores deben comunicar información de accesibilidad para los sistemas de núcleo se presenta debido a que hemos considerado únicamente la mecánica de la arquitectura de ruteo en una red de redes y no hemos considerado los aspectos administrativos. Las interconexiones, como las que se muestran en el ejemplo de la figura 15.2, que aparecen cuando una localidad en la columna vertebral de la red tiene una compleja estructura local, no deben ser pensadas como una red independiente múltiple, conectada hacia una red de redes, sino como una organización única que tiene múltiples redes bajo su control. Dado que las redes y los ruteadores se encuentran bajo una sola autoridad administrativa, esta autoridad puede garantizar que las rutas internas se mantengan consistentes y viables; más aún, la autoridad administrativa puede seleccionar a una de sus máquinas para servir como la máquina que aparecerá ante el mundo exterior como el acceso hacia la red. En el ejemplo de la figura 15.2, dado que los ruteadores R_2, R_3 y R_4 están bajo el control de una autoridad administrativa se puede arreglar que R_3 anuncie la accesibilidad para las redes *2*, *3* y *4* (asumimos que el sistema de núcleo ya tiene conocimiento sobre la red *1* ya que un ruteador núcleo está conectado directamente a ésta).

Para propósitos de ruteo a un grupo de redes y ruteadores controlados por una sola autoridad administrativa se le conoce como *sistema autónomo*. Los ruteadores dentro de un sistema autónomo son libres de seleccionar sus propios mecanismos de exploración, propagación, validación y verificación de la consistencia de las rutas. Nótese que bajo esta definición el ruteador núcleo en sí forma un sistema autónomo. Hemos dicho que los ruteadores núcleo originales de Internet utilizaban el GGP (Gateway to Gateway Protocol) para comunicarse entre ellos y que, en la última versión, ya emplean SPREAD. El cambio fue realizado sin afectar rutas en otros sistemas autónomos. En el siguiente

capítulo, se analizan los protocolos que emplean ahora los sistemas autónomos para propagar información de ruteo.

Conceptualmente la idea de un sistema autónomo es consecuencia directa y natural de la generalización de la arquitectura descrita en la figura 15.2, con sistemas autónomos remplazando redes de área local. La figura 15.3 ilustra la idea.

Para lograr que las redes ocultas dentro de un sistema autónomo sean accesibles a través de Internet, cada sistema autónomo debe acordar la difusión de la información de la accesibilidad de la red hacia los otros sistemas autónomos. Aun cuando los anuncios puedan ser enviados hacia cualquier sistema autónomo, en una arquitectura de núcleo es crucial que cada sistema autónomo difunda información hacia un ruteador núcleo. Usualmente un ruteador en un sistema autónomo tiene la responsabilidad de anunciar rutas e interactuar de manera directa con uno de los ruteadores núcleo. Es posible, sin embargo, tener varios ruteadores y que cada uno anuncie un subconjunto de redes.

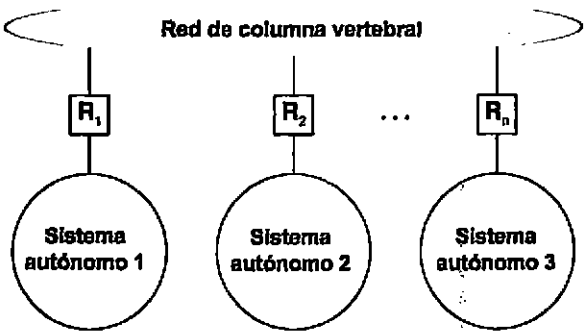


Figura 15.3 Arquitectura de una red de redes con sistemas autónomos en localidades de la columna vertebral de red. Cada sistema autónomo está formado por varias redes y ruteadores bajo una sola autoridad administrativa.

Podría parecer que nuestra definición de sistema autónomo es vaga, pero en la práctica las fronteras entre sistemas autónomos deben ser precisas para permitir que los algoritmos automatizados tomen decisiones de ruteo. Por ejemplo, un sistema autónomo, propiedad de alguna compañía, puede seleccionar no rutear paquetes a través de otro sistema autónomo propiedad de otra compañía, aun cuando estén directamente conectados. Para hacer posible que los algoritmos de ruteo automatizados distingan entre sistemas autónomos, a cada uno le asigna un *número de sistema autónomo* la misma autoridad central que está a cargo de asignar todas las direcciones de red en Internet. Cuando dos ruteadores intercambian información de accesibilidad de red, el mensaje transporta el identificador de sistema autónomo que el ruteador representa.

Podemos resumir las siguientes ideas:

Una red de redes extensa del TCP/IP tiene una estructura adicional para adaptarse a las fronteras administrativas: cada colección de redes y ruteadores controlados por una autoridad administrativa se considera como un solo sistema autónomo.

Un sistema autónomo tiene libertad para seleccionar una arquitectura de ruteo interna, pero debe reunir información sobre todas sus redes y designar uno o más ruteadores que habrán de transferir información de accesibilidad hacia otros sistemas autónomos. Debido a que la conexión de Internet se vale de una arquitectura de núcleo, todos los sistemas autónomos deben transferir información de accesibilidad hacia los ruteadores núcleo de Internet.

La siguiente sección presenta los detalles del protocolo que utilizan los ruteadores para anunciar la accesibilidad de red. En secciones posteriores regresaremos a cuestiones relacionadas con la arquitectura para analizar una restricción importante que el protocolo impone al ruteo. En estas secciones también se mostrará cómo puede extenderse el modelo de Internet.

15.5 Protocolo de pasarela exterior (EGP)

A dos ruteadores que intercambian información de ruteo se les llama *vecinos exteriores*, si pertenecen a dos sistemas autónomos diferentes, y *vecinos interiores* si pertenecen al mismo sistema autónomo. El protocolo que emplea vecinos exteriores para difundir la información de accesibilidad a otros sistemas autónomos se le conoce como *Protocolo de Pasarela Exterior* (*Exterior Gateway Protocol*)[1] o *EGP*, y los ruteadores que se utilizan se conocen como *ruteadores exteriores*. En la conexión de Internet, el EGP es especialmente importante ya que los sistemas autónomos lo emplean para difundir información de accesibilidad hacia el sistema de núcleo.

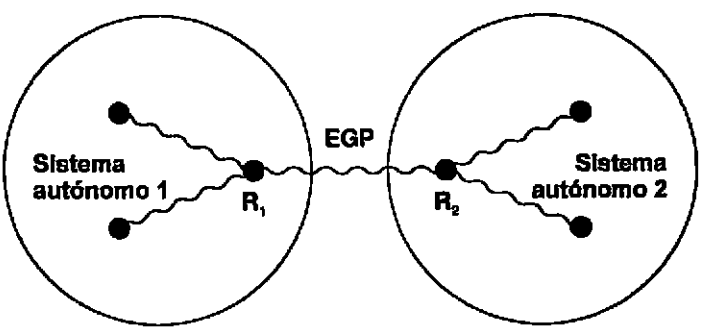


Figura 15.4 Ilustración conceptual de dos ruteadores exteriores, R_1 y R_2, que utilizan el EGP para anunciar redes en sus sistemas autónomos luego de reunir la información. Como su nombre lo indica, los ruteadores *exteriores* están por lo general cerca de la orilla exterior de un sistema autónomo.

[1] Recordemos que los científicos originalmente utilizaban el término *compuerta IP*, en lugar de *ruteador*.

La figura 15.4 muestra dos vecinos exteriores que utilizan el EGP. El ruteador R_1 recoge información acerca de las redes en el sistema autónomo *1* y reporta esta información al ruteador R_2 mediante el EGP, mientras el ruteador R_2 reporta información desde el sistema autónomo *2*.

El EGP tiene tres características principales. Primero, soporta un mecanismo de *adquisición de vecino* que permite a un ruteador solicitar a otro un acuerdo para que los dos comuniquen información de accesibilidad. Decimos que un ruteador *consigue* un *par EGP* (*EGP peer*) o un *vecino EGP*. Los pares EGP son vecinos sólo en el sentido en que estos intercambiarán información de ruteo, con lo cual no se hace alusión a su proximidad geográfica. Segundo, un ruteador prueba continuamente si su vecino EGP está respondiendo. Tercero, los vecinos EGP intercambian información de accesibilidad de red de manera periódica, transfiriendo un *mensaje de actualización de ruteo.*

15.6 Encabezado de mensajes (EGP)

Para implementar las tres funciones básicas, el EGP define nueve tipos de mensajes como se muestra en la siguiente tabla:

Tipo de mensaje EGP	Descripción
Acquisition Request	Solicitud para que un ruteador se defina como vecino (par)
Acquisition Confirm	Respuesta positiva a la solicitud de adquisición
Acquisition Refuse	Respuesta negativa a la solicitud de adquisición
Cease Request	Solicitud para terminar la relación con un vecino
Cease Confirm	Respuesta de confirmación para suspender la solicitud
Hello	Solicitud a un vecino para que responda si está activo
I Heard You	Respuesta al mensaje hello
Poll Request	Solicitud de actualización de ruteo de red
Routing Update	Información de accesibilidad de red
Error	Respuesta a un mensaje incorrecto

Todos los mensajes EGP comienzan con un encabezado fijo que identifica el tipo de mensaje. La figura 15.5 muestra el formato de un encabezado EGP.

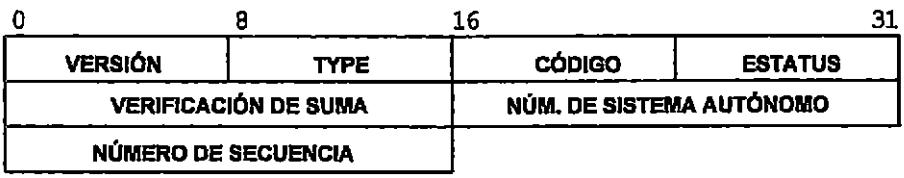


Figura 15.5 Encabezado fijo que precede a todos los mensajes EGP.

El campo de encabezado *VERSION* contiene un entero que identifica la versión de EGP utilizada para el formato del mensaje. Los receptores verifican el número de versión para comprobar que el software que se emplea tiene la misma versión que el protocolo. El campo *TYPE* identifica el tipo

de mensaje, junto con el campo *CODE* (*CÓDIGO*) que se utiliza para distinguir subtipos. El campo *STATUS* (*ESTATUS*) contiene mensajes que dependen de la información de estado.

El EGP se vale de una suma de verificación para comprobar que el mensaje llegó intacto. Utiliza el mismo algoritmo de verificación IP, tratando al mensaje EGP completo como una secuencia de enteros de 16 bits y tomando el complemento a uno del complemento a uno de la suma. Cuando realiza el cálculo, se asume que el campo *CHECKSUM* (*VERIFICACIÓN DE SUMA*) contiene ceros y el mensaje se llena con un múltiplo de 16 bits, añadiendo ceros.

El campo *AUTONOMOUS SYSTEM NUM* (*NÚM. DE SISTEMA AUTÓNOMO*) tiene el número asignado del sistema autónomo del ruteador que envía el mensaje y el campo *SEQUENCE NUMBER* (*NÚMERO DE SECUENCIA*) contiene un número que el emisor utiliza para asociar réplicas con el mensaje. Un ruteador establece una secuencia de valores inicial cuando consigue un vecino e incrementa el número de la secuencia cada vez que envía un mensaje. El vecino responde con el último número de secuencia que ha recibido, permitiendo al emisor establecer una correspondencia de las respuestas hacia las transmisiones.

15.7 Mensajes de adquisición de vecino EGP

Un ruteador envía un mensaje de *adquisición de vecino* para establecer comunicación EGP con otro ruteador. Note que EGP no especifica por qué o cómo selecciona un ruteador a otro ruteador como su vecino. Asumiremos que esta selección la realiza la organización responsable de la administración de ruteadores y no el software de protocolo.

Además del encabezado estándar con un número de secuencia, el mensaje de adquisición de vecino contiene valores iniciales para un intervalo de tiempo que se emplea para probar si el vecino está activo (llamado *intervalo de saludo o hello interval*), y un *intervalo de sondeo* (*polling interval*) que controla la frecuencia máxima de actualizaciones de ruteo. El emisor proporciona un intervalo de sondeo *n* para especificar que el receptor no debe obtener sondeos más que dentro de un intervalo de *n* segundos.[2] El emisor original puede cambiar el intervalo de sondeo de manera dinámica conforme transcurra el tiempo. Además, el intervalo de sondeo que utilizan los pares puede ser asimétrico, lo que permite que el sondeo de un elemento del par sea más frecuente que el otro. La figura 15.6 muestra el formato del mensaje de adquisición y las respuestas.

0	8	16	24 31
VERSIÓN	TYPE (3)	CÓDIGO (0 a 4)	ESTATUS
VERIFICACIÓN DE SUMA		NÚM. DE SISTEMA AUTÓNOMO	
NÚMERO DE SECUENCIA		INTERVALO DE SALUDO	
INTERVALO DE SONDEO			

Figura 15.6 Formato del mensaje de adquisición de vecino EGP. Los campos que se encuentran después del encabezado especifican los parámetros iniciales utilizados por el protocolo.

[2] En la práctica, la mayor parte de las implantaciones emplean el intervalo de sondeo como la frecuencia exacta a la que envían las solicitudes de sondeo.

El campo *CODE* identifica el mensaje específico como se muestra en la siguiente tabla:

Código	Significado
0	Solicitud de adquisición
1	Confirmación de adquisición
2	Rechazo de adquisición
3	Solicitud de interrupción
4	Confirmación de interrupción

15.8 Mensajes de accesibilidad de vecino EGP

El EGP permite dos formas de prueba para verificar si un vecino está activo. En el modo activo, el ruteador prueba al vecino periódicamente enviando un mensaje *Hello* (*saludo*) junto con un mensaje *poll* (*sondeo*) y espera una respuesta. En el modo pasivo, un ruteador depende de su vecino para enviar en forma periódica mensajes *Hello* o *poll*. Un ruteador opera en modo pasivo utilizando información del *campo de estado* de un mensaje de accesibilidad (ver más abajo) para deducir si el par está activo y si el par tiene conocimiento de esta actividad. Por lo general ambos ruteadores operan en el modo activo.

Separar el cálculo de accesibilidad de vecino del intercambio de accesibilidad de ruteo es importante porque permite disminuir la sobrecarga de la red. Dado que la información de ruteo de la red no cambia de manera tan frecuente como el estado de las máquinas de ruteo individualmente, no es necesario transferirla tan a menudo. Además, los mensajes de accesibilidad de vecino son pequeños y requieren de pocos cálculos adicionales, en tanto que los mensajes de intercambio de ruteo son largos y necesitan muchos cálculos. Así, separando las dos pruebas, los vecinos pueden realizar pruebas frecuentemente con un mínimo de cálculos y comunicación adicional. La figura 15.7 muestra la solicitud de accesibilidad de vecino que consiste sólo de un encabezado de mensaje EGP.

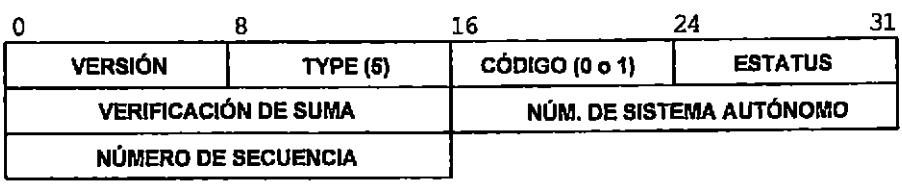


Figura 15.7 Formato del mensaje de sondeo de EGP. El valor *0* en CODE especifica una solicitud *Hello*, en tanto que el valor *1* especifica una respuesta *I Heard You*.

Dado que es posible que el mensaje *Hello* o la respuesta *I Heard You* se pierdan en el trayecto, el EGP utiliza una forma de la *regla "de n, k no llegan"* (*k-out-of-n*) para determinar si un par o vecino (*peer*) ha cambiado de un estado "up" a un estado "down". La mejor forma de pensar este algoritmo es imaginando que un ruteador envía una secuencia continua de mensajes *Hello* y recibe respuestas *I*

Heard You, y pensar en una ventana formada con los últimos *n* intercambios. Cuando menos *k* de los últimos *n* intercambios deben fallar para que el ruteador declare al vecino en un estado "down", y por lo menos *j* deben aparecer para que el ruteador declare al vecino en un estado "up", una vez que ha sido declarado "down". El estándar del protocolo sugiere valores para *j* y *k* que implican que dos mensajes sucesivos deben perderse (o recibirse) antes de que el EGP declare al par en estado "down" (o "up").

La histéresis introducida por *j* y *k* tiene un efecto importante en todo el desempeño del EGP. Como en cualquier algoritmo de ruteo, el EGP no difunde cambios innecesarios. La razón es obvia: los cambios no se detienen luego de que un ruteador los difunde a su par EGP. El par puede propagarlos hacia otros ruteadores. Minimizar los cambios de ruta rápidos es especialmente importante cuando un par EGP emplea un algoritmo de vector-distancia para difundir cambios, dado que los cambios continuos pueden hacer que los algoritmos vector-distancia sean inestables. Así, si los ruteadores exteriores reportaran cambios de accesibilidad cada vez que un mensaje se pierde, estos podrían ocasionar que el sistema de ruteo se mantuviera en transición continua.

15.9 Mensajes de solicitud de sondeo EGP

Los mensajes EGP de *solicitud de sondeo* y *respuesta de sondeo* permiten a un ruteador obtener información sobre la accesibilidad de un ruteador. La figura 15.8 muestra el formato del mensaje. El campo *IP SOURCE NETWORK* (*RED FUENTE IP*) especifica una red común para los sistemas autónomos que estén conectados al ruteador. La respuesta contendrá rutas que tienen distancias medidas con respecto a los ruteadores en la red fuente IP especificada.

0	8	16	24 31
VERSIÓN	TYPE (2)	CÓDIGO (0 o 1)	ESTATUS
VERIFICACIÓN DE SUMA		NÚM. DE SISTEMA AUTÓNOMO	
NÚMERO DE SECUENCIA		RESERVADO	
RED FUENTE IP			

Figura 15.8 Formato del mensaje de sondeo de EGP. El valor *0* en CODE especifica una solicitud *Hello*, en tanto que el valor *1* específica una respuesta *I Heard You*.

Puede ser difícil entender por qué en el formato EGP se selecciona hacer una solicitud de sondeo y se específica una red fuente. Hay dos razones. Primero, recordemos que un ruteador conecta dos o más redes físicas. Si una aplicación en el ruteador implanta EGP, podría no saber a qué interfaz llegó la solicitud EGP. Así, podría no saber hacia qué red remitir la solicitud. Segundo, los ruteadores que corren EGP con frecuencia reúnen información para un sistema autónomo entero. Cuando anuncian la accesibilidad de red, el ruteador exterior envía a los vecinos un conjunto de pares, los cuales especifican una red de destino en el sistema autónomo y el ruteador utilizado para alcanzar tal destino.

Por supuesto el ruteador empleado para alcanzar el destino depende de en dónde se introduzca el tráfico en el sistema autónomo. La red fuente mencionada en la solicitud de sondeo especifica el punto en el que los paquetes se introducirán al sistema autónomo. La figura 15.9 ilustra la idea de una red común utilizada como una base para la información de la accesibilidad de red.

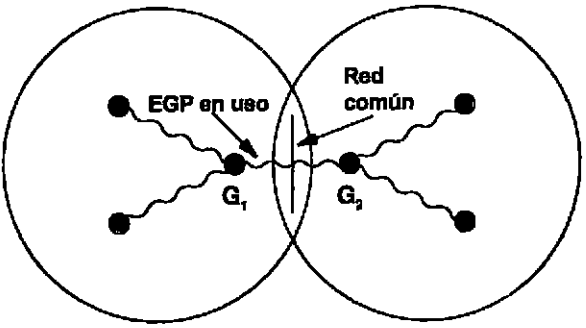


Figura 15.9 Ruteadores en dos sistemas autónomos que utilizan el EGP para comunicar información de accesibilidad de red. Por medio de estos ruteadores un mensaje de accesibilidad especifica rutas en una red común a ambos sistemas y destinos accesibles.

15.10 Mensaje de actualización de enrutamiento EGP

Un ruteador exterior envía un mensaje de *actualización de ruteo* a su vecino EGP para transportar información acerca de las redes accesibles. Por lo general el ruteador ha reunido la información y la pone a disposición de un ruteador en otro sistema autónomo. En principio, un ruteador que corre el EGP puede reportar dos tipos de accesibilidad hacia un par. El primer tipo consiste en redes de destino que son accesibles completamente dentro del sistema autónomo del ruteador. El segundo tipo consiste en redes de destino de las que el ruteador tiene información, pero que se localizan más allá de las fronteras del sistema autónomo del ruteador.

Es importante entender que el EGP no permite anunciar a cualquier ruteador la accesibilidad hacia cualquier red de destino. La restricción limita a un ruteador para que anuncie únicamente destinos autorizados, esto es:

> *El EGP restringe a los ruteadores (que no son núcleo) para que anuncien sólo a aquellas redes completamente accesibles desde dentro de su sistema autónomo.*

Mediante esta regla, a veces llamada *restricción de terceros de EGP* se intenta controlar la difusión de información y permite que cada sistema autónomo seleccione exactamente cómo anunciar su accesibilidad. Por ejemplo, si en cada campus universitario se forma un sistema autónomo, un ruteador en el campus de una universidad dada puede reunir información acerca de las redes en el campus y anunciarla a cualquiera que tenga una conexión a Internet en el campus, pero un campus no anuncia rutas para redes en otros campus. Naturalmente la restricción de terceros no se aplica a los sistemas de núcleo.

La figura 15.10 ilustra el formato de un mensaje de actualización de ruteo. El campo llamado *#INT.GWYS* y *#EXT.GWYS* tiene el número de ruteador interior y exterior unidos en el mensaje. La distinción entre ruteadores interiores y exteriores permite al ruteador saber si las distancias son comparables. Por desgracia, es imposible establecer una distinción para las direcciones de ruteadores únicos y el formato de mensaje no contiene mayor información. En la práctica las implantaciones del EGP resuelven el problema al enviar mensajes de actualización separados para ruteadores interiores y exteriores. El campo *IP SOURCE NETWORK* indica la red desde la que se mide toda la accesibilidad.

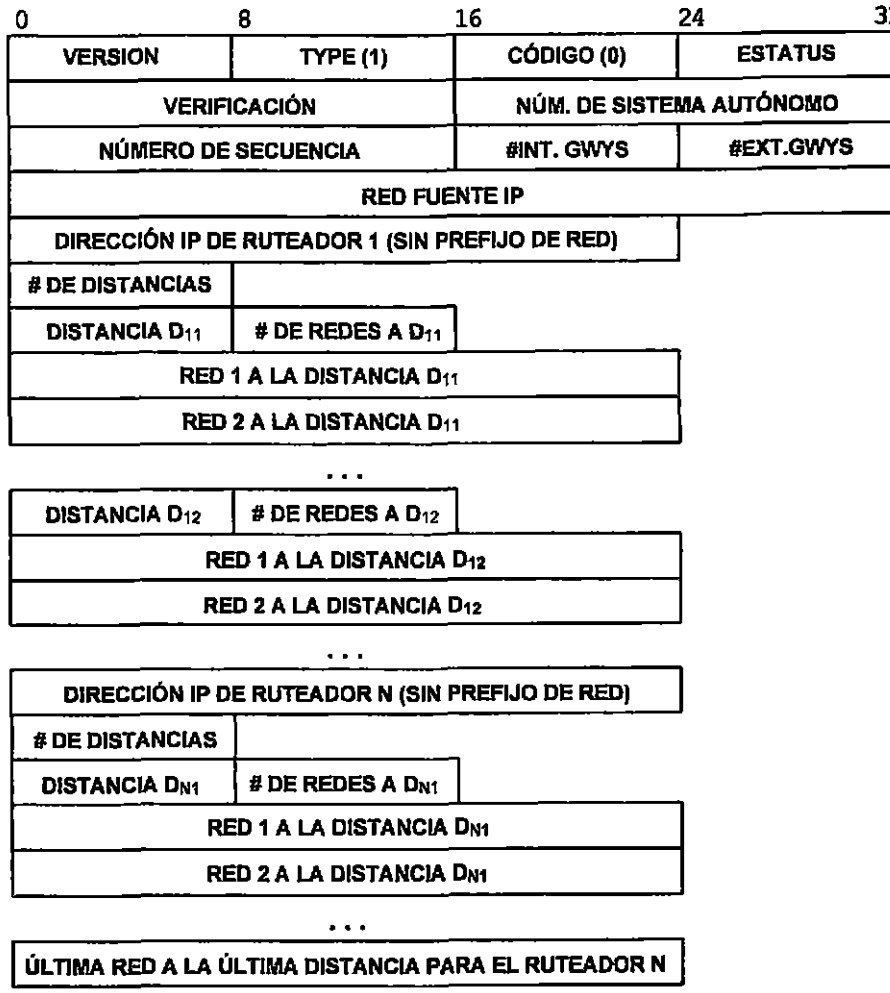


Figura 15.10 Formato del mensaje de actualización de ruteo EGP. Todas las rutas son relativas a una red específica. El mensaje lista los ruteadores en la red y la distancia de los destinos a través de éstos. Una dirección de red contiene *1*, *2* o *3* octetos.

En cierto sentido, los mensajes de actualización de ruteo EGP son una generalización de los mensajes de actualización de ruteo GGP dado que éstos se adaptan a varios ruteadores en lugar de a uno solamente. Así, el campo de mensaje de actualización de ruteo que sigue después de *IP SOURCE NETWORK* forma una secuencia de bloques, donde cada bloque tiene información de accesibilidad para uno de los ruteadores de la red fuente. Un bloque se forma con la dirección IP de un ruteador. La red accesible desde este ruteador está listada junto con su distancia. Como en el GGP, en el EGP las redes se agrupan en conjuntos en función de su "distancia". Para cada distancia, hay un contador de red, seguido de una lista de direcciones de red. Luego de listar todas las redes en una distancia dada, el patrón se repite para todos los valores de distancia.

15.11 Medición desde la perspectiva del receptor

A diferencia de la mayor parte de los protocolos que difunden información de ruteo, el EGP no indica sus propios costos de acceso a las redes de destino. De hecho, las distancias se miden desde una red de fuente común, así todas las distancias son correctas desde la perspectiva del par. La figura 15.11 ilustra esta idea.

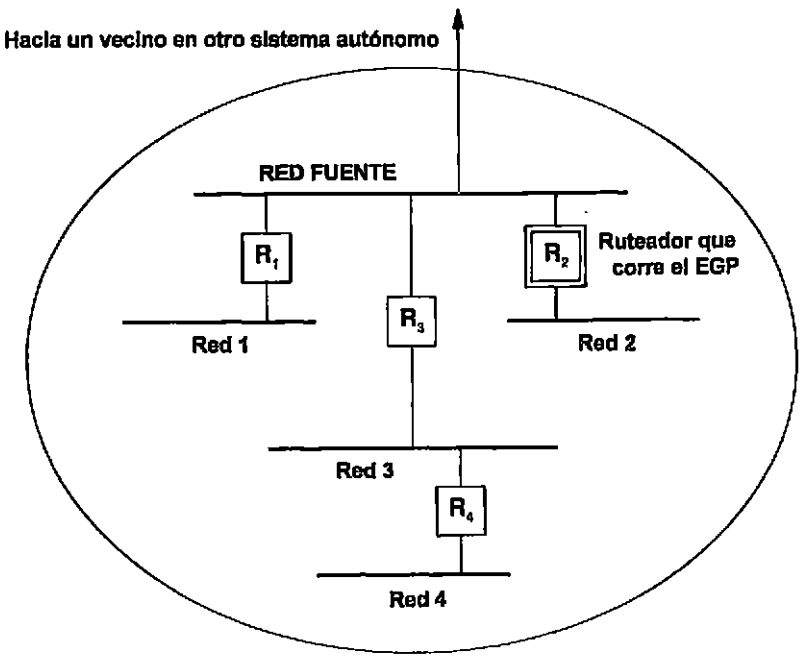


Figura 15.11 Ejemplo de un sistema autónomo. El ruteador R_2 corre el EGP y reporta distancias hacia todas las redes, medidas desde la red fuente, no desde su propia tabla de ruteo.

En el ejemplo de la figura 15.11 el ruteador R_2 ha sido designado para correr el EGP en nombre del sistema autónomo. Debe reportar la accesibilidad de las redes *1* a *4*. Debe reportar la red *1* como accesible a través del ruteador R_1, las redes *3* y *4* como accesibles desde el ruteador R_3 y la red *2* como accesible desde R_2. Desde la perspectiva de R_2, la red *2* está ubicada a una distancia *0*. Sin embargo, se reporta a la red *2* a la distancia *1*, que es la distancia que existe si se mide desde la red fuente.

15.12 La restricción clave de EGP

Hemos visto que EGP restringe a los ruteadores permitiéndoles anunciar sólo los destinos de red completamente accesibles dentro del sistema autónomo del ruteador. Sin embargo, hay una limitación más importante impuesta por el EGP:

> *El EGP no interpreta ninguna de las métricas de distancia que aparecen en los mensajes de actualización de ruteo.*

La regla especifica que un valor de 255 significa que la red es inaccesible, pero otros valores son también comparables si estos se refieren a ruteadores en el mismo sistema autónomo. En esencia el EGP utiliza el campo de distancia para especificar si una trayectoria existe; el valor no puede usarse para calcular la proximidad de dos rutas a menos que ambas se encuentren dentro de un solo sistema autónomo.

Podemos ver ahora por qué un ruteador en un sistema autónomo no debe anunciar la accesibilidad a redes en otros sistemas autónomos (esto es, porque existe la exclusión de terceros). La observación esencial es ésta: cuando un ruteador aprende algo sobre una red en otro sistema autónomo no obtiene una medición universalmente aceptada de distancias. Por lo tanto, no debería transferir la medida. Anunciar la accesibilidad con el EGP es equivalente a decir: "mi sistema autónomo proporciona *la* trayectoria para esta red". No hay forma de que el ruteador diga: "mi sistema autónomo proporciona una posible ruta hacia esta red".

Considerar la interpretación de distancia nos permite concluir que el EGP no puede utilizarse como un algoritmo de ruteo. En particular, incluso si un ruteador sabe algo de dos diferentes rutas de la misma red, no puede saber cuál es más corta. Sin información de ruteo, debemos ser cuidadosos para anunciar sólo la ruta que nosotros queremos seguir en el tráfico. Como resultado, hay sólo una trayectoria desde un núcleo hacia cualquier red. Así pues, podemos resumir lo siguiente:

> *Debido a que el EGP sólo difunde información de accesibilidad, restringe la topología de cualquier red de redes que utilice el EGP, a una estructura de árbol en la que un sistema de núcleo forma la raíz; no hay ciclos entre otros sistemas autónomos conectados.*

El punto clave aquí es que cualquier red de redes que utilice el EGP para proporcionar ruteo entre sistemas autónomos forma una topología en forma de árbol, en la cual un sistema autónomo de núcleo constituye la raíz.

La restricción en el EGP que produce una estructura de árbol resulta en parte de la evolución histórica de Internet centrada en una sola columna vertebral de red. Aun cuando podría parecer algo inocuo, la restricción tiene algunas consecuencias sorprendentes.

1. La conectividad universal falla si el sistema de ruteo de núcleo falla. Por supuesto es improbable que todo el núcleo de Internet falle simultáneamente, pero hay ejemplos interesantes de fallas menores. En particular, en varias ocasiones el rápido crecimiento de Internet provocó sobreflujos en las tablas de los ruteadores núcleo. Para evitar esto, el EGP se ha instalado con éxito en las rutas para las redes nuevas. Estas direcciones de red que no pueden instalarse en las tablas de núcleo son inaccesibles desde muchas partes de Internet.
2. El EGP sólo puede anunciar una trayectoria hacia una red dada. Esto es, en cualquier instante dado, todo el tráfico ruteado desde un sistema autónomo hacia una red, en otro sistema autónomo, recorrerá una trayectoria, incluso si están presentes varias conexiones físicas. Además, note que un sistema autónomo exterior sólo utilizará una trayectoria de retorno aun cuando el sistema fuente divida el tráfico que sale entre dos o más rutas. Como resultado, los retardos y el desempeño entre un par de máquinas pueden ser asimétricos, lo que hace que una red de redes sea difícil de monitorear o depurar.
3. El EGP no soporta compartir cargas de ruteadores entre sistemas autónomos arbitrarios. Si dos sistemas autónomos tiene varios ruteadores conectados, uno de ellos parecerá balancear el tráfico entre todos los ruteadores. El EGP permite a los sistemas autónomos dividir la carga por redes (por ejemplo, para dividirse ellos mismos en varios subconjuntos y tener varios ruteadores que anuncien particiones), pero no soporta la mayor parte de los procedimientos generales para compartir cargas.
4. Como caso especial del punto 3, el EGP es inadecuado para optimizar rutas en una arquitectura que tiene varias columnas vertebrales de redes interconectadas en varios puntos. Por ejemplo, la interconexión entre NSFNET y ARPANET, descrita en el capítulo 14, no puede utilizar el EGP sólo para intercambiar información de ruteo si las rutas deben ser óptimas. De hecho, los administradores dividen manualmente el conjunto de redes NSFNET y anuncian algunas de estas hacia el ruteador exterior y otros hacia diferentes ruteadores.
5. Es difícil conmutar hacia una trayectoria física alterna si una de las rutas falla, especialmente cuando las rutas atraviesan dos o más sistemas autónomos. Dado que EGP no interpreta distancias, terceras partes no pueden anunciar rutas y dependen del receptor para conmutar hacia rutas alternas si una falla. En lugar de ello, la responsabilidad para seleccionar rutas de costo mínimo recae sobre los ruteadores exteriores que anuncian accesibilidad.

15.13 Problemas técnicos

El EGP tiene varios puntos débiles, muchos de los cuales son tecnicismos triviales. Estas debilidades deben repararse antes de que el EGP tenga que soportar la rápida expansión del ambiente de Internet. Algunos intentos para resolver estos problemas se han concentrado en lo más urgente: reducir el tamaño de los mensajes de actualización. Recordemos de la figura 15.10 que los mensajes de actualización contienen largas listas de redes. Para sistemas autónomos extensos con muchos rutea-

dores y redes, el tamaño de un mensaje de actualización de ruteo EGP por sí solo puede exceder la mayor parte de las MTU de las redes. Algo muy importante, si un fragmento de un datagrama IP se pierde, el datagrama entero debe retransmitirse.

Aun cuando se han identificado muchos problemas técnicos, varios intentos para producir una nueva versión de EGP han fallado. Los esfuerzos conocidos como *EGP2* y *EGP3* fueron abandonados luego de que los participantes se reconocieron incapaces de acordar un método de solución y sus detalles. En tanto se exploraban algunas posibilidades, los grupos de trabajo estudiaron el remplazo de EGP y decidieron que, puesto que existía la necesidad de llevar a cabo muchos cambios fundamentales, sería inadecuado realizar simples mejoras. En consecuencia, el EGP continúa en uso y sin cambios.

15.14 Descentralización de la arquitectura Internet

Dos cuestiones importantes respecto a la arquitectura se mantienen sin respuesta. La primera se enfoca en la centralización: ¿cómo puede una arquitectura de red de redes modificarse para suprimir la dependencia en un sistema de ruteo de núcleo (centralizado)? La segunda se relaciona con los niveles de confianza: ¿una arquitectura de red de redes se puede expandir a fin de permitir que la cooperación cercana (confianza) se dé más entre algunos sistemas autónomos que entre otros?

Suprimir toda dependencia de un sistema de núcleos no es fácil. Aunque la arquitectura TCP/IP continúa evolucionando, las raíces de la centralización son evidentes en muchos protocolos. Así, como la mayor parte del software se construye a partir de protocolos existentes, la inercia se incrementa y los cambios se hacen más difíciles y caros. Algo también importante es que, debido a que los sistemas de núcleo conectados a Internet son confiables, apoyados por un grupo de profesionales que utilizan mecanismos automatizados para actualizar información de ruteo, es muy pequeña la motivación para el cambio. Finalmente, conforme el tamaño de una red de redes crece, el volumen de la información de ruteo que deben manejar los ruteadores también aumenta. Se debe contar con un mecanismo para limitar la información necesaria en cada nodo pues, de otra manera, el tráfico de actualizaciones inundará la red.

15.15 Más allá de los sistemas autónomos

Ampliar la noción de confianza entre sistemas autónomos es complejo. La forma más fácil es agrupar sistemas autónomos de manera jerárquica. Imagine, por ejemplo, tres sistemas autónomos en tres departamentos administrativos separados en un gran campus universitario. Es natural agrupar a estos tres juntos, ya que comparten vínculos administrativos. La motivación para agrupar jerárquicamente proviene, en primer término, de la noción de confianza. Los ruteadores dentro de un grupo confían uno en otro por su alto nivel de confidencialidad.

Agrupar sistemas autónomos requiere sólo de cambios menores para el EGP. Las nuevas versiones del EGP deben acordar el uso de un factor de escala artificial cuando reportan conteos de saltos, permitiendo que los contadores se incrementen cuando pasen a través de fronteras de un grupo a otro. La técnica, conocida con cierta ambigüedad como *transformación métrica*, divide los valores

de distancias en tres categorías. Por ejemplo, supongamos que los ruteadores dentro de un sistema autónomo emplean un valor de distancia inferior a 128. Podemos establecer la regla de que, cuando pase información de distancia a través de la frontera de un sistema autónomo dentro de un solo grupo, la distancia se deberá transformar dentro de un rango que abarcará desde 128 hasta 191. Por último, podremos establecer la regla de que, cuando pasen valores de distancia a través de las fronteras entre dos grupos, el valor tendrá que transformarse dentro del rango de 192 a 254.[3] El efecto de esta transformación es obvio: para cualquier red de destino dada, ninguna trayectoria que se encuentre completamente dentro del sistema autónomo tiene la garantía de tener un costo menor que la ruta de un sistema que se ubica en un sistema autónomo exterior. Además, entre todas las trayectorias que se desvían al exterior del sistema autónomo, las que se mantienen dentro del grupo tienen un costo menor que las que atraviesan las fronteras del grupo. La ventaja clave de la transformación métrica es que se vale de un protocolo ya existente: el EGP. Las transformaciones permiten a un administrador de sistema autónomo tener la libertad de seleccionar métricas de distancias internamente, lo cual hace posible para otros sistemas la comparación de costos de ruteo.

15.16 Resumen

Internet está compuesta por un conjunto de sistemas autónomos, donde cada sistema autónomo consiste en ruteadores y redes bajo una autoridad administrativa. Un sistema autónomo utiliza el Exterior Gateway Protocol (EGP) para anunciar rutas hacia otros sistemas autónomos. Específicamente, un sistema autónomo debe anunciar la accesibilidad de sus redes hacia otros sistemas antes de que sus redes sean accesibles para fuentes con este sistema. Dijimos que el EGP soporta tres funciones básicas: la adquisición de vecino (par), la prueba de accesibilidad de vecino y el anuncio de accesibilidad hacia los vecinos.

La arquitectura de la red global de Internet consiste en una parte central (construida alrededor de la columna vertebral NSFNET), con sistemas autónomos conectados al centro en una topología de estructura de árbol. El sistema de ruteo NSFNET forma el núcleo central, en tanto que la "periferia" consiste en redes de área local que tienen sólo una conexión hacia el resto de Internet. El paso de la estructura centralizada a la distribución completa requiere de cambios sustanciales en protocolos como el EGP.

PARA CONOCER MÁS

Mills (RFC 904) contiene las especificaciones formales del protocolo EGP. Una de las primeras versiones del EGP se encuentra en Rosen (RFC 827), que también analiza la restricción a una topología de estructura de árbol. Se puede encontrar información adicional en los documentos sobre los primeros ruteadores en Seamonson y Rosen (RFC 888) y Mills (RFC 975). Braden y Postel (RFC

[3] El término *confederación autónoma* se ha utilizado para describir un grupo de sistemas autónomos; las fronteras de las confederaciones autónomas corresponden a las transformaciones que se encuentran más allá del valor 191.

1009) tratan los requerimientos para los ruteadores de Internet y resaltan algunos de los problemas con el EGP (ver también el predecesor, RFC 985). Lougheed y Rekhter (RFC 1105) describen el Border Gateway Protocol, *BGP*, un protocolo semejante al EGP utilizado dentro de NSFNET. El BGP ha pasado a través de tres revisiones significativas; la última versión aparece en (RFC 1163, 1267 y 1654). Por último, Kirton (RFC 911) describe la implantación ampliamente utilizada del EGP, el BSD, que corre bajo UNIX Berkeley 4.3.

EJERCICIOS

15.1 Si su localidad está conectada a NSFNET, averigüe si los ruteadores corren el EGP. ¿Cuántos ruteadores difunde NSFNET?

15.2 Las implantaciones del EGP utilizan un mecanismo "hold down" que hace que el protocolo para retardo acepte una *solicitud de adquisición* desde un vecino para un tiempo fijo, seguido de la recepción de un mensaje de *solicitud de interrupción* desde el vecino. Lea las especificaciones del protocolo para entender por qué.

15.3 Para las redes de la figura 15.2, ¿qué ruteadores deben correr el EGP? ¿Por qué?

15.4 La especificación formal del EGP incluye una máquina de estado finito que explica cómo opera el EGP. ¿Por qué el mensaje *confirm* toma la máquina de estado finito EGP desde el estado *acquisition* hasta el estado *down*, en lugar de tomar desde el estado *acquisition* hasta el estado *up*?

15.5 ¿Qué sucede si un ruteador en un sistema autónomo envía un mensaje de actualización de ruteo EGP a un ruteador en otro sistema autónomo, afirmando que tiene accesibilidad para todos los destinos posibles de la red de redes?

15.6 ¿Pueden establecer dos sistemas autónomos un ciclo de ruteo enviando mensajes de actualización EGP hacia otro? Explíquelo.

15.7 ¿Los ruteadores deben tratar al EGP por separado con respecto a sus tablas de ruteo propias? Por ejemplo, ¿un ruteador debe anunciar siempre accesibilidad si no cuenta con una ruta hacia esa red en su tabla de ruteo? ¿Por qué sí o por qué no?

15.8 Lea RFC 1654 y compare el BGP4 con el EGP. ¿Qué características adicionales soporta el BGP4?

15.9 Si usted trabaja en una compañía grande, averigüe si ésta tiene más de un sistema autónomo. De ser así, ¿cómo intercambian información de ruteo?

15.10 ¿Cuál es la mayor ventaja de dividir una corporación multinacional extensa en varios sistemas autónomos? ¿Cuál la mayor desventaja?

15.11 Las corporaciones *A* y *B* utilizan el EGP para intercambiar información de ruteo. Para mantener separadas las computadoras de *B* y limitar su acceso a las máquinas en una de sus redes, *N*, el administrador de red en la corporación *A*, configura el EGP para omitir a *N* en los anuncios enviados hacia *B*. ¿Es segura la red *N*? ¿Por qué sí o por qué no?

16

Ruteo en un sistema autónomo (RIP, OSPF, HELLO)

16.1 Introducción

En el capítulo anterior se introdujo el concepto de sistema autónomo y se examinó el protocolo Exterior Gateway Protocol (EGP) que utiliza un ruteador para anunciar redes dentro de su sistema a otros sistemas autónomos. En este capítulo, se completa la panorámica del ruteo de red de redes examinando cómo un ruteador en un sistema autónomo aprende sobre otras redes dentro de un sistema autónomo.

16.2 Rutas interiores dinámicas y estáticas

A dos ruteadores dentro de un sistema autónomo se les llama *interiores* con respecto a otro. Por ejemplo, dos ruteadores núcleo Internet son interiores en comparación con otro debido a que el núcleo forma un solo sistema autónomo. Dos ruteadores en un campus universitario son considerados interiores con respecto a otros mientras las máquinas en el campus estén reunidas en un solo sistema autónomo.

¿Cómo pueden los ruteadores en un sistema autónomo aprender acerca de redes dentro del sistema autónomo? En redes de redes pequeñas que cambian lentamente, los administradores pueden establecer y modificar rutas a mano. El administrador tiene una tabla de redes y actualiza la tabla si una red nueva se añade o se elimina del sistema autónomo. Por ejemplo, consideremos la red de redes de la pequeña corporación mostrada en la figura 16.1. El ruteo para cada red de redes es

insignificante porque sólo existe una ruta entre cualquiera de los dos puntos. El administrador puede configurar manualmente las rutas en todos los anfitriones y ruteadores. Si la red de redes cambia (por ejemplo, si se añade una nueva red), el administrador debe reconfigurar las rutas en todas las máquinas.

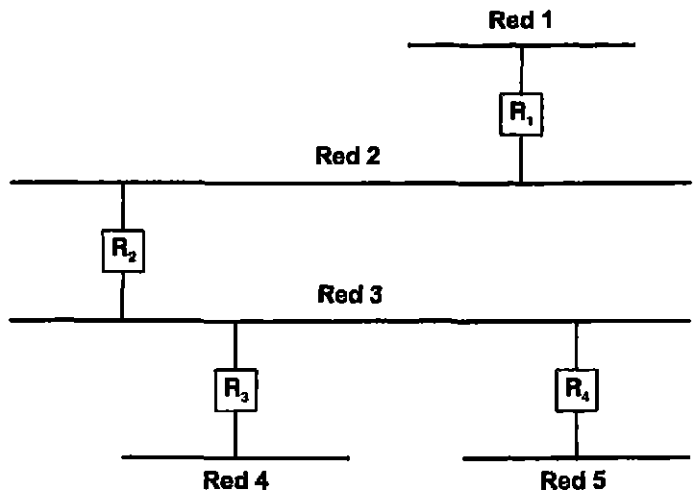


Figura 16.1 Ejemplo de una pequeña red de redes, formada por 5 redes Ethernet y 4 ruteadores en una sola localidad. Sólo puede existir un ruteador entre cualquiera de los dos anfitriones en esta red de redes.

La desventaja de un sistema manual es obvia; los sistemas manuales no se pueden adaptar al crecimiento o a los cambios rápidos. En un ambiente de cambios rápidos como el de Internet, la gente simplemente no puede responder a los cambios lo suficientemente rápido como para resolver los problemas; son necesarios pues métodos automatizados. Estos métodos pueden también ayudar a mejorar la confiabilidad y la respuesta a las fallas en pequeñas redes de redes que tienen rutas alternativas. Para ver cómo, consideremos lo que sucede si añadimos una ruta adicional a la red de redes en la figura 16.1, obteniendo la red de redes que se muestra en la figura 16.2.

En arquitecturas de red de redes que tienen varias rutas físicas, los administradores por lo regular seleccionan una de ellas como ruta primaria. Si el ruteador instalado a lo largo de la trayectoria primaria falla, las rutas se deben cambiar para enviar el tráfico hacia una ruta alternativa. Cambiar las rutas manualmente toma tiempo y es una labor propensa a los errores. Así, incluso en pequeñas redes de redes debe usarse un sistema automatizado para cambiar las rutas rápidamente y de manera confiable.

Para automatizar de manera segura el trabajo de información sobre la accesibilidad de una red dada, los ruteadores interiores normalmente se comunican con otros, intercambian información de accesibilidad de red o información de ruteo de red, a partir de la cual la accesibilidad se puede deducir. Una vez que la información de accesibilidad para un sistema autónomo completo se ha ensamblado, uno de los ruteadores en el sistema puede anunciarlo a otros sistemas autónomos utilizando el EGP.

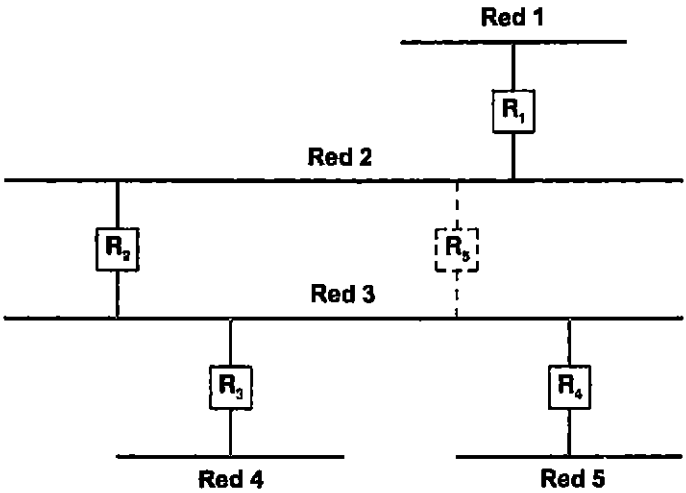


Figura 16.2 La adición del ruteador R_5 introduce una ruta alterna entre las redes *2* y *3*. El software de ruteo puede adaptarse rápidamente a una falla y conmutar rutas automáticamente hacia trayectorias alternas.

A diferencia de esto, la comunicación de un ruteador exterior, para el cual el EGP proporciona un estándar ampliamente aceptado, no se ha desarrollado un solo protocolo que se utilice con los sistemas autónomos. Una de las razones de esta diversidad proviene de la variedad de topologías y tecnologías que se utilizan en los sistemas autónomos. Otra de las razones se deriva del compromiso entre la simplicidad y la funcionalidad —los protocolos que son fáciles de instalar y configurar no proporcionan una funcionalidad sofisticada. Como resultado, sólo un puñado de protocolos se han vuelto populares; la mayoría de los sistemas autónomos utiliza uno de ellos exclusivamente para difundir información de ruteo internamente.

Dado que no se trata de un solo estándar, utilizaremos el término protocolo de pasarela interior o *IGP* (*interior gateway protocol*) como una descripción genérica para referirnos a cualquier algoritmo que utilicen ruteadores interiores cuando intercambian información sobre accesibilidad de red y ruteo. Por ejemplo, el ruteador de núcleo Butterfly forma un sistema autónomo especializado que utiliza SPREAT como su IGP. Algunos sistemas autónomos utilizan el EGP como su IGP, aun cuando esto casi siempre tiene sentido para sistemas autónomos pequeños y que abarcan redes de área local con capacidad de difusión.

La figura 16.3 ilustra un sistema autónomo que utiliza un IGP para difundir accesibilidad entre ruteadores interiores.

En la figura, IGP_1 se remite al protocolo de ruteador interior utilizado dentro del sistema autónomo *1*, e IGP_2 se remite al protocolo utilizado dentro del sistema autónomo 2. La figura también ilustra una idea importante:

> *Un solo ruteador puede utilizar 2 diferentes protocolos de ruteo simultáneamente, uno para la comunicación al exterior del sistema autónomo y otro para la comunicación al interior del sistema autónomo.*

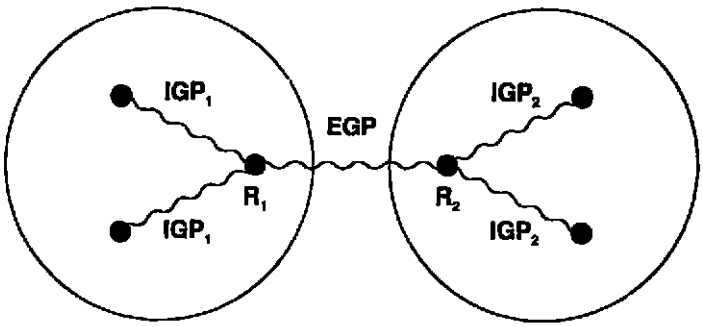


Figura 16.3 Representación del concepto de dos sistemas autónomos, cada uno utiliza su propio IGP internamente, pero se vale del EGP para realizar la comunicación entre un ruteador exterior y el otro sistema.

En particular, los ruteadores que corren el EGP para anunciar accesibilidad por lo general necesitan correr también un IGP para obtener información desde el interior del sistema autónomo.

16.3 Protocolo de información de ruteo (RIP)

Uno de los IGP más ampliamente utilizados es el *Protocolo de Información de Ruteo* (*RIP, Routing Information Protocol*), también conocido con el nombre de un programa que lo implementa, *routed*.[1] El software routed fue originalmente diseñado en la Universidad de Berkeley en California para proporcionar información consistente de ruteo y accesibilidad entre las máquinas de su red local. Este se apoya en la difusión de red física para realizar el intercambio de ruteo rápidamente. No fue diseñado para usarse en redes de área amplia (aunque ahora así se hace).

Con base en las primeras investigaciones de enlaces de redes realizadas en la corporación Xerox en el Centro de Investigación de Palo Alto (PARC), el *routed* implementa un protocolo derivado del *Protocolo de Información de Ruteo NS* de Xerox, pero se generalizó para cubrir varias familias de redes.

Al margen de mejoras menores con respecto a sus predecesores, la popularidad de RIP, como un IGP, no reside en sus méritos técnicos. Por el contrario, es el resultado de que Berkeley distribuyó el software *routed* junto con su popular sistema 4BSD de UNIX. Así, muchas localidades TCP/IP adoptaron e instalaron *routed* y comenzaron a utilizar RIP sin considerar sus méritos o limitaciones técnicas. Una vez instalado y corriendo, se convirtió en la base del ruteo local y varios grupos de investigadores lo adoptaron para redes amplias.

Posiblemente el hecho más sorprendente relacionado con el RIP es que fue construido y adoptado antes de que se escribiera un estándar formal. La mayor parte de las implantaciones se deriva del código Berkeley, teniendo entre sus limitaciones para el entendimiento del programador detalles no documentados y sutilezas relacionadas con la interoperabilidad. Conforme aparecen

[1] El nombre proviene de la convención de UNIX de insertar una "d" a los nombres de los procesos daemon; se pronuncia "route-d"

nuevas versiones, surgen más problemas. Un estándar RFC aparecido en junio de 1988 hizo posible que los vendedores aseguraran la interoperabilidad.

El protocolo subyacente RIP es consecuencia directa de la implantación del ruteo de vector-distancia para redes locales. En principio, divide las máquinas participantes en *activas* y *pasivas* (*silenciosas*). Los ruteadores activos anuncian sus rutas a los otros; las máquinas pasivas listan y actualizan sus rutas con base en estos anuncios, pero no anuncian. Sólo un ruteador puede correr RIP de modo activo; un anfitrión debe utilizar el modo pasivo.

Un ruteador que corre RIP de modo activo difunde un mensaje cada 30 segundos. El mensaje contiene información tomada de la base de datos de ruteo actualizada. Cada mensaje consiste de pares, donde cada par contiene una dirección de red IP y un entero que representa la distancia hacia esta red. RIP utiliza una *métrica de conteo de saltos* (*hop count metric*) para medir la distancia hacia un destino. En la métrica RIP, un ruteador define un salto[2] desde la red conectada directamente, dos saltos desde la red que está al alcance a través de otro ruteador, y así sucesivamente. De esta manera, el *número de saltos* (*number of hops*) o el *contador de saltos (hop count)* a lo largo de una trayectoria desde una fuente dada hacia un destino dado hace referencia al número de ruteadores que un datagrama encontrará a lo largo de la trayectoria. Debe ser obvio que utilizar el conteo de saltos para calcular la trayectoria más corta no siempre produce resultados óptimos. Por ejemplo, una trayectoria con un conteo de saltos igual a 3 que cruza tres redes Ethernet puede ser notablemente más rápido que una trayectoria con un contador de saltos igual a 2 que atraviesa dos líneas seriales lentas. Para compensar las diferencias tecnológicas, muchas implantaciones RIP permiten que los administradores configuren artificialmente los contadores de saltos con valores altos cuando deban anunciar conexiones hacia redes lentas.

Tanto los participantes RIP activos como los pasivos "escuchan" todos los mensajes difundidos y actualizan sus tablas de acuerdo al algoritmo vector-distancia descrito anteriormente. Por ejemplo, en la red de redes de la figura 16.2, el ruteador R_1 difundirá un mensaje en la red *2* que contiene el par (*1,1*), dando a entender que puede alcanzar la red *1* al costo *1*. Los ruteadores R_2 y R_5 recibirán la difusión e instalarán una ruta hacia la red *1* a través de R_1 (al costo *2*). Después, los ruteadores R_2 y R_5 incluirán el par (*1*,2) cuando difundan sus mensajes RIP en la red *3*. Finalmente, todos los ruteadores y anfitriones instalarán una ruta hacia la red *1*.

RIP especifica unas cuantas reglas para mejorar el desempeño y la confiabilidad. Por ejemplo, una vez que un ruteador aprende una ruta desde otro ruteador, debe conservar esta ruta hasta que aprenda otra mejor. En nuestro ejemplo, si los ruteadores R_2 y R_3 anuncian la red *1* al costo *2*, los ruteadores R_3 y R_4 instalarán una ruta a través del que logre anunciarlo primero. Así pues, podemos resumir lo siguiente:

> *Para prevenir que los ruteadores oscilen entre dos o más trayectorias de costos iguales, RIP especifica que se deben conservar las rutas existentes hasta que aparezca una ruta nueva con un costo estrictamente menor.*

¿Qué sucede si falla el primer ruteador que anuncia en la ruta (es decir, si queda fuera de funcionamiento)? RIP especifica que todos los escuchas deben asociar un tiempo límite a las rutas que aprenden por medio de RIP. Cuando un ruteador instala una ruta en su tabla, inicia un temporizador para tal ruta. Este tiempo debe iniciarse cada vez que el ruteador recibe otro mensaje RIP anun-

---

[2] Algunos protocolos definen las conexiones directas con un costo cero.

ciando la ruta. La ruta queda inválida si transcurren 180 segundos sin que el ruteador haya recibido un anuncio nuevamente.

RIP debe manejar tres tipos de errores ocasionados por los algoritmos subyacentes. En primer lugar, dado que el algoritmo no especifica detección de ciclos de ruteo, RIP debe asumir que los participantes son confiables o deberá tomar precauciones para prevenir los ciclos. En segundo lugar, para prevenir inestabilidades, RIP debe utilizar un valor bajo para la distancia máxima posible (RIP utiliza *16*). Así, para una red de redes en la que es válido un contador de saltos de cerca de *16*, los administradores deben dividir la red de redes en secciones o utilizar un protocolo alternativo. Tercero, el algoritmo vector-distancia empleado por RIP crea un problema de *convergencia lenta* (*slow convergence*) o *conteo al infinito* (*count to infinity*), problema en el cual aparecerán inconsistencias, debido a que los mensajes de actualización de ruteo se difunden lentamente a través de la red. Seleccionando un infinito pequeño (*16*) se ayuda a limitar la convergencia lenta, pero no se elimina.

La inconsistencia en la tabla de ruteo no es exclusiva de RIP. Este es un problema fundamental que se presenta cuando cualquier protocolo vector-distancia en el que los mensajes de actualización transportan únicamente pares de redes de destino y distancias hacia estas redes. Para comprender el problema consideremos el conjunto de ruteadores mostrados en la figura 16.4 La figura describe rutas hacia la red *1* para la red de redes mostrada en la figura 16.2.

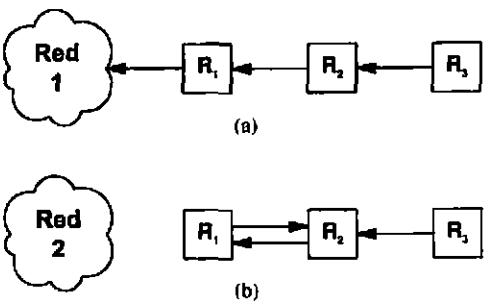


Figura 16.4 El problema de la convergencia lenta. En (a) tres ruteadores que tienen una ruta hacia la red *1*. En (b) la conexión hacia la red *1* ha desaparecido, pero R_2 ocasiona un ciclo cerrado al anunciarlo.

Como se muestra en la figura 16.4a, el ruteador R_1 tiene una conexión directa hacia la red *1*, así tiene una ruta en su tabla con la distancia *1*; éste incluye la ruta en sus difusiones periódicas. El ruteador R_2 ha aprendido la ruta desde R_1, instala la ruta en su tabla de ruteo y anuncia la ruta con una distancia igual a *2*. Finalmente, R_3 ha aprendido la ruta de R_2 y la anuncia con una distancia *3*.

Ahora, supongamos que la conexión de R_1 hacia la red *1* falla. R_1 actualizará su tabla de ruteo inmediatamente para hacer la distancia igual a *16* (infinita). En la siguiente difusión, R_1 reportará el alto costo de la ruta. Sin embargo, a menos que el protocolo incluya mecanismos extra para prevenirlo, cualquier otro ruteador podría difundir sus rutas antes que R_1. En particular, supongamos que R_2 logra anunciar sus rutas justo después de que la conexión de R_1 falló. Si esto sucede, R_1 recibirá los mensajes de R_2 y seguirá el algoritmo usual de vector-distancia: éste notará que R_2 ha anuncia-

do una ruta hacia la red *1* a un costo bajo, calculando que ahora se encuentra a *3* saltos para alcanzar la red *1* (*2* para que R_2 alcance la red *1*, más *1* para alcanzar R_2) *e instalará una nueva ruta a través de* R_2. La figura 16.4b describe el resultado. En este punto, si R_1 o R_2 reciben un datagrama destinado para la red *1*, rutearán el datagrama de regreso y así sucesivamente hasta que su tiempo de vida límite se cumpla.

En las subsecuentes difusiones de RIP para los dos ruteadores no se resolverá el problema rápidamente. En el siguiente ciclo de intercambio de ruteo, R_1 difundirá sus tablas de rutas completas. Cuando R_2 aprenda que las rutas de R_1 hacia la Red *1* tienen una longitud igual a *3*, ésta calculará una nueva longitud para tal ruta, haciéndola igual a *4*. En el tercer ciclo, R_1 recibe un reporte del incremento desde R_2 e incrementa la distancia en su tabla a *5*. Este proceso continuará contando hasta el infinito de RIP.

16.3.1 Solución al problema de la convergencia lenta

Para el ejemplo de la figura 16.4, es posible resolver el problema de la convergencia lenta mediante una técnica conocida como *actualización de horizonte separado* (*split horizon update*). Cuando se utilizan horizontes separados, un ruteador registra la interfaz por la que ha recibido una ruta particular y no difunde esta información acerca de la ruta de regreso sobre la misma interfaz. En el ejemplo, el ruteador R_2 no anunciará su ruta de longitud *2* hacia la red *1* de regreso hacia el ruteador R_1, así, si R_1 pierde su conexión hacia la red *1*, podrá detener el anuncio de la ruta. Luego de unos cuantos ciclos de actualización de ruteo, todas las máquinas podrán acordar que la red es inaccesible. Sin embargo, la separación de horizonte no abarca todas las topologías como el ejercicio lo sugiere.

Otra forma de pensar el problema de la convergencia lenta es en términos de flujo de información. Si un ruteador anuncia una ruta corta hacia alguna red, todos los ruteadores receptores responderán rápidamente instalando la ruta. Si un ruteador deja de anunciar una ruta, el protocolo deberá depender de un mecanismo de límite de tiempo antes de considerar la ruta como inaccesible. Una vez que el límite de tiempo se cumple, el ruteador busca una ruta alternativa y comienza a difundir la información. Por desgracia, un ruteador no puede saber si la ruta alternativa depende justamente de la ruta que ha desaparecido. Así, la información negativa no siempre se difunde con rapidez. Hay una frase que resume la idea y explica el fenómeno:

Las buenas noticias viajan con rapidez y las malas lentamente.

Otra técnica utilizada para resolver el problema de la convergencia lenta emplea un método conocido como *hold down* (*mantener abajo*). Esta técnica obliga a los ruteadores participantes a ignorar información acerca de una red durante un periodo de tiempo fijo luego de la recepción de un mensaje que afirma que la red es inaccesible. Por lo general, el periodo hold down se establece en 60 segundos. La idea es esperar lo suficiente como para asegurar que todas las máquinas reciban las malas noticias y no aceptan un mensaje erróneamente que está fuera de fecha. Se debe notar que todas las máquinas participantes en un intercambio RIP necesitan usar una noción idéntica de hold down o pueden presentarse ciclos de ruteo. La desventaja de la técnica hold down es que, si se presenta un ciclo de ruteo, éste se mantendrá por la duración del periodo hold down. Algo muy importante, la técnica hold down preserva todas las rutas incorrectas durante el periodo hold down, aun cuando existan alternativas.

Una técnica final para resolver el problema de la convergencia lenta se conoce como *poison reverse*.[3] Una vez que una conexión desaparece, el ruteador anuncia la conexión conservando la entrada de información por varios periodos de actualización e incluye un costo infinito en la difusión. Para hacer el poison reverse más efectivo, se debe combinar con las *triggered updates* (*actualizaciones activadas*). Las actualizaciones activadas obligan a un ruteador a que envíe una difusión inmediatamente que recibe malas noticias, en lugar de esperar el próximo periodo de difusión. Al enviar una actualización inmediatamente, un ruteador minimiza el tiempo en que es vulnerable por recibir las buenas noticias.

Por desgracia, mientras la activación de actualizaciones, el poison reverse, la técnica hold down y la de horizonte dividido resuelven algunos problemas, también inducen a otros. Por ejemplo consideremos lo que sucede con la activación de actualizaciones cuando muchos ruteadores comparten una red común. Una sola difusión puede cambiar todas las tablas de rutina, activando un nuevo ciclo de difusión. Si el segundo ciclo de difusión cambia las tablas, activará más difusiones. Esto puede conducir a una avalancha de difusiones.[4]

Tanto el uso de la difusión, que potencialmente puede provocar ciclos de ruteo, como el uso de la técnica hold down para prevenir la convergencia lenta pueden hacer que RIP sea muy ineficiente en una red de área amplia. La difusión siempre ocupa un ancho de banda importante. Aun cuando no se presenten problemas de avalancha, el hecho de que todas las máquinas tengan difusiones periódicamente significa que el tráfico se incrementa conforme el número de ruteadores aumenta. La posibilidad de que los ciclos de ruteo puedan presentarse es mortal cuando la capacidad de la línea es limitada. Una vez que las líneas comienzan a saturarse por los ciclos de paquetes, puede ser difícil o imposible para los ruteadores intercambiar los mensajes de ruteo necesarios para romper el ciclo. También en una red de área amplia, los periodos hold down son más largos que los temporizadores utilizados por los protocolos de alto nivel, lo cual puede concluir su periodo de tiempo y conducir a la ruptura de conexiones. A pesar de que estos problemas son bien conocidos, muchos grupos continúan utilizando RIP como un IGP en sus redes de área amplia.

16.3.2 Formato del mensaje RIP

Los mensajes RIP pueden ser clasificados, a grandes rasgos, en dos tipos: mensajes de información de ruteo y mensajes utilizados para solicitar información. Ambos se valen del mismo formato, consistente en un encabezado fijo seguido por una lista opcional de pares de redes y distancias. La figura 16.5 muestra el formato de los mensajes:

En la figura, el comando *COMMAND* especifica una operación de acuerdo con la siguiente tabla:

Comando	Significado
1	Solicitud para información parcial o completa de ruteo
2	Respuesta con distancias de red de pares desde la tabla de ruteo del emisor
3	Activar el modo de trazado (obsoleto)
4	Desactivar el modo de trazado (obsoleto)
5	Reservado para uso interno de Sun Microsystems

[3] N. del T.: textualmente antídoto o contraveneno.

[4] Para ayudar a evitar las colisiones, RIP requiere que cada ruteador espere un pequeño lapso de tiempo aleatorio antes de enviar una actualización de activación.

0	8	16	24	31
COMANDO (1-5)	VERSIÓN (1)	DEBE ESTAR PUESTO A CERO		
FAMILIA DE RED 1		DEBE ESTAR PUESTO A CERO		
DIRECCIÓN IP DE LA RED 1				
DEBE ESTAR PUESTO A CERO				
DEBE ESTAR PUESTO A CERO				
DISTANCIA HACIA LA RED 1				
FAMILIA DE RED 2		DEBE ESTAR PUESTO A CERO		
DIRECCION IP DE LA RED 2				
DEBE ESTAR PUESTO A CERO				
DEBE ESTAR PUESTO A CERO				
DISTANCIA HACIA LA RED 2				
...				

Figura 16.5 Formato de un mensaje RIP. Luego del encabezado de 32 bits, el mensaje contiene una secuencia de pares, donde cada par consiste en una dirección IP de red y un entero para la distancia hacia la red.

Un ruteador o anfitrión puede solicitar información de ruteo a otro para enviar un comando *request*. El ruteador responde a la solicitud mediante el comando *response*. Sin embargo, en la mayoría de los casos, los ruteadores difunden mensajes de respuestas no solicitados periódicamente. El campo *VERSION* (*VERSIÓN*) contiene el número de la versión del protocolo (actualmente *1)* y lo utiliza el receptor para verificar que interpretará el mensaje de manera correcta.

16.3.3 Convenciones de direccionamiento RIP

La generalidad de RIP es también evidente en la forma en que transmite direcciones de red. El formato de dirección no está limitado al uso con TCP/IP; puede utilizarse con múltiples conjuntos de protocolos de red. Como se muestra en la figura 16.5, cada dirección de red reportada por RIP puede tener una dirección de hasta 14 octetos. Por supuesto, las direcciones IP necesitan sólo 4; RIP especifica que los octetos restantes deben ser iguales a cero.[5] El campo *FAMILY OF NET i* identifica la familia de protocolo bajo la que la dirección de red deberá interpretarse. RIP utiliza valores asignados para familias de direcciones bajo el sistema operativo UNIX 4BSD (las direcciones IP están asignadas a un valor 2).

Además de las direcciones normales IP, RIP utiliza la convención de que la dirección *0.0.0.0* denota una *ruta por omisión*. RIP asocia una métrica de distancia para todas las rutas anunciadas,

[5] Los diseñadores seleccionaron el tercero de seis octetos para asignar la dirección IP del campo de dirección, a fin de asegurar una alineación de 32 bits.

incluyendo las rutas por omisión. Así, es posible hacer que dos ruteadores anuncien una ruta por omisión a diferentes métricas (esto es una ruta hacia el resto de la red de redes), haciendo una de ellas de ruta primaria y la otra de ruta de respaldo.

El campo final en cada entrada de información en un mensaje RIP, *DISTANCE TO NET i*, contiene un contador entero de la distancia hacia la red especificada. La distancia es medida en saltos de ruteador, pero los valores están limitados al rango entre *1* y *16*, con la distancia *16* utilizada para dar a entender una distancia infinita (esto significa que la ruta no existe).

16.3.4 Transmisión de mensajes RIP

Los mensajes RIP no contienen un campo de longitud explícito. De hecho, RIP asume que los mecanismos de entrega subyacentes dirán al receptor la longitud de un mensaje entrante. En particular, cuando se utiliza con el TCP/IP, los mensajes RIP dependen del UDP para informar al receptor la longitud del mensaje. RIP opera el puerto *520* en UDP. Aun cuando una solicitud RIP puede originar otro puerto UDP, el puerto de destino UDP para solicitudes es siempre *520*, que es el puerto de origen desde el cual en principio RIP difunde los mensajes.

El uso de RIP como protocolo de ruteo interior limita el ruteo a una métrica basada en contadores de saltos. Casi siempre los contadores de saltos proporcionan sólo una medición general de la respuesta de red o de la capacidad que no produce rutas óptimas. Además, calcular rutas con base en el conteo mínimo de saltos tiene la severa desventaja de que hace el ruteo relativamente estático, dado que las rutas no pueden responder a los cambios en las cargas de la red.

16.4 Protocolo Hello

El protocolo *HELLO* proporciona un ejemplo de un IGP que utiliza una métrica de ruteo basada en retardos en la red en lugar de contadores de saltos. A pesar de que ahora HELLO es obsoleto, es importante en la historia de Internet porque fue el IGP empleado entre los primeros ruteadores "fuzzball" de la columna vertebral NSF net. HELLO es importante para nosotros porque proporciona un ejemplo de un algoritmo vector-distancia que no utilizan contadores de saltos.

HELLO proporciona dos funciones: sincroniza los relojes entre un conjunto de máquinas y permite que cada máquina calcule las rutas de trayecto más corto hacia su destino. Así, los mensajes HELLO transportan información de sello de hora así como información de ruteo. La idea básica oculta o subyacente en HELLO es sencilla: cada máquina participante en el intercambio HELLO mantiene una tabla de sus mejores estimaciones de los relojes en las máquinas vecinas. Antes de transmitir un paquete, una máquina añade su sello de hora copiando el valor de reloj actual dentro del paquete. Cuando un paquete llega, el receptor calcula el retardo actual en el enlace. Para hacerlo, el receptor sustrae el sello de hora en el paquete entrante de su valor estimado para el reloj actual en el vecino. De manera periódica, las máquinas sondean a sus vecinos a fin de restablecer sus estimaciones para los relojes.

Los mensajes HELLO también permiten a las máquinas participantes calcular nuevas rutas. El algoritmo trabaja en forma parecida a RIP, pero utiliza retardos en lugar de contadores de salto. Cada máquina envía periódicamente a su vecino una tabla de los retardos estimados para todas las

otras máquinas. Supongamos que la máquina *A* envía a la máquina *B* una tabla de ruteo que especifica destinos y retardos. *B* examina cada entrada de información en la tabla. Si los retardos actuales de *B* para alcanzar un destino dado, *D*, son mayores que el retardo desde *A* hasta *B* más el retardo desde *B* hasta *A*, *B* cambia su ruta y envía el tráfico hacia *D* vía *A*. Esto es, *B* rutea el tráfico hacia *A* y toma la trayectoria de retraso más corto.

Como en cualquier algoritmo de ruteo, HELLO no puede cambiar rutas rápidamente o se volvería inestable. La inestabilidad en un algoritmo de ruteo produce un efecto de oscilación de dos estados en el cual el tráfico conmuta "de ida y de regreso" entre rutas alternas. En el primer estado, la máquina encuentra una trayectoria ligeramente cargada y de manera abrupta conmuta el tráfico hacia ésta, sólo para encontrar que comienza a estar completamente sobrecargada. En el segundo estado, la máquina conmuta el tráfico de regreso de la ruta sobrecargada, sólo para encontrar que la trayectoria comienza a sobrecargarse, y el ciclo continúa. Estas oscilaciones pueden presentarse. Para evitarlas, las implantaciones de HELLO seleccionan cambiar rutas sólo cuando la diferencia en el retardo es grande.

La figura 16.6 muestra el formato del mensaje HELLO. El protocolo es más complejo que el formato de mensaje indicado puesto que distingue las conexiones de redes locales de los saltos múltiples hacia afuera, los límites de tiempo caducan en entradas de información en las tablas de ruteo y utiliza identificadores locales para los anfitriones en lugar de direcciones IP completas.

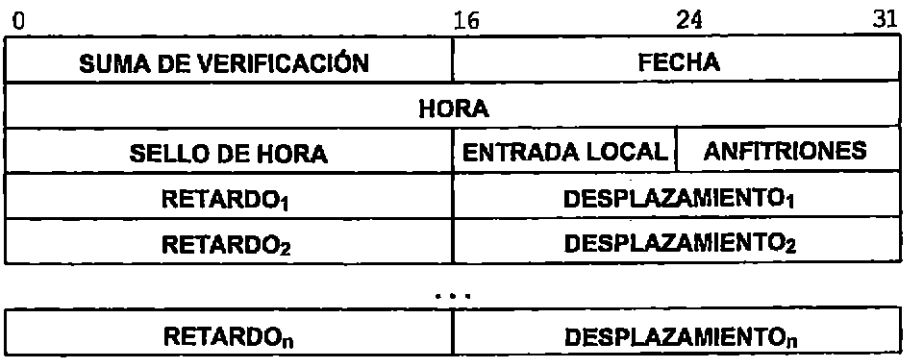


Figura 16.6 Formato del mensaje HELLO. Cada mensaje transporta una entrada de datos para la fecha y la hora, así como un sello de hora que el protocolo utiliza para estimar los retardos de red.

El campo *CHECK SUM* (*VERIFICACIÓN DE SUMA*) contiene una suma de verificación relacionada con el mensaje, el campo *DATE* (*FECHA*) contiene la fecha local del emisor y el campo *TIME* contiene la hora local de acuerdo al reloj del emisor. El campo *TIMES STAMP* (*SELLO DE HORA*) se utiliza para calcular el ciclo completo o (viaje redondo).

El campo con el nombre *#HOSTS* especifica cuántas entradas de información siguen en la lista de anfitriones y el campo llamado *LOCAL ENTRY* (*ENTRADA LOCAL*) apunta hacia la lista para marcar el bloque de entradas de información utilizadas por la red local. Cada introducción de información contiene dos campos, *DELAY* (*RETRASO*) y *OFFSET* (*DESPLAZAMIENTO*), que

proporcionan el retraso para alcanzar un anfitrión y la estimación actual del emisor respecto a la diferencia entre el reloj del anfitrión y el reloj del emisor.

16.5 Combinación de RIP, Hello y EGP

Hemos observado ya que un solo ruteador puede usar tanto un IGP para asociar información de ruteo dentro de su sistema autónomo como un EGP para anunciar rutas hacia otros sistemas autónomos. En principio sería fácil construir una sola pieza de software que combinara los dos protocolos e hiciera posible asociar rutas y anunciarlas sin la intervención humana. En la práctica obstáculos técnicos y políticos hacen que esto sea muy complejo.

Técnicamente, tanto los protocolos IGP como RIP y HELLO son protocolos de ruteo. Un ruteador utiliza estos protocolos para actualizar su tabla de ruteo con base en la información que adquiere de otros ruteadores ubicados en su sistema autónomo. A diferencia de los protocolos de ruteo interior, el EGP trabaja además con la tabla de ruteo usual de un ruteador. Un ruteador utiliza el EGP para comunicar información de accesibilidad hacia otros sistemas autónomos, independientemente de su propia tabla de ruteo. Así, *routed*, el programa UNIX que implanta RIP, anuncia información desde la tabla de ruteo local y cambia la tabla de ruteo local cuando recibe actualizaciones. Dependen de estas máquinas que utilizan RIP para transferir los datos correctamente. En contraste, el programa que implanta el EGP no anuncia rutas desde la tabla de ruteo local; mantiene una base de datos separada de accesibilidad de redes.

Un ruteador que utiliza el EGP para anunciar accesibilidad debe tener cuidado de difundir sólo las rutas que tiene autorizado anunciar o podría afectar otras partes de la red de redes. Por ejemplo, si un ruteador en un sistema autónomo logra difundir una distancia de ruta *0* para una red en la Universidad de Purdue cuando no tiene esta ruta, RIP instalará la ruta en otras máquinas y comenzará a pasar el tráfico dirigido a Purdue hacia el ruteador que inició el error. Como resultado, será imposible para algunas máquinas en el sistema autónomo llegar a Purdue. Si el EGP propaga este error fuera del sistema autónomo, podría ser imposible alcanzar Purdue desde algunas partes de la red de redes.

El programa *gated*[6] combina múltiples IGP y EGP de acuerdo a un conjunto de reglas que restringen las rutas anunciadas hacia los ruteadores exteriores. Por ejemplo, *gated* puede aceptar mensajes RIP y modificar la tabla de ruteo de las computadoras locales como sucedería con el programa *routed*. Puede anunciar rutas desde el interior de su sistema autónomo utilizando el EGP. Las reglas permiten a un administrador de sistema especificar exactamente qué redes *gated* podrán y no podrán anunciar y cómo reportarán distancias para estas redes. Así, aun cuando *gated* no es un IGP, juega un papel importante en el ruteo porque demuestra que es factible construir un mecanismo automatizado enlazando un IGP con un EGP sin sacrificar la seguridad.

Gated realiza otra tarea útil al implantar transformaciones métricas. Recordemos del capítulo 15 que las extensiones para EGP permiten a los sistemas autónomos tomar decisiones de ruteo inteligentes, mientras todos los ruteadores que utilizan EGP acuerden una interpretación amplia de la métrica de distancias. En particular, el ruteador dentro de un sistema autónomo debe acordar el uso de valores de distancia más allá de un umbral fijo, digamos 128. Cada vez que un ruteador exterior

---

[6] El nombre *gated* se pronuncia "gate d" de "gate daemon".

anuncia accesibilidad fuera de su sistema autónomo, pero dentro de su confederación autónoma, debe transformar la métrica de las distancias hacia un rango mayor (por ejemplo 128-191). La transformación tiende a mantener el tráfico dentro de un sistema autónomo incrementando artificialmente el costo hacia las rutas exteriores. Finalmente, los ruteadores transforman distancias hacia un rango mayor (esto es 192-254), cuando pasan a través de las fronteras de una confederación autónoma, para forzar al tráfico a mantenerse dentro de tal confederación. Debido a que *gated* proporciona la interfaz entre su sistema autónomo y otros sistemas autónomos, puede implantar esta transformación fácilmente.

16.6 Protocolo de SPF abierto (OSPF)

En el capítulo 14, dijimos que el algoritmo de propagación de rutas SPF escala mejor que los algoritmos vector-distancia. Un grupo de trabajo de la Fuerza de Tarea de Ingeniería de Internet ha diseñado un IGP que utiliza el algoritmo SPF. Llamado *Open SPF (OSPF)*, el nuevo protocolo se propone varios objetivos ambiciosos.

- La especificación está disponible en la información pública, lo que la hace un estándar abierto y que cualquiera puede implantar sin pagar licencias de uso. Los diseñadores esperan que muchos vendedores soporten OSPF y lo conviertan en un reemplazo popular para protocolos propietarios.
- El OSPF incluye un *ruteo de servicio de tipo*. Los administradores pueden instalar múltiples rutas hacia un destino dado, uno por cada tipo de servicio (por ejemplo, retardo bajo o rendimiento alto). Cuando se rutea un datagrama, un ruteador que corre OSPF utiliza la dirección de destino y el campo de servicio de tipo en un encabezado IP para seleccionar una ruta. El OSPF está entre los primeros protocolos TCP/IP que ofrecen un ruteo de servicio de tipo.
- El OSPF proporciona *balance de carga*. Si un administrador especifica múltiples rutas hacia un destino dado con el mismo costo, el OSPF distribuye el tráfico entre todas las rutas de la misma manera. Por el contrario, el OSPF se encuentra entre los primeros IGP abiertos en ofrecer balance de carga; los protocolos como RIP calculan una sola ruta para cada destino.
- Para permitir el crecimiento y hacer las redes de una localidad fáciles de manejar, el OSPF permite que una localidad divida sus redes y ruteadores en subconjuntos llamados *áreas*. Cada área es autónoma; el conocimiento de la topología de un área se mantiene oculto para las otras áreas. Así, varios grupos dentro de una localidad dada pueden cooperar en el uso del OSPF para rutear, lo que permite que cada grupo conserve la capacidad de cambiar su topología de red interna de manera independiente.
- El protocolo OSPF especifica que todos los intercambios entre ruteadores deben ser *autenticados*. El OSPF permite una variedad de esquemas de autenticación y también permite seleccionar un esquema para una área diferente al esquema de otra área. La idea detrás de la autenticación es garantizar que sólo ruteadores confiables difundan información de ruteo. Para entender por qué éste podría ser un problema considere qué puede suceder cuando se usa RIP, el cual no tiene la capacidad de autenticación. Si una persona maliciosa utiliza una computadora personal para propagar mensajes RIP anunciando rutas de bajo costo, otros ruteadores y anfitriones que estén corriendo RIP cambiarán sus rutas y comenzarán a enviar datagramas hacia la computadora personal.

- El OSPF soporta rutas específicas para anfitriones y rutas de subred así como rutas específicas de red. Todos estos tipos pueden ser necesarios en una red de redes extensa.
- Para adaptar redes de accesos múltiples como Ethernet, el OSPF amplía el algoritmo SPF descrito en el capítulo 14. Describimos el algoritmo utilizando un grafo de punto a punto y diciendo que cada ruteador corre SPF difundiendo periódicamente mensajes de estado de enlace sobre cada vecino accesible. Si se tienen K ruteadores conectados a una red Ethernet, éstos difundirán K^2 mensajes de accesibilidad. El OSPF minimiza la difusión permitiendo una topología de grafo complejo en el que cada nodo representa un ruteador o una red. Consecuentemente, el OSPF permite a todas las redes de accesos múltiples tener un *ruteador designado* (llamada *compuerta designada, designated gateway*, en el estándar) que envía mensajes de estado de enlace en nombre de todos los enlaces de la red a los ruteadores conectados con la red. El OSPF también se vale de capacidades de difusión de hardware, si existen, para entregar mensajes de estado de enlace.
- Para permitir una flexibilidad máxima, el OSPF permite que los administradores describan una topología de red virtual que haga abstracción de los detalles de las conexiones físicas. Por ejemplo, un administrador puede configurar un enlace virtual entre dos ruteadores en el grafo de ruteo, aunque la conexión física entre los dos ruteadores requiera de comunicaciones a través de una red de tránsito.
- El OSPF permite a los ruteadores intercambiar información de ruteo aprendida desde otras localidades (externas). Básicamente, uno o más ruteadores con conexiones hacia otras localidades reciben información sobre otras localidades y la incluyen cuando envían mensajes de actualización. El formato de mensaje distingue entre información adquirida de fuentes externas e información adquirida de ruteadores en el interior de la localidad, para evitar ambigüedad acerca de la fuente o de la confiabilidad de las rutas.

16.6.1 Formato del mensaje OSPF

Cada mensaje OSPF comienza con un encabezado fijo de 24 octetos como se muestra en la figura 16.7.

0	8	16 24 31
VERSIÓN (1)	TIPO	LONGITUD DE MENSAJE
DIRECCIÓN IP DEL RUTEADOR FUENTE		
ÁREA ID		
SUMA DE VERIFICACIÓN		TIPO DE AUTENTICACIÓN
AUTENTICACIÓN (octetos 0-3)		
AUTENTICACIÓN (octetos 4-7)		

Figura 16.7 Encabezado fijo de 24 octetos del mensaje OSPF.

El campo *VERSION* especifica la versión del protocolo. El campo *TYPE* (*TIPO*) identifica el tipo de mensaje según lo siguiente:

Tipo	Significado
1	Hello (se utiliza para pruebas de accesibilidad)
2	Descripción de Base de datos (topología)
3	Solicitud de estado de enlace
4	Actualización de estado de enlace
5	Acuse de recibo de estado de enlace

El campo *SOURCE ROUTER IP ADDRESS* (*DIRECCIÓN IP DEL RUTEADOR FUENTE*) tiene la dirección del emisor y el campo con el nombre *AREA ID* tiene un número de identificación de 32 bits para el área.

Dado que cada mensaje puede incluir la autenticación, el campo *AUTHENTICATION TYPE* (*TIPO DE AUTENTICACIÓN*) especifica qué esquema de autenticación se está utilizando (actualmente, *0* significa que no se está empleando ninguna autenticación y *1* que se está usando una simple clave de acceso).

16.6.2 Formato del mensaje Hello de OSPF

El OSPF envía mensajes *hello* en cada enlace periódicamente para establecer y probar la accesibilidad del vecino. La figura 16.8 muestra el formato.

Figura 16.8 Formato del mensaje *hello* del OSPF. Un par de ruteadores vecinos intercambian estos mensajes periódicamente para probar la accesibilidad.

El campo *NETWORK MASK* (*MÁSCARA DE RED*) contiene una máscara para la red sobre la cual se está enviando el mensaje (ver el capítulo 10 para mayores detalles sobre las máscaras). El campo *DEAD TIMER* contiene el tiempo en segundos después del cual se considera sin actividad a un vecino que no responde. El campo *HELLO INTER* es el periodo normal, en segundos, entre mensajes HELLO. El campo *GWAY PRIO* es un entero que señala la prioridad del ruteador y se utiliza en la selección de un ruteador designado de respaldo. Los campos con el nombre *DESIGNATED ROUTER* (*RUTEADOR DESIGNADO*) y *BACKUP DESIGNATED ROUTER* (*RUTEADOR DESIGNADO DE RESPALDO*) contienen direcciones IP que proporcionan la visión del emisor del ruteador designado y del ruteador designado de respaldo para la red sobre la que está enviando el mensaje. Por último, los campos llamados *NEIGHBOR$_n$ IP ADDRESS* (*DIRECCIÓN IP DE VECINO$_n$*) contienen las direcciones IP de todos los vecinos de los que el emisor ha recibido recientemente mensajes Hello.

16.6.3 Formato del mensaje de descripción de la base de datos del OSPF

Los ruteadores intercambian mensajes de *descripción de base de datos* OSPF para iniciar su base de datos de la topología de red. En el intercambio, un ruteador sirve como maestro, mientras que otro es un esclavo. El esclavo envía acuses de recibo de cada mensaje de descripción de base de datos con una respuesta. La figura 16.9 muestra el formato.

Debido a que puede ser extensa, la base de datos de la topología puede dividirse en varios mensajes utilizando los bits *I* y *M*. El bit *I* expuesto a *1* en el mensaje inicial; el bit *M* expuesto a *1* si siguen mensajes adicionales. El bit *S* indica si un mensaje fue enviado por un amo (valor *1)* o por un esclavo (valor *0)*. El campo *DATABASE SEQUENCE NUMBER* (*NÚMERO DE SECUENCIA DE BASE DE DATOS*) numera los mensajes secuencialmente, así el receptor puede saber si uno

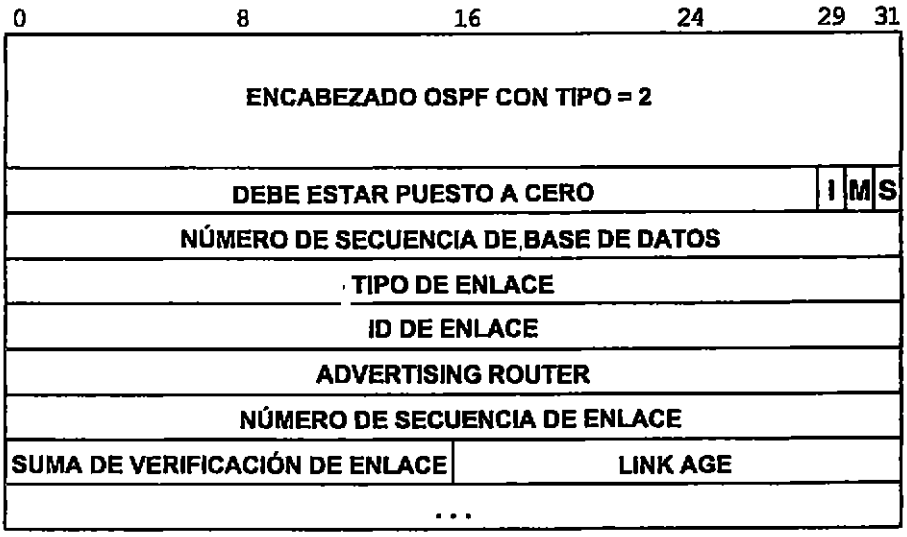


Figura 16.9 Formato del mensaje *descripción de la base de datos* OSPF. El comienzo del campo en *LINK TYPE* (*TIPO DE ENLACE*) está repetido para cada enlace especificado.

está equivocado. El mensaje inicial contiene un entero aleatorio *R*; los mensajes subsecuentes contienen enteros en secuencia que comienzan con *R*.

Los campos desde *LINK TYPE* (*TIPO DE ENLACE*) hasta *LINK AGE* describen un enlace en la topología de red; éstos se repiten para cada enlace. El campo LINK TYPE describe un enlace de acuerdo a lo descrito en la siguiente tabla.

Tipo de enlace	Significado
1	Enlace de ruteador
2	Enlace de red
3	Resumen de enlace (red IP)
4	Resumen de enlace (enlace para ruteador de frontera)
5	Enlace externo (enlace hacia otras localidades)

El campo *LINK ID* (*ENLACE ID*) contiene una identificación para el enlace (que puede ser la dirección IP de un ruteador o una red, dependiendo del tipo de enlace).

El campo *ADVERTISING ROUTER* especifica la dirección del ruteador que anuncia este enlace y *LINK SEQUENCE NUMBER (NÚMERO DE SECUENCIA DE ENLACE)* contiene un entero generado por el ruteador para asegurar que el mensaje no está equivocado o se recibe fuera de orden. El campo *LINK CHECKSUM* (*SUMA DE VERIFICACIÓN DE ENLACE*) proporciona otra garantía de que la información del enlace no se ha alterado. Por último, el campo *LINK AGE* también ayuda a ordenar los mensajes —contiene el tiempo en segundos desde que el enlace fue establecido.

16.6.4 Formato del mensaje de solicitud de estado de enlace del OSPF

Luego de intercambiar mensajes de descripción de bases de datos con un vecino, un ruteador puede descubrir que algunas partes de su base de datos están fuera de fecha. Para solicitar que el vecino proporcione información actualizada, el ruteador envía un mensaje del *solicitud de estado de enlace* (*link status request*). El mensaje lista enlaces específicos como se muestra en la figura 16.10. El

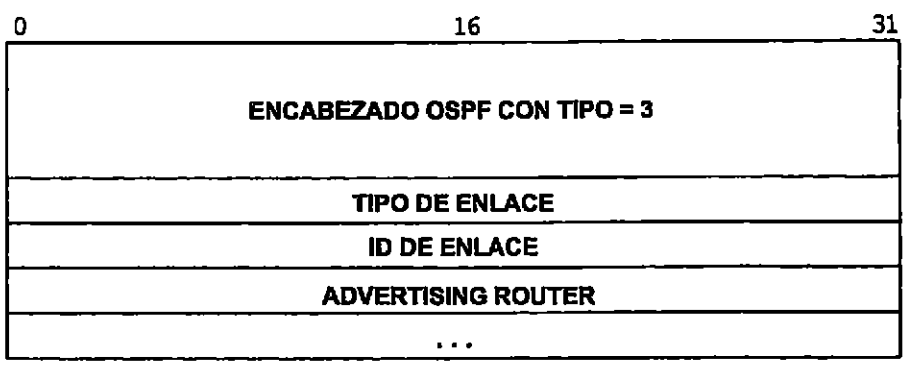


Figura 16.10 Formato del mensaje *solicitud de estado de enlace* del OSPF. Un ruteador envía este mensaje hacia un vecino para solicitar información actualizada sobre un conjunto específico de enlaces.

vecino responde con la información más actualizada que tiene en relación a estos enlaces. Los tres campos que se muestran se repiten para cada enlace del que se solicitó el estatus. Si la lista de solicitud es larga, puede ser necesario más de un mensaje de solicitud.

16.6.5 Formato del mensaje de actualización de estado de enlace OSPF

Los ruteadores difunden el estado de enlace con un mensaje de *actualización de estado de enlace* (*link status update*). Cada actualización consiste en una lista de anuncios, como se muestra en la figura 16.11.

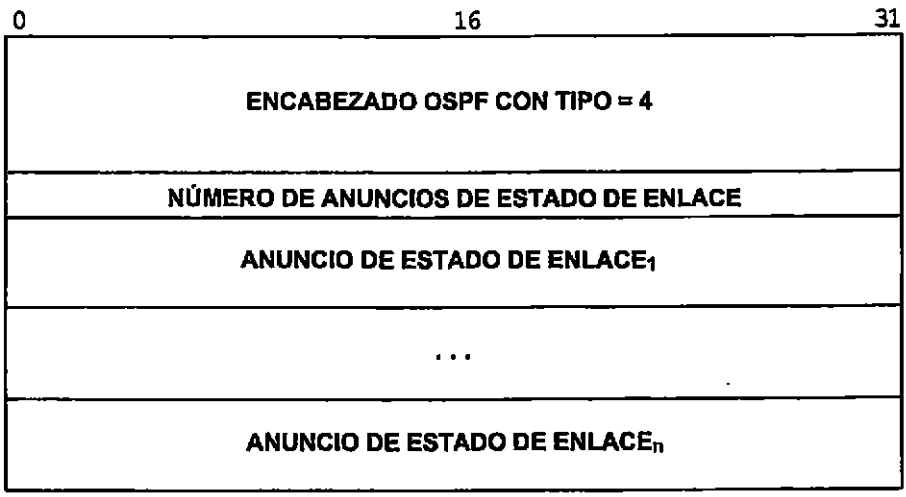


Figura 16.11 Formato del mensaje de *actualización de estado de enlace* de OSPF. Un ruteador envía este mensaje para difundir información a todos los otros ruteadores sobre sus enlaces conectados directamente.

Cada anuncio de estado de enlace tiene un formato de encabezado como se muestra en la figura 16.12. El valor utilizado en cada campo es el mismo que en el mensaje de descripción de base de datos.

A continuación del encabezado de estado de enlace, se incluye uno de cuatro posibles formatos para describir el enlace desde un ruteador hacia un área dada, el enlace desde un ruteador hacia una red específica, el enlace desde un ruteador hacia una red física que comprende una sola subred de una red IP (ver capítulo 10) o el enlace desde un ruteador hacia una red en otra localidad. En todos los casos, el campo *LINK TYPE* en el encabezado de estado de enlace especifica cuál de los formatos se ha utilizado. Así, un ruteador que recibe un mensaje de actualización de estado de enlace sabe exactamente cuál de los destinos descritos se ubica dentro de la localidad y cuáles son externos.

0	16 31
LINK AGE	TIPO DE ENLACE
ID DE ENLACE	
ADVERTISING ROUTER	
NÚMERO DE SECUENCIA DE ENLACE	
SUMA DE VERIFICACIÓN DE ENLACE	LONGITUD

Figura 16.12 Formato del encabezado utilizado para todos los anuncios de estado de enlace.

16.7 Ruteo con información parcial

Comenzamos nuestro análisis de la arquitectura de ruteo de una red de redes y del ruteo con el concepto de información parcial. Los anfitriones pueden rutear con información parcial pues dependen de los ruteadores. Debe ser claro ahora que no todos los ruteadores tienen información completa. La mayor parte de los sistemas autónomos tienen un solo ruteador que forma un puente al conectar el sistema autónomo con otros sistemas autónomos. Si la localidad está conectada con Internet, el último ruteador debe tener una conexión que se dirija desde la localidad hacia una columna vertebral de una red nacional. Los ruteadores dentro del sistema autónomo tienen conocimiento sobre los destinos dentro de este sistema autónomo, pero éstos rutearán todo el tráfico restante hacia el puente.

Hacer el ruteo con información parcial comienza a ser obvio si examinamos la tabla de ruteo de un ruteador. Los ruteadores en un sistema núcleo tienen un conjunto completo de rutas hacia todos los destinos posibles; éstos no utilizan el ruteo por omisión. De hecho, si una dirección de red de destino no aparece en las tablas del núcleo, sólo existen dos posibilidades: la dirección no es una dirección IP de destino válida o la dirección es válida pero actualmente inaccesible (por ejemplo, si el único ruteador que conducía hacia esa dirección ha fallado). Los ruteadores no-núcleo usualmente no tienen un conjunto completo de rutas; éstos dependen de una ruta por omisión para manejar direcciones de redes que no entienden.

Utilizar rutas por omisión para la mayor parte de los ruteadores no-núcleo tiene dos consecuencias. Primero, significa que los errores de ruteo locales podrían no detectarse. Por ejemplo, si una máquina en un sistema autónomo rutea incorrectamente un paquete hacia un sistema autónomo externo en lugar de hacerlo hacia un ruteador local, el sistema externo lo rutearà de regreso (posiblemente enviando un mensaje de redireccionamiento ICMP hacia la fuente original). Así, la conectividad podría parecer que se preserva incluso si el ruteo es incorrecto. El problema podría no ser severo para sistemas autónomos pequeños que tienen redes de área local de alta velocidad, pero en una red de área amplia con líneas de velocidad relativamente bajas, las rutas incorrectas pueden ser desastrosas. En segundo lugar, por el lado positivo, tener rutas por omisión significa que el mensaje de actualización de ruteo IGP será mucho más pequeño que las actualizaciones de ruteo utilizadas en un sistema núcleo.

16.8 Resumen

Los administradores deben seleccionar cómo transferir información de ruteo entre los ruteadores locales dentro de un sistema autónomo. El mantenimiento manual de la información de ruteo es suficiente sólo para pequeñas redes de redes que cambian lentamente y que tienen un mínimo de interconexiones; en la mayor parte de los casos se requiere de procedimientos automatizados que descubran y actualicen rutas automáticamente. Los ruteadores bajo el control de una sola administración corren un Protocolo de Pasarela Interior (Interior Gateway Protocol o IGP por sus siglas en inglés) para intercambiar información de ruteo.

Un IGP implementa el algoritmo de vector-distancia o el algoritmo SPF. Examinamos tres IGP específicos: RIP, HELLO y OSPF. RIP, un protocolo vector-distancia, implantado en el programa *routed* de UNIX, es el más popular. Utiliza técnicas de horizonte dividido, hold down y poison reverse a fin de ayudar a eliminar los ciclos de ruteo y el problema del conteo al infinito. A pesar de que es obsoleto, Hello es interesante porque ilustra un protocolo de vector-distancia que emplea el retardo en lugar del conteo de saltos como una métrica de distancia. Por último, OSPF es un protocolo que implanta el algoritmo de estado de enlace.

También dijimos que el programa *gated* proporciona una interfaz entre un protocolo de ruteo interior como RIP y el protocolo de pasarela exterior EGP, automatizando el proceso de asociar rutas desde dentro de un sistema autónomo y anunciándolo a otros sistemas autónomos.

PARA CONOCER MÁS

Hedrick (RFC 1058) trata los algoritmos para el intercambio de información de ruteo en general y contiene las especificaciones estándar para RIP. El protocolo HELLO está documentado en Mills (RFC 891). Mills y Braun (1987) consideran el problema de la conversión entre las métricas de retardo y de conteo de saltos. Moy (RFC 1583) contiene la especificación de longitud de OSPF, así como un análisis de la motivación subyacente en ella. Fedor (junio de 1988) describe el concepto *gated*.

EJERCICIOS

16.1 ¿Qué familias de redes soportan RIP? Sugerencia: lea la sección 4.3 sobre redes del Manual del programador de BSD de UNIX.

16.2 Considere un sistema autónomo extenso que emplee un protocolo de ruteo interior como HELLO que basa su ruteo en el retraso. ¿Qué dificultad tendrá este sistema autónomo si un subgrupo decide usar RIP en sus ruteadores?

16.3 Dentro de un mensaje RIP, cada dirección IP es alineada en una frontera de 32 bits. ¿Qué direcciones serán alineadas en una frontera de 32 bits si el datagrama IP transporta el mensaje comenzando en una frontera de 32 bits.

16.4 Un sistema autónomo puede ser tan pequeño como una red de área local o tan extenso como múltiples redes de gran alcance. ¿Por qué la variación en el tamaño hace difícil encontrar un estándar IGP?

16.5 Defina las circunstancias bajo las cuales la técnica de horizonte dividido puede prevenir la convergencia lenta.

16.6 Considere una red de redes compuesta por muchas redes de área local que corren RIP, así como un IGP. Encuentre un ejemplo que muestre cómo pueden producirse ciclos de ruteo, aun cuando el código utilice "hold down" luego de recibir información acerca de una red inaccesible.

16.7 ¿Debe un anfitrión correr RIP siempre en modo activo? ¿Por qué sí o por qué no?

16.8 Bajo qué circunstancias una métrica de conteo de saltos produciría mejores rutas que una métrica que utiliza retardos?

16.9 Puede usted imaginar una situación en la que un sistema autónomo elija *no* anunciar todas sus redes? Sugerencia: piense en una Universidad.

16.10 En términos generales podemos decir que RIP distribuye las tablas de ruteo local, mientras que el EGP distribuye una tabla de redes conocidas y ruteadores utilizados para alcanzarlas. Esto es, un ruteador puede enviar un anuncio EGP para una red sin instalar una ruta hacia esa red en su tabla de ruteo. ¿Cuáles son las ventajas de cada método?

16.11 Considere una función utilizada para convertir métricas de retardo en métricas de conteo de saltos. ¿Puede usted encontrar propiedades de cada función que sean suficientes para prevenir los ciclos de ruteo? ¿Sus propiedades también son necesarias? (Sugerencia: consulte Mills y Braun (1987)).

16.12 Bajo qué circunstancias un protocolo SPF puede formar ciclos de ruteo. Sugerencia: piense en la entrega con el mejor esfuerzo.

16.13 Construya un programa de aplicación que envíe una solicitud hacia un ruteador y que muestre las rutas recorridas.

16.14 Lea la especificación RIP cuidadosamente. ¿Las rutas reportadas en respuesta a una solicitud pueden diferir de las rutas reportadas por un mensaje de actualización de ruteo? Si es así, ¿cómo sucede?

16.15 Lea la especificación OSPF cuidadosamente. ¿Cómo puede un administrador utilizar la característica de enlace virtual?

16.16 El OSPF permite a los administradores asignar varios de sus propios identificadores, haciendo posible la duplicación de valores en varias localidades. ¿Qué identificador(es) puede necesitar cambiar si dos localidades que corren el OSPF deciden juntarse?

16.17 Compare la versión del OSPF disponible bajo 4-BSD de UNIX con la versión de RIP para el mismo sistema. ¿Cuáles son las diferencias en el tamaño del código fuente?, ¿en el tamaño del código objeto?, ¿en el tamaño de almacenamiento de datos? ¿Qué puede usted concluir?

16.18 ¿Puede utilizar mensajes de redireccionamiento ICMP para transferir información de ruteo entre ruteadores *interiores*? ¿Por qué sí o por qué no?

16.19 Escriba un programa que tome como entrada una descripción de la red de redes de su organización, utilice solicitudes RIP para obtener rutas de los ruteadores y reporte cualquier inconsistencia.

17

Multidifusión Internet (IGMP)

17.1 Introducción

En el capítulo 4, se describe las tres clases principales de direcciones IP y en el 10 se presenta el direccionamiento de subred, una extensión de dirección que permite a múltiples redes físicas compartir una sola dirección de red IP. En este capítulo, exploraremos una adición al esquema de direccionamiento IP, que permite la entrega multipunto de datagramas. Comenzaremos con una breve revisión del soporte de hardware. En secciones posteriores se describe la extensión de las direcciones IP que utilizan la entrega multipunto; también se presenta un protocolo experimental utilizado para difundir información de ruteo especial entre ruteadores.

17.2 Difusión por hardware

Muchas tecnologías de hardware tienen mecanismos para enviar paquetes hacia destinos múltiples simultáneamente (o casi simultáneamente). En el capítulo 2, se revisa varias tecnologías y se analiza la forma más común de entrega multipuntos: la *difusión*. La entrega por difusión significa que la red entrega una copia de un paquete para cada destino. En las tecnologías como Ethernet, la difusión puede completarse con la transmisión de un solo paquete. En las redes compuestas por conmutadores con conexiones punto a punto, el software tiene que implantar la difusión enviando copias de los paquetes a través de conexiones individuales hasta que todas las computadoras han recibido una copia.

En el caso de la mayor parte del hardware, el usuario especifica la entrega de difusión enviando el paquete hacia una dirección de destino especial y reservada, llamada *dirección de difu-*

sión. Por ejemplo, las direcciones de hardware de Ethernet consisten en identificadores de 48 bits en cada una de las direcciones utilizadas para la difusión indicada. El hardware en cada máquina reconoce la dirección de hardware de la máquina así como la dirección de difusión y acepta paquetes entrantes que tienen tanto una dirección como un destino.

La mayor desventaja de la difusión es que toda difusión consume recursos en todas las máquinas. Por ejemplo, sería posible destinar un conjunto de protocolos de red de redes alternativo que utilice la difusión para entregar datagramas en una red local y confiar en el software IP para descartar datagramas no proyectados por la máquina local. Sin embargo, un esquema semejante puede ser caro ya que todas las computadoras en la red local recibirán y procesarán todos los datagramas enviados en la red, aunque la mayoría de las máquinas descarte la mayor parte de los datagramas que llegan. Así, los diseñadores del TCP/IP utilizaron direcciones con la asignación de mecanismos como ARP para eliminar la entrega mediante difusión.

17.3 Multidifusión por hardware

Algunas tecnologías de hardware soportan una segunda forma de entrega de multipunto, menos común, llamada *multidifusión*. A diferencia de la difusión, la multidifusión permite que cada máquina elija si quiere participar en ella. Por lo general, una tecnología de hardware reserva un conjunto extenso de direcciones para usarse con la multidifusión. Cuando un grupo de máquinas quiere comunicarse selecciona una *dirección de multidifusión* en particular para utilizarla durante la comunicación. Luego de configurar el hardware de interfaz de red, para reconocer la dirección de multidifusión seleccionada, todas las máquinas en el grupo recibirán una copia de cada paquete enviado hacia tal dirección de multidifusión.

El direccionamiento de multidifusión puede considerarse como una generalización de todas las otras formas de difusión. Por ejemplo, podemos pensar en una *dirección de unidifusión* convencional como en una forma de direccionamiento de multidifusión en la que hay exactamente una máquina en el grupo de multidifusión. De la misma forma, podemos pensar en el direccionamiento de difusión como en una forma de multidifusión en la que cada máquina es un miembro de un grupo de multidifusión. Otras direcciones de multidifusión pueden corresponder a conjuntos indeterminados de máquinas.

Ethernet proporciona el mejor ejemplo de multidifusión en hardware. Ethernet utiliza el bit de orden menor del octeto de orden mayor para distinguir la dirección de unidifusión convencional (con valor *0*) de la dirección de multidifusión (con valor *1*). En la notación hexadecimal con puntos,[1] el bit de multidifusión se toma como

$$01.00.00.00.00.00_{16}$$

Inicialmente, el hardware de interfaz de red está configurado para aceptar paquetes destinados a la dirección de difusión de Ethernet o a la dirección de hardware de la máquina. Sin embar-

[1] La notación hexadecimal con puntos representa cada octeto como dos dígitos hexadecimales separados por puntos; el subíndice *16* se puede omitir sólo cuando el contexto es suficientemente claro.

go, una interfaz puede también reconfigurarse con facilidad a fin de permitir el reconocimiento de un pequeño conjunto de direcciones de multidifusión.

17.4 Multidifusión IP

La *multidifusión IP* es la abstracción de red de redes del hardware de multidifusión. Permite la transmisión de un datagrama IP a un conjunto de anfitriones que forma un solo grupo de multidifusión. Es posible para los miembros del grupo propagarse a través de redes físicas separadas. La multidifusión IP emplea el mismo concepto de entrega con el mejor esfuerzo que en otras entregas de datagramas IP, esto significa que los datagramas de multidifusión pueden perderse, borrarse, duplicarse o entregarse sin orden.

La pertenencia a un grupo de multidifusión IP es un proceso dinámico. Un anfitrión puede unirse o abandonar un grupo en cualquier momento. Además, un anfitrión puede ser miembro de un número indeterminado de grupos de multidifusión. Los miembros de un grupo determinan si el anfitrión recibirá datagramas enviados hacia el grupo de multidifusión; un anfitrión puede enviar datagramas hacia un grupo de multidifusión sin ser un miembro.

Cada grupo de multidifusión tiene una dirección de multidifusión única (de clase *D*). Como los puertos de protocolo, algunas direcciones de multidifusión IP son asignadas por la autoridad de Internet y corresponden a grupos que siempre existen aun cuando actualmente no tengan miembros. Estas direcciones se dice que son *bien conocidas*. Otras direcciones de multidifusión están disponibles para usos temporales. Corresponden a *grupos transitorios de multidifusión* que se crean cuando son necesarios y se desechan cuando el número de miembros llega a cero.

La multidifusión IP puede utilizarse en una sola red física o a través de una red de redes. En este último caso, *ruteadores de multidifusión* especiales envían los datagramas de multidifusión. Sin embargo, los anfitriones no necesitan saber explícitamente sobre rutas de multidifusión. Los anfitriones admiten datagramas de multidifusión que utilicen las capacidades de multidifusión de la red local. Si un ruteador de multidifusión está presente, recibirá el datagrama y lo enviará hacia otra red conforme sea necesario. Los ruteadores de multidifusión utilizarán las capacidades de multidifusión de hardware local para entregar el datagrama en la red o las redes de destino que soporten la multidifusión. El campo de tiempo de vida en un datagrama de multidifusión limita la difusión a través de los ruteadores de la misma manera que el campo de tiempo de vida en un datagrama de unidifusión limita su difusión. La transmisión de multidifusión la pueden proporcionar por ruteadores independientes físicamente, o bien, puede añadirse esta capacidad a los ruteadores convencionales.

El estándar TCP/IP para multidifusión define el direccionamiento de multidifusión IP, especifica cómo envían y reciben los anfitriones datagramas de multidifusión y describe el protocolo que los ruteadores utilizan para determinar los miembros de un grupo de multidifusión en una red. En la siguiente sección, se examina cada aspecto con mayor detalle.

17.5 Direcciones de multidifusión IP

Como en el hardware de multidifusión, la multidifusión IP se vale de la dirección de destino de datagrama para especificar una entrega de multidifusión. La multidifusión IP utiliza direcciones clase *D* en la forma en que se muestra en la figura 17.1.

Figura 17.1 Formato de las direcciones IP de clase *D* utilizado para la multidifusión IP. Los bits del *4* a *31* identifican un grupo de multidifusión en particular.

Los primeros *4* bits contienen *1110* e identifican la dirección como una multidifusión. Los 28 bits restantes especifican un grupo de multidifusión particular. No hay otra estructura en el grupo de bits. En particular, el campo de grupo no identifica el origen del grupo ni contiene una dirección de red como en las direcciones de clase *A*, *B* y *C*.

Cuando se expresa en notación decimal con puntos, el rango de direcciones de multidifusión abarca de

224.0.0.0 a 239.255.255.255

Sin embargo, la dirección 224.0.0.0 está reservada; no se puede asignar a ningún grupo. Además, la dirección 224.0.0.1 está asignada permanentemente al *grupo de todos los anfitriones*, el cual incluye a todos los anfitriones y ruteadores que participan en la multidifusión IP. En general, la dirección del grupo de todos los anfitriones se utiliza para alcanzar todas las máquinas que participan en la multidifusión IP en una red local; no hay direcciones de multidifusión IP que hagan referencia a todos los anfitriones en la red de redes.

Las direcciones de multidifusión IP sólo pueden emplearse como direcciones de destino. Éstas nunca podrán aparecer en el campo de dirección de la fuente de un datagrama ni pueden aparecer en una ruta fuente o en el registro de una opción de rutas. Sin embargo, no hay manera de que se generen mensajes de error ICMP relacionados con datagramas de multidifusión (por ejemplo, destinos inaccesibles, una fuente desactivada o eco de réplica por tiempo excedido).

17.6 Transformación de multidifusión IP en multidifusión Ethernet

Aun cuando el estándar no cubre todos los tipos de hardware de red, especifica *cómo transformar* las direcciones de multidifusión IP en direcciones de multidifusión Ethernet. La transformación es eficaz y fácil de entender:

Para transformar una dirección de multidifusión IP en su correspondiente dirección de multidifusión Ethernet, colocar los 23 bits de orden menor de la dirección de multidifusión IP dentro de los 23 bits de orden inferior de la dirección de multidifusión Ethernet especial $01.00.5E.00.00.00_{16}$.

Por ejemplo, la dirección de multidifusión IP 224.0.0.1 se convierte en la dirección de multidifusión Ethernet $01.00.5E.00.00.01_{16}$.

Es interesante el hecho de que la transformación no es única. Dado que las direcciones de multidifusión IP tienen 28 bits significativos que identifican al grupo de multidifusión, más de un grupo puede transformarse en la misma dirección de multidifusión Ethernet. Los diseñadores seleccionan este esquema como un compromiso. Por un lado, utilizar 23 de los 28 bits para una dirección de hardware significa que la mayor parte de las direcciones de multidifusión están incluidas. El conjunto de direcciones es lo suficientemente extenso como para que la posibilidad de que dos grupos seleccionen direcciones idénticas con los 23 bits de orden inferior sea muy pequeña. Por otro lado, el arreglo para IP que utiliza una parte fija del espacio de dirección de multidifusión Ethernet hace que la depuración sea mucho más fácil y elimina la interfaz entre IP y otros protocolos que comparten una red Ethernet. La consecuencia de este diseño es que algunos datagramas de multidifusión pueden recibirse en un anfitrión, aunque los datagramas no estén destinados a tal anfitrión. Así, el software IP debe verificar cuidadosamente las direcciones en todos los datagramas entrantes y descartar cualquier datagrama no esperado.

17.7 Extensión de IP para manejar la multidifusión

Un anfitrión participa en una multidifusión IP en uno de tres niveles, como se muestra en la figura 17.2:

Nivel	Significado
0	El anfitrión no puede ni enviar ni recibir multidifusión IP
1	El anfitrión puede enviar pero no recibir multidifusión IP
2	El anfitrión puede enviar y recibir multidifusión IP

Figura 17.2 Los tres niveles de participación en la multidifusión IP.

Las modificaciones que permiten a una máquina enviar multidifusión IP no son difíciles. El software IP debe permitir a un programa de aplicación especificar una dirección de multidifusión como una dirección IP de destino y el software de interfaz de red debe ser capaz de transformar una dirección de multidifusión IP en la correspondiente dirección de multidifusión de hardware (o utilizar la difusión si el hardware no soporta la multidifusión).

Ampliar el software del anfitrión para recibir datagramas de multidifusión IP es más complejo. El software IP en el anfitrión debe tener una interfaz que permita a un programa de aplicación declarar si desea unirse o abandonar un grupo de multidifusión en particular. Si diversos programas

de aplicación se unen al mismo grupo, el software IP debe recordar cada uno de ellos para transferir una copia de los datagramas que llegan destinados para este grupo. Si todos los programas de aplicación abandonan un grupo, el anfitrión debe recordar que no quedan participantes en el grupo. Además, como veremos en la siguiente sección, el anfitrión tiene que correr un protocolo que informe a los ruteadores de multidifusión locales del estado de los miembros de un grupo. Buena parte de la complejidad asociada con esto proviene de una idea básica:

Los anfitriones se unen a grupos de multidifusión IP específicos en redes específicas.

Esto es, un anfitrión con diversas conexiones de red puede unirse a un grupo de multidifusión en particular en una red y no en otra. Para entender la razón de mantener a un grupo de miembros asociados con una red, recordemos que es posible utilizar la multidifusión IP entre conjuntos locales de máquinas. El anfitrión podría desear utilizar una aplicación de multidifusión para interactuar con máquinas en una red física, pero no con las máquinas en otra red.

Dado que un grupo de miembros está asociado con una red en particular, el software debe mantener listas separadas de direcciones de multidifusión para cada red a la que la máquina está conectada. Además, un programa de aplicación debe especificar una red particular cuando solicita unirse o abandonar un grupo de multidifusión.

17.8 Protocolo de gestión de grupos de Internet

Para participar en la multidifusión IP dentro de una red local, un anfitrión debe tener el software que le permita enviar y recibir datagramas de multidifusión. Para participar en una multidifusión que cubra varias redes, el anfitrión debe informar a los ruteadores de multidifusión local. El ruteador local se pone en contacto con otros ruteadores de multidifusión, pasando información hacia los miembros y estableciendo rutas. La idea es muy similar a la difusión de rutas tradicional entre ruteadores de red de redes convencionales.

Antes de que un ruteador de multidifusión pueda difundir información a los miembros de multidifusión, debe determinar si uno o más anfitriones en la red local han decidido unirse a un grupo de multidifusión. Para hacerlo, los ruteadores de multidifusión y los anfitriones que implantan la multidifusión deben utilizar el Protocolo de Administración de Grupos de Internet (*Internet Group Management Protocol* o *IGMP* por sus siglas en inglés) para comunicar información a los miembros del grupo.

El IGMP es análogo al ICMP.[2] Como el ICMP, éste utiliza datagramas IP para transportar mensajes. También, como el ICMP, el IGMP proporciona un servicio utilizado por el IP. Sin embargo,

Aun cuando el IGMP se vale de datagramas IP para transportar mensajes, pensamos a éste como una parte integral del IP, no como un protocolo separado.

---

[2] En el capítulo 9, se analiza el ICMP, el Protocolo de Mensajes de Control de Internet.

Además, el IGMP es un estándar para el TCP/IP; éste es requerido en todas las máquinas que participan en multidifusión IP en el nivel *2*.

Conceptualmente, el IGMP tiene dos fases. Fase 1: cuando un anfitrión se une a un nuevo grupo de multidifusión envía un mensaje IGMP para la dirección de multidifusión "todos los anfitriones", declarando su membresía. Los ruteadores de multidifusión local reciben el mensaje y establecen el ruteo necesario para difundir la información de membresía del grupo hacia otros ruteadores de multidifusión a través de la red de redes. Fase 2: debido a que la membresía es dinámica, los ruteadores de multidifusión local muestrean de manera periódica a los anfitriones en la red local para determinar qué anfitriones se mantienen como miembros de qué grupos. Si en un grupo no se reportan miembros después de varios muestreos, el ruteador de multidifusión asume que no hay anfitriones en la red que se mantengan en el grupo y deja de anunciar miembros del grupo a otros ruteadores de multidifusión.

17.9 Implantación IGMP

El IGMP está diseñado cuidadosamente para evitar congestionamientos en una red local. En primer lugar, toda la comunicación entre anfitriones y ruteadores de multidifusión utilizan multidifusión IP. Esto es, cuando los mensajes IGMP están encapsulados en un datagrama IP para su transmisión, la dirección de destino IP es la dirección de multidifusión de todos los anfitriones. Así, los datagramas que transportan mensajes IGMP son transmitidos mediante hardware de multidifusión si éste está disponible. Como resultado, en las redes, en las que el hardware soporta la multidifusión, los anfitriones que no participan en la multidifusión IP nunca reciben mensajes IGMP. En segundo lugar, un ruteador de multidifusión no enviará mensajes de solicitud individuales para cada grupo de multidifusión, sino un mensaje de muestreo para solicitar información relacionada con la membresía en todos los grupos. La cantidad de muestreos está restringida, a lo sumo, a una solicitud por minuto. En tercer lugar, los anfitriones que son miembros de varios grupos no envían respuestas múltiples al mismo tiempo. Por el contrario, luego de que un mensaje de solicitud IGMP llega desde un ruteador de multidifusión, el anfitrión asigna un retardo aleatorio de entre *0* y *10* segundos para cada grupo en el que tiene miembros, y envía una respuesta para este grupo después del retardo. Así, un anfitrión separa sus respuestas aleatoriamente dentro de un lapso de *10* segundos. En cuarto lugar, los anfitriones escuchan las respuestas de otros anfitriones y suprimen cualquiera de estas respuestas que sea innecesaria.

Para entender por qué una respuesta puede ser innecesaria, recordemos por qué los ruteadores envían un mensaje de muestreo. Los ruteadores no necesitan conservar un registro exacto de los miembros de un grupo porque todas las transmisiones hacia el grupo serán enviadas mediante hardware de multidifusión. De hecho, los ruteadores de multidifusión sólo necesitan saber si se conserva un miembro del grupo en el último anfitrión en la red. Luego de que un ruteador de multidifusión envía un mensaje de muestreo, todos los anfitriones asignan un retardo aleatorio para la respuesta. Cuando el anfitrión con el retardo más pequeño envía su respuesta (mediante la multidifusión), otros anfitriones participantes reciben una copia. Cada anfitrión asume que el ruteador de multidifusión también recibió una copia de la primera respuesta y cancelan sus respuestas. Así, en la práctica, sólo un anfitrión de cada grupo responde a un mensaje de solicitud desde el ruteador de multidifusión.

17.10 Transiciones del estado de la membresía de grupo

El IGMP debe recordar el estado de cada grupo de multidifusión al que el anfitrión pertenece. Podríamos pensar que un anfitrión conserva una tabla en la que registra la información de los miembros de un grupo. Inicialmente, todos los espacios en la tabla estarán sin usarse. Cada vez que un programa de aplicación en el anfitrión se una a un nuevo grupo, el software IGMP asignará un espacio y lo llenará con información acerca del grupo. Entre la información, el IGMP establecerá un contador de referencia de grupo, el cual se inicia en *1*. Si aplicaciones adicionales se unen al grupo, el IGMP incrementará el contador de referencia en la información almacenada. Conforme los programas de aplicación abandonan el grupo, el IGMP decrementa el contador. El anfitrión deja el grupo de multidifusión cuando el contador llega a cero.

Las acciones que el software IGMP toma en respuesta a los mensajes IGMP pueden explicarse mejor mediante el diagrama de transición de estados que se muestra en la figura 17.3.

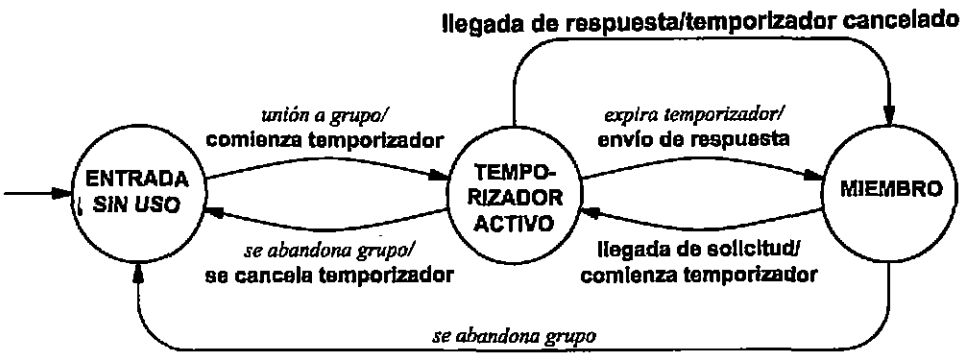


Figura 17.3 Los tres posibles estados de una entrada de información en la tabla del grupo de multidifusión de un anfitrión y las transiciones entre estos estados. Las transiciones son provocadas por la llegada de un mensaje IGMP o por eventos en el anfitrión (que se muestran en cursivas).

Como se puede observar en la figura 17.3, un sólo mecanismo temporizador puede utilizarse para generar tanto el mensaje de respuesta inicial como la respuesta a la solicitud desde el ruteador de multidifusión. Una solicitud de unión a un grupo coloca la entrada de información en el estado *TIMER ACTIVE* (*TEMPORIZADOR ACTIVO*) y ajusta el temporizador con un valor pequeño. Cuando el temporizador expira, el IGMP genera y envía un mensaje de respuesta y cambia la entrada de información al estado *MEMBER* (*MIEMBRO*).

En el estado *MEMBER*, la recepción de una solicitud IGMP ocasiona que el software elija un valor de tiempo excedido (aleatoriamente), inicie un temporizador para la entrada de información y cambie la información de entrada al estado *TIMER ACTIVE*. Si otros anfitriones envían una respuesta para el grupo de multidifusión, luego de que el temporizador expira, el IGMP cancela el temporizador y cambia la entrada de información de nuevo al estado *MEMBER*.

17.11 Formato de los mensajes IGMP

Como se muestra en la figura 17.4, los mensajes IGMP tienen un formato simple.

0	4	8	16 31
VERS	TYPE	SIN USO	SUMA DE VERIFICACIÓN
DIRECCIÓN DE GRUPO (CERO EN SOLICITUD)			

Figura 17.4 Formato de un mensaje IGMP.

El campo *VERS* conserva la información sobre la versión del protocolo (el valor actual es *1*). El campo *TYPE* identifica al mensaje como una solicitud enviada por un ruteador de multidifusión (*tipo 1*) o como una respuesta enviada por un anfitrión (*tipo 2*). El campo *UNUSED (SIN USO)* debe contener 0 y el campo *CHECKSUM* (*SUMA DE VERIFICACIÓN*) contiene una suma de verificación para el mensaje IGMP de 8 octetos. (La suma de verificación IGMP se calcula con el mismo algoritmo utilizado para las sumas de verificación TCP e IP.) Por último, los anfitriones emplean el campo *GROUP ADDRESS* (*DIRECCIÓN DE GRUPO*) para reportar su membresía en un grupo de multidifusión en particular (en una solicitud, el campo contiene ceros y no tiene ningún significado).

17.12 Asignación de direcciones de multidifusión

El estándar no especifica exactamente cómo son asignados los grupos de máquinas a las direcciones de multidifusión, pero sugiere varias posibilidades. Por ejemplo, si el sistema operativo local asigna un identificador entero a un conjunto de procesos o a un conjunto de aplicaciones, este identificador puede usarse para formar una dirección de multidifusión IP. Por supuesto, es posible para un administrador de red asignar direcciones manualmente. Otra posibilidad es permitir que una máquina forme aleatoriamente direcciones de multidifusión hasta que descubra una que no se encuentre en uso.

17.13 Difusión de información de ruteo

Aun cuando la multidifusión IP descrita en este capítulo es un estándar para el TCP/IP, no existe un estándar para la difusión de información de ruteo entre ruteadores de multidifusión. Sin embargo, la literatura describe un protocolo experimental llamado *Protocolo de Ruteo de multidifusión Vector Distancia (Distance Vector Multicast Routing Protocol, DVMRP)*. Los ruteadores de multidifusión utilizan el *DVMRP* para transferir información de membresía entre ellos. Se valen de la información para establecer rutas y, así, entregar la copia de un datagrama de multidifusión a todos los miembros del grupo de multidifusión.

El DVMRP recuerda al protocolo RIP descrito en el capítulo 16, pero incorpora ideas que lo hacen más eficiente y complejo. En esencia, el protocolo transfiere información sobre los miembros del grupo de multidifusión actuales y sobre el costo de alcance entre ruteadores. Para cada posible grupo de multidifusión, el ruteador impone un árbol de ruteo en la parte superior del grafo de las interconexiones físicas. Cuando un ruteador recibe un datagrama destinado a una dirección de multidifusión IP, envía una copia del datagrama hacia los enlaces de red que corresponden con las ramas en el árbol de ruteo.

El DVMRP utiliza mensajes IGMP para transportar información. Define tipos de mensajes IGMP que permiten a los ruteadores declarar la membresía en un grupo de multidifusión, abandonar un grupo de multidifusión e interrogar a otros ruteadores. La extensión también proporciona mensajes que transportan información de ruteo, incluyendo costos métricos. El protocolo se ha implantado, pero es necesaria una mayor experimentación antes de llegar a conclusiones sobre su desempeño.

17.14 El programa mrouted

Mrouted es un programa bien conocido que maneja ruteo de multidifusión en sistemas UNIX. Como *routed*,[3] *mrouted* coopera de cerca con el núcleo del sistema operativo para instalar información de ruteo de multidifusión. A diferencia de *routed, mrouted* no utiliza la tabla de ruteo estándar. De hecho, puede utilizarse sólo con una versión especial del sistema UNIX, conocida como *núcleo de multidifusión*. Un núcleo de multidifusión UNIX contiene una tabla de ruteo de multidifusión especial así como el código necesario para enviar datagramas de multidifusión. *Mrouted* maneja:

- *Propagación de ruta. Mrouted* utiliza el DVMRP para difundir información de ruteo de multidifusión de un ruteador a otro. Una computadora que corre *mrouted* también interpreta la información de ruteo de multidifusión y elabora una tabla de ruteo de multidifusión mediante un algoritmo conocido como *Truncated Reverse Path Broadcast (TRPB). Mrouted* no reemplaza a los protocolos de propagación de ruta convencionales; una computadora por lo general corre *mrouted* además del software de protocolo de ruteo estándar.
- *Procedimiento de túnel de multidifusión.* Uno de los mayores problemas con la multidifusión en las redes de redes se debe a que no todos los ruteadores de red de redes pueden enviar datagramas de multidifusión. *Mrouted* puede implementar un *túnel* para datagramas de multidifusión de un ruteador a otro a través de ruteadores intermedios que no participen en el ruteo de multidifusión.

Aun cuando un solo programa *mrouted* puede realizar las dos tareas, una computadora dada no necesitaría ambas funciones. Para permitir que un administrador defina exactamente cómo debe operar, *mrouted* utiliza un archivo de configuración. El archivo de configuración contiene entradas de datos que especifican los grupos de multidifusión *mrouted* que está permitido anunciar en cada

---

[3] Recuerde que *routed* es el programa de UNIX que implanta RIP.

interfaz y cómo deben enviarse los datagramas. Además, el archivo de configuración asocia una métrica y un umbral con cada ruta. La métrica permite a un administrador asignar un costo para cada ruta (por ejemplo, para asegurar que el costo asignado a una ruta sobre una red de área local sea menor que el costo de una ruta a través de un enlace serial lento). El umbral proporciona el *tiempo de vida* (*time to live*, *TTL*) IP mínimo que un datagrama necesita para completar la ruta. Si un datagrama no tiene un tiempo de vida TTL suficiente para alcanzar su destino, un núcleo de multidifusión no enviará el datagrama. De hecho, descartará el datagrama, con lo cual se evita el desperdicio del ancho de banda.

El procedimiento mediante túneles de multidifusión es posiblemente la capacidad más interesante de *mrouted*. Un túnel es necesario cuando a) dos o más computadoras desean participar en aplicaciones de multidifusión, y b) cuando uno o más ruteadores, en la parte de la red de redes que separa a las computadoras participantes, no corre software de ruteo de multidifusión. La figura 17.5 ilustra este concepto.

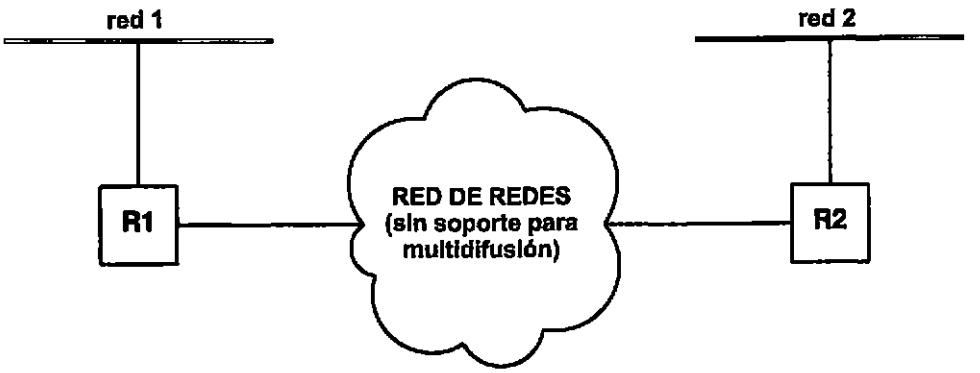


Figura 17.5 Ejemplo de la configuración de una red de redes que requiere de un procedimiento mediante túneles para las computadoras conectadas a la red *1* y *2* a fin de participar en la comunicación de multidifusión. Los ruteadores en la red de redes que separan a las dos redes no difunden rutas de multidifusión y no pueden enviar datagramas hacia una dirección de multidifusión.

Para permitir que las computadoras en las redes separadas se comuniquen mediante la multidifusión, los administradores de ruteadores de cada localidad configuran *mrouted* a fin de utilizar un *túnel* para la comunicación entre dos localidades. De hecho, el túnel consiste únicamente en un acuerdo entre los programas *mrouted* que corren en los dos ruteadores. Cada ruteador escucha en su red local si existen datagramas enviados hacia el grupo de multidifusión para los que el túnel ha sido configurado. Cuando un datagrama de multidifusión llega y la dirección de destino corresponde a la del túnel, *mrouted* envía el datagrama al *mrouted* en el otro ruteador empleando una dirección de unidifusión IP convencional. Cuando recibe un datagrama de unidifusión a través del túnel, *mrouted* extrae el datagrama de multidifusión y entonces utiliza el hardware de multidifusión para entregar el datagrama a las computadoras en su red local.

¿Cómo pueden dos programas enviar un datagrama de multidifusión por medio de una dirección de unidifusión? La respuesta es la encapsulación. *Mrouted* almacena información de ruteo de multidifusión en el núcleo, lo cual ocasiona que el núcleo coloque el datagrama de multidifusión completo dentro de un datagrama convencional IP, como se muestra en la figura 17.6.

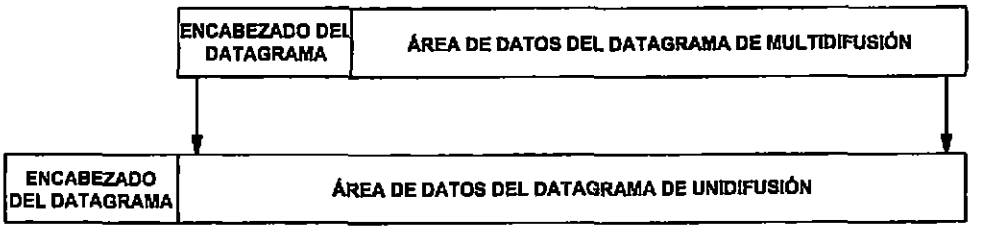


Figura 17.6 Un datagrama de multidifusión encapsulado en un datagrama IP convencional. Los ruteadores de multidifusión utilizan esta encapsulación para realizar el procedimiento mediante túneles con el tráfico de multidifusión a través de ruteadores que no manejan la multidifusión. Luego de atravesar el túnel, el ruteador receptor extrae el datagrama de multidifusión y emplea la dirección de destino de multidifusión para entregarlos.

Como se muestra en la figura, el datagrama de multidifusión, incluyendo el encabezado, viaja dentro del área de datos de un datagrama de unidifusión convencional. En la máquina de recepción, el núcleo de multidifusión extrae y procesa el datagrama de multidifusión como si llegara en una interfaz local. En particular, la máquina de recepción decrementa en 1 el campo de tiempo de vida en el encabezado, antes de enviar el datagrama de multidifusión. Así, cuando crea un túnel, *mrouted* trata a la red de redes conectada a dos ruteadores de multidifusión como una sola red física. Note que un datagrama de unidifusión que transporta un datagrama de multidifusión tiene su propio contador de tiempo de vida, el cual opera independientemente con respecto al contador de tiempo de vida en el encabezado del datagrama de multidifusión. De este modo, es posible limitar el número de saltos físicos a través de un túnel dado, independientemente del número de saltos lógicos que un datagrama de multidifusión deba recorrer en su jornada, de la fuente original al destino final.

Los túneles de multidifusión forman la base del *Multicast Backbone* de Internet (*MBONE*). El MBONE consiste en un conjunto de ruteadores que acuerda enviar tráfico de multidifusión a través de Internet. Ha sido utilizado por servicios que incorporan teleconferencias con audio y video.

17.15 Resumen

La multidifusión IP es una abstracción de la multidifusión de hardware. Ésta permite una entrega eficiente de datagramas a múltiples destinos. El IP utiliza las direcciones de clase *D* para especifi-

car la entrega de multidifusión, las transmisiones actuales se valen de la multidifusión de hardware si está disponible.

Los grupos de multidifusión IP son dinámicos: un anfitrión puede unirse o abandonar un grupo en cualquier momento. Para la multidifusión local, los anfitriones sólo necesitan contar con la capacidad de enviar y recibir datagramas de multidifusión. Sin embargo, la multidifusión IP no está limitada a una sola red física —los ruteadores de multidifusión difunden información sobre la membresía de grupos y arreglan las rutas de manera que cada miembro de un grupo de multidifusión reciba una copia de todos los datagramas enviados al grupo.

Los anfitriones comunican su membresía de grupo a los ruteadores de multidifusión mediante el IGMP. El IGMP ha sido diseñado para ser eficiente y para que no ocupe recursos de red. En la mayor parte de los casos, el solo tráfico del IGMP introduce un mensaje periódico desde un ruteador de multidifusión y una sola réplica para cada grupo de multidifusión al que pertenecen los anfitriones de esta red.

No todos los ruteadores en la red global de Internet difunden rutas de multidifusión o envían tráfico de multidifusión. Los grupos de dos o más localidades separadas por una red de redes que no soporta el ruteo de multidifusión pueden emplear un túnel IP para transferir datagramas de multidifusión. Cuando se utiliza un túnel, un programa encapsula un datagrama de multidifusión en un datagrama de unidifusión convencional. El receptor debe extraerlo y manejar el datagrama de multidifusión.

PARA CONOCER MÁS

Deering (RFC 1112) especifica el estándar para la multidifusión IP, descrito en este capítulo. Waitzman, Partridge y Deering (RFC 1075) describen la difusión de rutas de multidifusión mediante un protocolo similar al RIP. Los primeros bosquejos de estas ideas se pueden encontrar en Deering (RFC 1054 y 988) y en Deering y Cheriton (RFC 966). Deering y Cheriton (mayo 1990) consideran las modificaciones de varios algoritmos de ruteo para soportar multidifusión de área amplia. Se puede encontrar información sobre *mrouted* en las páginas del manual distribuido con el programa.

Eriksson (1994) explica la multidifusión de red de columna vertebral. Casner y Deering (julio 1992) reportan la primera multidifusión de un encuentro IETF.

EJERCICIOS

17.1 El estándar sugiere utilizar 23 bits de una dirección de multidifusión IP para formar una dirección de multidifusión de hardware. En un esquema como éste, ¿cuántas direcciones de multidifusión IP se transforman en una sola dirección de multidifusión de hardware?

17.2 Explique por qué una dirección de multidifusión IP debe utilizar sólo 23 de los 28 posibles bits. Sugerencia: ¿cuáles son los límites prácticos en el número de grupos a los que puede pertenecer un anfitrión y el número de anfitriones en una sola red?

17.3 El IP siempre debe verificar la dirección de destino en los datagramas de multidifusión entrantes y descartar los datagramas si el anfitrión no está en el grupo de multidifusión especificado. Explique cómo puede recibir el anfitrión una multidifusión destinada a un grupo del que el anfitrión no es miembro.

17.4 ¿Hay alguna ventaja en tener ruteadores de multidifusión que conozcan el conjunto de anfitriones en la red local que pertenece a un grupo de multidifusión dado?

17.5 Encuentre tres aplicaciones en su ambiente que se puedan beneficiar de la multidifusión IP.

17.6 El estándar dice que el software IP debe arreglarse para entregar una copia de cualquier datagrama de multidifusión, que sale hacia programas de aplicación, en el anfitrión que pertenece al grupo de multidifusión especificado. ¿Este diseño hace la programación más fácil o más difícil?

17.7 Cuando el hardware subyacente no soporta la multidifusión, la multidifusión IP utiliza el hardware de difusión para la entrega. ¿De qué manera esto podría causar problemas? ¿Existe alguna ventaja de utilizar la multidifusión IP en estas redes?

17.8 Lea sobre el DVMRP en RFC 1075. ¿Qué es lo que hace al DVMRP más complejo que al RIP?

17.9 La dirección de multidifusión IP de todos los anfitriones se refiere sólo a la red local, mientras que todas las otras direcciones de multidifusión IP se refieren a los grupos de multidifusión para redes de redes amplias. Explique por qué podría ser ventajoso reservar un conjunto de direcciones de multidifusión IP para uso local solamente.

17.10 El IGMP no incluye una estrategia para los acuses de recibo o la retransmisión, aun cuando se usa en redes que emplean la entrega con el mejor esfuerzo. ¿Qué sucede si una solicitud se pierde? ¿Qué sucede si una respuesta se pierde?

17.11 Explique por qué un anfitrión de localidades múltiples podría necesitar unirse a un grupo de multidifusión en una red y no en otra. Sugerencia: considere una teleconferencia de audio.

18

TCP/IP en redes ATM

18.1 Introducción

En capítulos anteriores se explican partes fundamentales del TCP/IP y se muestra cómo los componentes operan en redes LAN y WAN convencionales de conmutación de paquetes. Este capítulo explora cómo el TCP/IP, que fue diseñado para redes sin conexión, puede utilizarse en una tecnología orientada a la conexión. Veremos que el TCP/IP es muy flexible —aun cuando algunos detalles de la asignación de direcciones cambie, la mayor parte de los protocolos se mantiene sin cambios.

Para hacer el análisis más completo y relacionarlo con el hardware disponible, utilizaremos el *Asynchronous Transfer Mode* (*Modo de Transferencia Asíncrono* o *ATM*, por sus siglas en inglés) en todos los ejemplos. El ATM ofrece alta velocidad, pueden utilizarlo tanto redes de área local como redes de área amplia, soporta una variedad de aplicaciones incluyendo audio y video en tiempo actual, así como la comunicación convencional de datos. Este capítulo amplía la breve descripción presentada en el capítulo 2 y cubre detalles adicionales. En particular, las próximas secciones describen la topología física de una red ATM, la conectividad lógica proporcionada, los paradigmas de conexión de ATM y el protocolo de ATM para la transferencia de datos.

En secciones posteriores, se explica la relación entre ATM y TCP/IP. Se muestra cómo se relacionan las direcciones de anfitrión de ATM con las direcciones de anfitrión de IP. Se describe una forma modificada del Protocolo de Resolución de Direcciones (*Address Resolution* Protocol, ARP) utilizado para resolver una dirección IP hacia una conexión ATM y una forma modificada de ARP inversa, utilizada para auxiliar a los administradores en la asignación de direcciones en un servidor. Algo muy importante, veremos cómo los datagramas IP viajan a través de una red ATM sin fragmentación IP.

18.2 Hardware ATM

El componente básico de una red ATM es un conmutador electrónico de propósito especial diseñado para transferir datos a velocidades muy altas. Un conmutador pequeño común puede conectar entre 16 y 32 computadoras. Para permitir la comunicación de datos a altas velocidades, cada conexión entre una computadora y un conmutador ATM utiliza un par de fibras ópticas.[1] La figura 18.1 ilustra la conexión entre una computadora y un conmutador ATM.

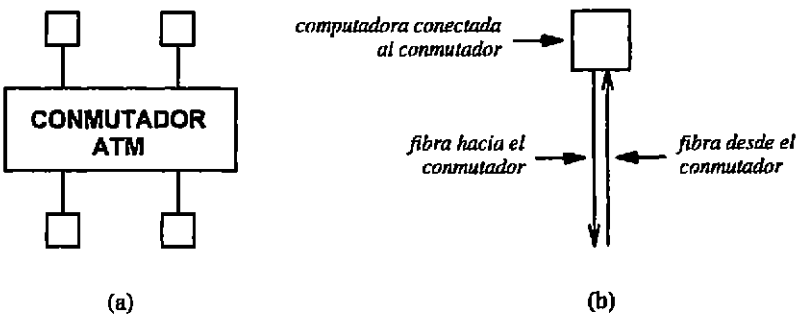


Figura 18.1 (a) Diagrama esquemático de un solo conmutador ATM con cuatro computadoras conectadas y (b) detalle de una sola conexión. Un par de fibras ópticas transporta datos hacia y desde el conmutador.

Físicamente, se conecta una tarjeta de interfaz de anfitrión dentro del bus de la computadora. El hardware de interfaz incluye un diodo emisor de luz (LED) o un láser en miniatura junto con la circuitería necesaria para convertir los datos en pulsos de luz que viajan hacia la fibra y hacia el conmutador. La interfaz también contiene el hardware necesario para percibir los pulsos de luz que vienen desde el conmutador y convertirlos de nuevo en bits de datos en forma electrónica. Como una fibra dada puede transportar luz sólo en una dirección, la conexión requiere de un par de fibras para permitir a la computadora tanto el envío como la recepción de datos.

18.3 Redes ATM grandes

Aun cuando un solo conmutador ATM tiene una capacidad finita, se pueden conectar varios conmutadores para formar una red extensa. En particular, para conectar computadoras en dos localidades a la misma red, en cada localidad puede instalarse un interruptor y pueden conectarse los dos conmutadores. La conexión entre dos conmutadores difiere ligeramente de la conexión entre una computadora anfitrión y un conmutador. Por ejemplo, la conexión entre conmutadores puede operar a velocidades altas y utilizar protocolos ligeramente modificados. La figura 18.2 ilustra la topología y muestra la diferencia entre una *Network to Network Interface* (*Interfaz de Red a Red* o *NNI*,

[1] En la mayor parte de las instalaciones se usa el tipo de fibras multimodo.

por sus siglas en inglés) y una *User to Network Interface* (*Interfaz de Usuario a Red* o *UNI*, por sus siglas en inglés).

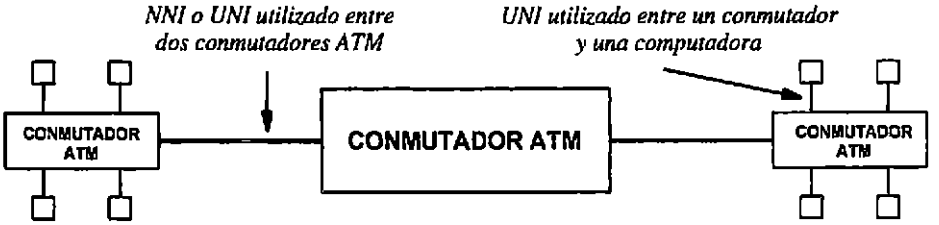


Figura 18.2 Tres conmutadores ATM combinados para formar una red extensa. Aun cuando una interfaz NNI está diseñada para utilizarse entre conmutadores, las conexiones UNI pueden utilizarse entre conmutadores ATM en una red privada.

La distinción entre UNI y NNI se debe a que las compañías de teléfonos que diseñaron la tecnología ATM utilizaron el mismo paradigma que para las redes de voz. En general, en una compañía de teléfonos que ofrece servicios de datos ATM para clientes, éstos también serán conectados con otras compañías telefónicas. Los diseñadores concibieron UNI como la interfaz entre el equipo en una localidad de cliente y el conmutador del propio equipo para el transporte común y NNI como la interfaz entre conmutadores propios y los operados por dos compañías telefónicas diferentes.

18.4 El aspecto lógico de una red ATM

Para una computadora conectada con una red ATM, una instalación completa de conmutadores ATM parece ser una red homogénea. Como en el sistema telefónico de voz o en una red Ethernet puenteada, ATM oculta los detalles del hardware físico y conserva la apariencia de una sola red física con muchas computadoras conectadas. Por ejemplo, la figura 18.3 ilustra cómo el sistema de conmutación ATM, en la figura 18.2, une lógicamente a otras computadoras que están conectadas con ésta.

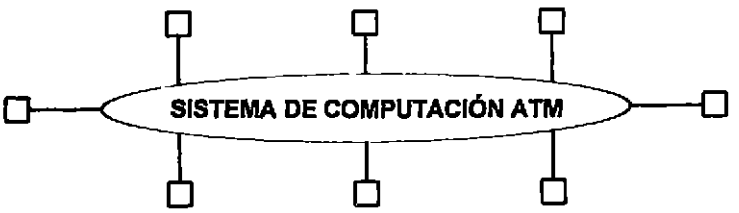


Figura 18.3 Esquema lógico del conmutador ATM de la figura 18.2. ATM tiene la apariencia de una red uniforme; cualquier computadora puede comunicarse con cualquier otra.

Así, ATM proporciona la misma abstracción general a través del hardware ATM homogéneo que proporciona el TCP/IP para sistemas heterogéneos:

> *A pesar de una arquitectura física que permite a una instalación de conmutadores contener varios conmutadores, el hardware ATM proporciona a las computadoras conectadas la apariencia de una sola red física. Cualquier computadora en una red ATM puede comunicarse de manera directa con cualquier otra; las computadoras se mantienen ignorantes de la estructura de red física.*

18.5. Los dos paradigmas de la conexión ATM

ATM proporciona una interfaz orientada a la conexión para conectar anfitriones. Para alcanzar un destino remoto en una red ATM, un anfitrión establece una *conexión*, una abstracción que recuerda a una llamada telefónica. ATM ofrece dos formas de conexión. La primera se conoce como *Switched Virtual Circuit* (*Circuito Virtual Conmutado* o *SVC*) y la segunda como *Permanent Virtual Circuit* (*Circuito Virtual Permanente* o *PVC*).

18.5.1. Circuitos virtuales conmutados

Un circuito virtual conmutado opera como una llamada telefónica de voz convencional. Un anfitrión se comunica con su conmutador ATM para solicitar que el conmutador establezca un SVC. El anfitrión especifica la dirección completa de una computadora anfitrión remota y la calidad del servicio solicitado. Entonces, el anfitrión espera una señal de la red ATM para crear un circuito. El sistema de *señalización*[2] ATM se establece y define una trayectoria desde el anfitrión que originó la llamada, a través de la red ATM (posiblemente a través de varios conmutadores), hacia la computadora anfitrión remota. La computadora remota debe acordar la aceptación del circuito virtual.

Durante la señalización, cada conmutador ATM, a lo largo de la trayectoria, examina la calidad de servicio solicitado para el circuito. Si se acuerda enviar los datos, un conmutador graba información sobre el circuito y envía la solicitud hacia el próximo conmutador en la trayectoria. Cada acuerdo requiere un compromiso de los recursos de hardware y software en cada conmutador. Cuando la señalización se completa, el conmutador ATM local reporta el éxito de la operación hacia ambos extremos del circuito virtual conmutado.

La interfaz UNI de ATM se vale de un entero de 24 bits para identificar cada circuito virtual. Cuando un anfitrión crea o acepta un circuito virtual nuevo, el conmutador ATM local asigna un identificador para el circuito. Un paquete transmitido a través de una red ATM no contiene direcciones de fuente ni de destino. De hecho, un anfitrión etiqueta cada paquete que sale y el conmutador etiqueta cada paquete entrante con un indicador de circuito.

Nótese que hemos omitido varios detalles de la señalización, incluyendo el protocolo que emplea un anfitrión para solicitar un nuevo circuito y el protocolo que utiliza un conmutador para informar al anfitrión que ha llegado una solicitud de conexión desde un anfitrión remoto. Además,

[2] El término *señalización* se deriva de la jerga telefónica; la señalización ya no es parte del estándar ATM.

hemos omitido unos cuantos detalles que son importantes en la práctica. Por ejemplo, la comunicación de dos vías requiere que se reserven recursos a lo largo de la trayectoria inversa así como en la trayectoria de envío.

18.5.2 Circuitos virtuales permanentes

La alternativa para un circuito virtual conmutado es mundana: un administrador interactúa con los conmutadores en una red ATM para configurar los circuitos virtuales a mano. El administrador especifica la fuente y el destino del circuito, la calidad de servicio que el circuito recibirá y los identificadores de 24 bits que cada anfitrión utilizará para accesar el circuito. Aun cuando los circuitos virtuales conmutados proporcionan accesibilidad, los circuitos virtuales permanentes son importantes por tres razones. En primer lugar, hasta que todos los vendedores acuerden un mecanismo de señalización estándar, los conmutadores que provengan de los vendedores deberán valerse de PVC para operar entre sí. En segundo lugar, PVC puede emplearse en líneas arrendadas. En tercer lugar, PVC puede utilizarse en redes para mantenimiento y depuración .

18.6 Rutas, circuitos e identificadores

ATM asigna un identificador entero único para cada circuito cuando un anfitrión ha sido abierto; el anfitrión utiliza el identificador cuando realiza operaciones de entrada y salida o cuando cierra el circuito. Un identificador de circuito es análogo a un descriptor que utiliza un programa para realizar operaciones de entrada y salida. Como un descriptor de entrada/salida, un identificador de circuito es corto comparado con la información necesaria para crear un circuito. También, como un descriptor de entrada/salida, un identificador de circuito se mantiene válido mientras el circuito está abierto. Además, un identificador de circuito es significativo sólo a través de un solo salto —los identificadores de circuito obtenidos por los anfitriones en los dos extremos de un circuito virtual dado por lo común difieren. Por ejemplo, el emisor puede utilizar el identificador 17 mientras que el receptor usa el identificador *49*; cada conmutador ATM transfiere el identificador de circuito en un paquete como el siguiente paquete desde un anfitrión hacia el otro.

Técnicamente, un identificador de circuito utilizado con la interfaz UNI consiste en un entero de 24 bits dividido en dos campos.[3] La figura 18.4 muestra cómo divide ATM los 24 bits en un *identificador de ruta virtual* (*VPI, virtual pathe identificar*) de 8 bits y un *identificador de circuito virtual (VCI, virtual circuit identifier*) de 16 bits. Casi siempre, el identificador completo se conoce como *par VPI/VCI*

La motivación para dividir un identificador de conexión en los campos VPI y VCI es similar a la razón para dividir una dirección IP en campos de red y anfitrión. Si un conjunto de circuitos virtuales sigue la misma trayectoria, un administrador puede arreglar todos los circuitos en el conjunto para utilizar el mismo VPI. El hardware ATM puede entonces usar el VPI para rutear el tráfico de manera eficiente. Las compañías de comunicación comercial pueden también utilizar VPI para su contabilidad —una compañía de comunicación puede establecer un cargo a un cliente por una ruta virtual lo que permite al cliente decidir cómo multiplexar circuitos virtuales múltiples dentro de una ruta.

[3] El identificador de circuito utilizado por NNI tiene un formato ligeramente diferente y una longitud distinta.

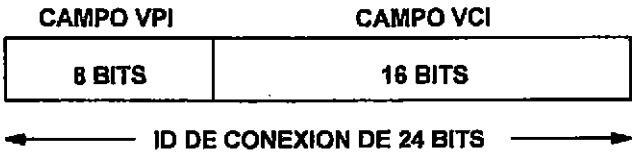


Figura 18.4 Identificador de conexión, formado por 24 bits, utilizado con UNI. El identificador se divide en partes de ruta virtual y circuito virtual.

18.7 Transporte de celdas ATM

En el nivel inferior, una red ATM utiliza tramas de tamaño fijo, llamadas *celdas*, para transportar datos. ATM requiere que todas las celdas sean del mismo tamaño porque, al hacerlo así, al hardware de conmutación le es posible trabajar más rápido. Cada celda ATM tiene una longitud de 53 octetos y consiste en un encabezado de 5 octetos, seguido por 48 octetos de datos. La figura 18.5 muestra el formato del encabezado de una celda.

0 1 2 3	4 5 6 7
CONTROL DE FLUJO	VPI (PRIMEROS 4 BITS)
VPI (ÚLTIMOS 4 BITS)	VCI (PRIMEROS 4 BITS)
VCI (8 BITS MEDIOS)	
VCI (ÚLTIMOS 4 BITS)	TIPO PAYLOAD PRIO
VERIFICACIÓN POR REDUNDANCIA CÍCLICA	

Figura 18.5 Contenido de los cinco octetos que comprenden la forma UNI de un encabezado de celda ATM, en el que se muestra un octeto en cada línea. Los datos en la celda siguen inmediatamente del encabezado.

18.8 Capas de adaptación ATM

Aun cuando ATM conmuta celdas pequeñas en el nivel inferior, los programas de aplicación que transfieren datos en ATM no leen o escriben celdas. Por ejemplo, una computadora interactúa con

ATM a través de una *capa de adaptación ATM*, la cual es parte del estándar ATM. La capa de adaptación realiza varias funciones, incluyendo la detección y la corrección de errores, como los provocados por celdas perdidas o alteradas. Usualmente, los microprogramas que implantan una capa de adaptación ATM están localizados en una interfaz de anfitrión, junto con el hardware y los microprogramas que proporcionan la transmisión y recepción de celdas. La figura 18.6 ilustra la organización de una interfaz ATM común y muestra cómo los datos pasan del sistema operativo de una computadora, a través de la tarjeta de interfaz, hasta a la red ATM.

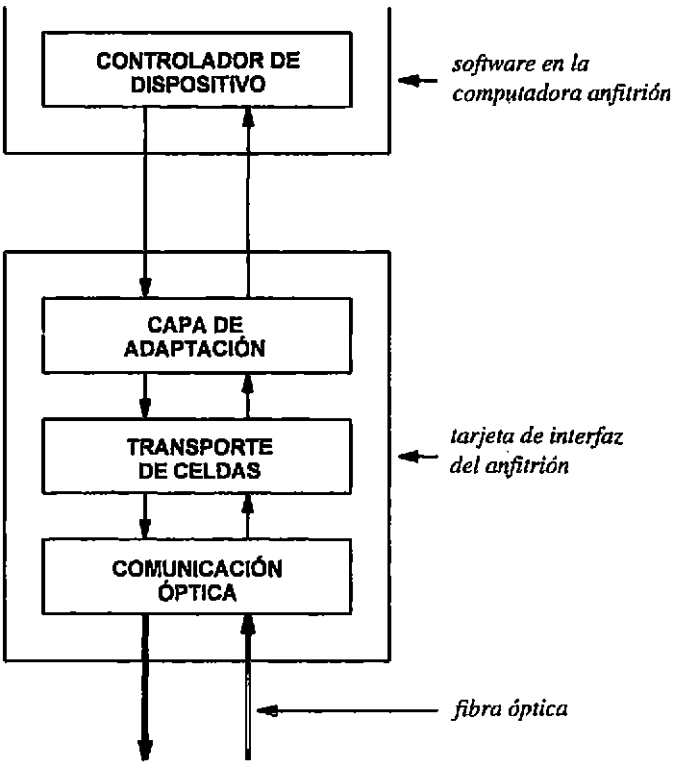


Figura 18.6 Organización conceptual del hardware de interfaz ATM y flujo de datos a través de éste. Una computadora anfitrión interactúa con un protocolo de capa de adaptación para enviar y recibir datos. La capa de adaptación convierte los datos en celdas que salen y extrae los datos de las celdas entrantes. La capa de transporte de celdas transfiere celdas hacia el conmutador ATM.

Cuando se establece una conexión, un anfitrión debe especificar qué capa de adaptación de protocolo utilizar. Ambos extremos de la conexión tienen que estar de acuerdo en la selección, y la capa de adaptación no puede cambiarse una vez que la conexión se ha establecido. Para resumir:

Aun cuando el hardware ATM utiliza celdas pequeñas de tamaño fijo para transportar datos, una capa superior de protocolo, llamada capa de adaptación *ATM, proporciona el servicio de transferencia de datos para computadoras que emplean ATM. Cuando se crea un circuito virtual, ambos extremos del circuito deben acordar el tipo de protocolo de adaptación que será utilizado.*

18.8.1 Capa 1 de adaptación ATM

Sólo dos protocolos de capa de adaptación ATM interesantes han sido definidos: uno para enviar audio ó video y otro para enviar paquetes de datos convencionales. La *capa 1 de adaptación ATM (AAL1)* acepta y envía datos a través de una red ATM en una cantidad de bits fija. Una conexión creada para enviar video utiliza AAL1 debido a que el servicio de cantidad fija es necesario para garantizar que la transmisión de video no ocasione que la imagen sea inestable o se interrumpa.

18.8.2 Capa 5 de adaptación ATM

Las computadoras utilizan la *capa 5 de adaptación ATM (AAL5)*[4] para enviar paquetes de datos convencionales a través de una red ATM. Aun cuando ATM utiliza celdas pequeñas de tamaño fijo en el nivel más bajo, AAL5 presenta una interfaz que acepta y entrega paquetes largos y de longitud variable. En particular, AAL5 permite que cada paquete contenga entre 1 y 65,535 octetos de datos. La figura 18.7 ilustra el formato del paquete que utiliza AAL5.

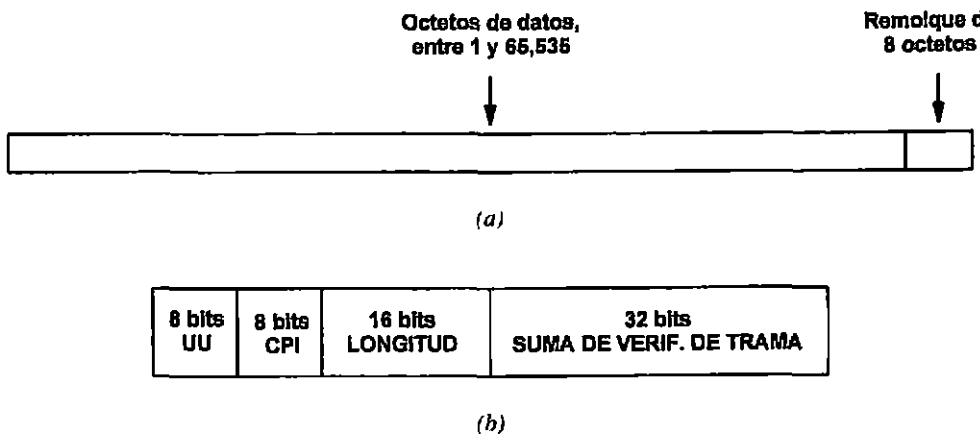


Figura 18.7 (a) Formato del paquete básico que AAL5 acepta y entrega, y (b) campo en el remolque de 8 octetos colocado después de los datos.

[4] Originalmente AAL3 y AAL4 se definieron para la transmisión de datos. Se combinaron en AAL3/4 y les siguió AAL5.

A diferencia de la mayor parte de las tramas de red que colocan la información de control en un encabezado, AAL5 la coloca en un registro a remolque en el extremo del paquete. El remolque de AAL5 contiene un campo de longitud equivalente a 16 bits, un verificador por redundancia cíclica de 32 bits (CRC), utilizado como una suma de verificación de trama, y 2 campos de 8 bits llamados *UU* y *CPI* que actualmente no tienen uso.[5]

Cada paquete AAL5 debe dividirse en celdas para transportarse a través de una red ATM y debe recombinarse para formar un paquete antes de que sea entregado al anfitrión receptor. Si el paquete, incluyendo el remolque de 8 octetos, es un múltiplo exacto de 48 octetos, la división producirá celdas completamente llenas. Si el paquete no es un múltiplo exacto de 48 octetos, la celda final no estará llena. Para adaptarse a la longitud indeterminada de los paquetes, AAL5 permite que la celda final contenga entre 0 y 40 octetos de datos, seguidos por un relleno de ceros, y por un remolque de 8 octetos. En otras palabras, AAL5 coloca el remolque en los últimos 8 octetos del final de la celda, donde se pueden encontrar y extraer sin conocer la longitud del paquete.

18.9 Convergencia, segmentación y reensamblaje de AAL5

Cuando una aplicación envía datos sobre una conexión ATM por medio de AAL5, el anfitrión entrega un bloque de datos a la interfaz AAL5. AAL5 genera un remolque, divide la información en bloques de 48 octetos y transfiere cada bloque a través de la red ATM en una sola celda. En el extremo de recepción de la conexión, AAL5 reensambla las celdas entrantes en paquete, verifica el CRC para asegurarse de que el paquete llegó correctamente y transfiere el resultado al software del anfitrión. El proceso de dividir el paquete en celdas y reagruparlo se conoce como *segmentation and reassembly* (*segmentación y reensamblado* o *SAR*) ATM.[6]

¿Cómo sabe AAL5, en el lado del receptor, cuántas celdas comprende un paquete? El emisor AAL5 utiliza el bit de orden inferior del campo *type payload* del encabezado de la celda ATM para marcar el final de la celda en un paquete. Pensemos en un *bit de final de paquete*. Así, el receptor AAL5 reúne celdas entrantes hasta que encuentra una con el bit de fin de paquete activado. El estándar ATM utiliza el término *convergencia* para describir el mecanismo que reconoce el fin de un paquete. Aun cuando AAL5 emplea un solo bit en el encabezado de celda para la convergencia, otros protocolos de capa de adaptación ATM utilizan otros mecanismos de convergencia.

En resumen:

> *Una computadora utiliza la capa 5 de adaptación ATM para transferir un bloque extenso de datos en un circuito virtual ATM. En el anfitrión emisor, AAL5 genera un remolque, divide el bloque de datos en celdas y envía cada celda por un circuito virtual. En el anfitrión receptor, AAL5 reensambla las celdas a fin de reproducir el bloque original de datos, retira el remolque y entrega los datos al anfitrión de recepción. AAL5 utiliza un bit en el encabezado de la celda para marcar la celda final de un bloque de datos dado.*

---

[5] El campo *UU* puede contener cualquier valor; el campo *CPI* debe estar puesto en cero.

[6] El uso del término *reensamblado* sugiere una fuerte similitud entre la segmentación AAL5 y la fragmentación IP: ambos mecanismos dividen un bloque extenso de datos en unidades pequeñas para su transferencia.

18.10 Encapsulación de datagramas y tamaño de MTU de IP

Debería ser fácil entender como AAL5 puede usarse para encapsular un datagrama IP y, así, transferirlo a través de una red ATM. En la forma más sencilla, un emisor establece un circuito virtual permanente o conmutado a través de la red ATM hacia una computadora destino y especifica que el circuito utiliza AAL5. Entonces el emisor puede pasar un datagrama IP completo hacia AAL5 para entregarlo a través del circuito. AAL5 genera un remolque, divide el datagrama en celdas y transfiere las celdas a través de la red. En el lado del receptor, AAL5 reensambla el datagrama, utiliza la información en el remolque para verificar que los bits no hayan sido alterados o se hayan perdido y transfiere el resultado hacia el IP.

Dijimos que AAL5 utiliza un campo con una longitud de 16 bits, lo que hace posible enviar 64K de octetos en un solo paquete. A pesar de la capacidad de AAL5, el TCP/IP restringe el tamaño de los datagramas que pueden enviarse en una red ATM. El estándar impone un límite de 9,180 octetos[7] por datagrama. Esto es, el IP impone una MTU de 9,180 en las redes ATM. Como en el caso de cualquier interfaz de red, cuando un datagrama que sale hacia el interior es mayor que el MTU de la red, el IP fragmenta el datagrama y transfiere cada fragmento hacia a AAL5. Así, AAL5 acepta, transfiere y entrega datagramas de 9,180 octetos o menos. En resumen:

> *Cuando el TCP/IP envía datos a través de una red ATM, transfiere un datagrama entero utilizando la capa 5 de adaptación ATM. Aun cuando AAL5 puede aceptar y transferir paquetes que contengan más de 64K octetos, el estándar TCP/IP restringe la MTU efectiva a 9,180 octetos. El IP debe fragmentar cualquier datagrama superior a 9,180 octetos antes de transferirlos a AAL5.*

18.11 Tipos y multiplexión de paquetes

Los lectores observadores habrán notado que el remolque AAL5 no incluye un campo de tipo. Así, una trama AAL5 no es autoidentificable. Como resultado, la forma más sencilla de encapsulación, descrita arriba, no siempre es suficiente. De hecho, existen dos posibilidades:

- Las dos computadoras en los dos extremos del circuito virtual acuerdan *a priori* que el circuito debe utilizarse para un protocolo específico (esto es, el circuito no será empleado sólo para enviar datagramas IP).
- Las dos computadoras en los extremos del circuito virtual acuerdan *a priori* que algunos octetos del área de datos serán reservados para utilizarse como un campo tipo.

El esquema anterior, en el que las computadoras acuerdan un protocolo de alto nivel para un circuito dado, tiene la ventaja de no necesitar información adicional en un paquete. Por ejemplo, si las computadoras establecen un acuerdo para transferir el IP, un emisor puede transferir cada

[7] El tamaño de 9180 fue seleccionado para hacer compatible ATM con una tecnología anterior llamada *Switched Multimegabit Data Service (SMDS)*.

datagrama de manera directa hacia AAL5 para su transferencia; no se necesita enviar nada fuera del datagrama y el remolque AAL5. La mayor desventaja de este esquema radica en la duplicación de circuitos virtuales: una computadora debe crear un circuito virtual separado para cada protocolo de alto nivel. (Como una compañía de comunicaciones puede generar cargos por cada circuito virtual, la creación de varios circuitos virtuales entre un par de computadoras puede añadir costos innecesarios.)

En el último esquema, en que dos computadoras utilizan un circuito virtual para varios protocolos se tiene la ventaja de permitir que todo tráfico viaje sobre el mismo circuito, pero la desventaja es que requiere que cada paquete contenga octetos que identifiquen el tipo de protocolo. El esquema también tiene la desventaja de que los paquetes de todos los protocolos viajan con el mismo retraso y la misma prioridad.

El estándar TCP/IP especifica que las computadoras pueden seleccionar entre los dos métodos de uso de AAL5. Tanto el emisor como el receptor deben acordar cómo se utilizará el circuito; el acuerdo debe comprender configuraciones manuales. Además, el estándar sugiere que, cuando las computadoras seleccionan incluir el tipo de información en el paquete, éstas deben utilizar el encabezado estándar IEEE 802.2 *Logical Link Control* (*LLC*) seguido por un encabezado *SubNetwork Attachment Point* (*SNAP*). La figura 18.8 ilustra la información LLC/SNAP prefijada para un datagrama antes de ser enviado hacia un circuito virtual ATM.

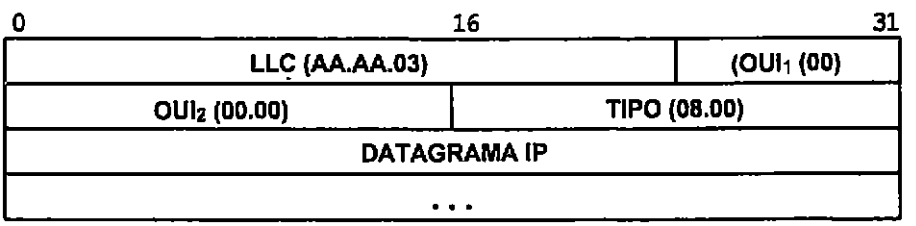


Figura 18.8 Formato del paquete utilizado para enviar un datagrama IP en un AAL5 cuando se multiplexan varios circuitos en un solo circuito virtual. El encabezado LLC/SNAP de 8 octetos identifica el contenido como un datagrama IP.

Como se muestra en la figura, el campo LLC consiste en 3 octetos que contienen el valor hexadecimal *AA.AA.03*.[8] El encabezado SNAP consiste en 5 octetos: 3 que contienen un *Organizationally Unique Identifier (OUI)*, y 2 para el tipo.[9] El campo *OUI* identifica una organización que administra los valores en el campo *TYPE*, y el campo *TYPE* identifica el tipo de paquete. Para un datagrama IP, el campo *OUI* contiene *00.00.00* que identifica a la organización responsable de los estándares Ethernet, y el campo *TYPE contiene 08.00*, el valor utilizado cuando se encapsula el IP en una trama Ethernet. El software en el anfitrión emisor debe prefijar el encabezado LLC/SNAP para cada paquete antes de enviarlo hacia AAL5, y el software en el anfitrión de recepción debe hacerlo con el encabezado para determinar cómo manejar el paquete.

[8] La notación representa cada octeto como un valor hexadecimal separado por puntos.

[9] Para acomodar octetos adicionales de encabezado, la MTU, de una conexión ATM que utiliza un encabezado LLC/SNAP, tiene un valor de 9188.

18.12 Enlace de direcciones IP en una red ATM

Hemos visto que la encapsulación de datagramas para su transmisión a través de una red ATM se deduce de manera directa. En contraste, la asignación de direcciones IP puede ser difícil. Como en otras tecnologías de red, ATM asigna a cada computadora conectada una dirección física que puede emplearse cuando se establece un circuito virtual. Por un lado, como las direcciones físicas de ATM son más grandes que las direcciones IP, una dirección física ATM no puede codificarse dentro de una dirección IP. Así, el IP no puede utilizar la asignación de direcciones estáticas para redes ATM. Por otro lado, el hardware ATM no soporta la difusión. Por lo tanto, el IP no puede utilizar la ARP convencional para asignar direcciones en redes ATM.

Los circuitos virtuales permanentes ATM complican aún más la asignación de direcciones. Debido a que un administrador configura manualmente cada circuito virtual permanente, un anfitrión sólo conoce el par de circuitos VPI/VCI. El software en el anfitrión no conoce la dirección IP ni la dirección de hardward ATM del extremo remoto. Un mecanismo de asignación de direcciones IP debe proporcionar la identificación de una computadora remota conectada a un PVC así como la creación dinámica de SVC para destinos conocidos.

Las tecnologías de conmutación orientadas a la conexión complican aún más la asignación de direcciones porque requieren dos niveles de asignación. En primer lugar, cuando crean un circuito virtual sobre el que serán enviados los datagramas, las direcciones IP de los destinos deben transformarse en direcciones de puntos extremos ATM. Las direcciones de puntos extremos se usan para crear un circuito virtual. En segundo lugar, cuando se envía un datagrama a una computadora remota en un circuito virtual existente, las direcciones IP de dos destinos se deben transformar en el par VPI/VCI para el circuito. El segundo direccionamiento se utiliza cada vez que un datagrama es enviado en una red ATM; el primer direccionamiento es necesario sólo cuando un anfitrión crea un SVC.

18.13 Concepto lógico de subred IP

Aunque ningún protocolo ha sido propuesto para resolver el caso general de la asignación de direcciones para redes ATM extensas, un protocolo se vislumbra como una forma restringida. La restricción de la forma radica en que un grupo de computadoras utilizan una red ATM en lugar de una red física única (a menudo, local). El grupo forma una *Logical IP Subnet* (*Subred IP Lógica* o *LIS*). Varias subredes lógicas IP pueden definirse entre un conjunto de computadoras conectadas al mismo hardware de red ATM. Por ejemplo, la figura 18.9 ilustra 8 computadoras conectadas a una red ATM divididas en dos LIS.

Como se muestra en la figura, todas las computadoras están conectadas a la misma red física ATM. Las computadoras *A*, *C*, *D*, *E* y *F* participan en una LIS, mientras que las computadoras *B*, *F*, *G* y *H* lo hacen en otra. Cada subred IP lógica funciona como una LAN separada. Las computadoras participan en un LIS estableciendo circuitos virtuales entre ellas para intercambiar datagramas.[10] Dado que cada LIS forma una red separada conceptualmente, el IP aplica la regla estándar para una red física a cada LIS. Por ejemplo, todas las computadoras en una LIS comparten un solo prefijo de

[10] Los circuitos que forman el enlace de una LIS deben usar encapsulación LLC/SNAP.

red IP y este prefijo difiere de los prefijos utilizados por otras subredes lógicas. Además, aun cuando las computadoras en una LIS pueden seleccionar una MTU estándar, todas las computadoras deben utilizar la misma MTU en todos los circuitos virtuales que comprenden a la LIS. Por último, aun con el hardware de ATM que proporciona la conectividad potencial, un anfitrión en una LIS no debe comunicarse de manera directa con un anfitrión en otra LIS. De hecho, todas las comunicaciones lógicas entre sus redes deben proceder a través de un ruteador que participe en varias subredes lógicas. En la figura 18.9, por ejemplo, la máquina *F* puede ser un ruteador IP entre las dos subredes lógicas dado que participa en ambas.

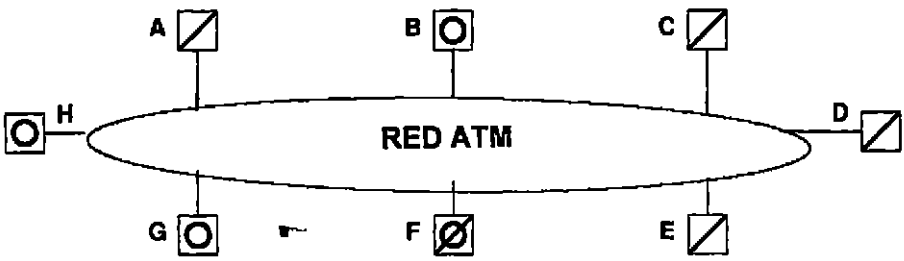


Figura 18.9 Ocho computadoras conectadas a una red ATM que participan en dos subredes IP lógicas. Las computadoras marcadas con una diagonal participan en una LIS y las computadoras marcadas con un círculo en otra LIS.

En suma:

El TCP/IP permite a un subconjunto de computadoras conectadas a una red ATM operar como una LAN independiente. Cada grupo se conoce como subred IP lógica *(LIS); las computadoras en una LIS comparten una sola dirección de red IP. Una computadora en una LIS puede comunicarse de manera directa con cualquier otra computadora en la misma LIS, pero se requiere de un ruteador cuando se comunica con una computadora en otra LIS.*

18.14 Gestión de conexiones

Los anfitriones deben manejar cuidadosamente los circuitos virtuales ATM porque la creación de un circuito toma tiempo y, para los servicios ATM comerciales, se puede incurrir en costos económicos adicionales. Así, el enfoque simplista de la creación de un circuito virtual, en el que se envía un datagrama y luego se cierra un circuito, es demasiado caro. En realidad, un anfitrión debe mantener un registro de circuitos abiertos conforme éstos son utilizados.

La administración de circuitos se da en el software de interfaz de red más allá del IP. Cuando un anfitrión necesita enviar un datagrama, se vale del ruteo IP convencional para encontrar la dirección

del próximo salto apropiado, *N*,[11] y lo pasa junto con el datagrama hacia la interfaz de red. La interfaz de red examina su tabla de circuitos virtuales abiertos. Si existe un circuito abierto para *N*, el anfitrión emplea AAL5 para enviar el datagrama. De otra manera, antes de que el anfitrión pueda enviar el datagrama, deberá localizar una computadora con dirección *N*, crear un circuito y añadir el circuito a su tabla.

El concepto de subredes IP lógicas restringe el ruteo IP. En una tabla de ruteo configurada adecuadamente, la dirección del próximo salto para cada destino debe ser una computadora con la misma subred lógica que el emisor. Para entender esta restricción, recordemos que cada LIS es designada para operar como una sola LAN. La misma restricción se mantiene para los anfitriones conectados a una LAN, a saber, cada dirección del próximo salto en la tabla de ruteo debe ser un ruteador conectado con la LAN.

Una de las razones para dividir las computadoras en subredes lógicas proviene de restricciones en el hardware y en el software. Un anfitrión no puede mantener arbitrariamente un número extenso de circuitos virtuales abiertos al mismo tiempo, ya que cada circuito requiere recursos en el hardware ATM y en el sistema operativo. Al dividir las computadoras en subredes lógicas, se limita el número máximo de circuitos abiertos simultáneamente al número de computadoras en la LIS.

18.15 Enlace de direcciones dentro de una LIS

Cuando un anfitrión crea un circuito virtual para una computadora en su LIS, el anfitrión debe justificar una dirección de hardware ATM para el destino. ¿Cómo puede un anfitrión transformar una dirección del próximo salto en una dirección de hardware ATM apropiada? El anfitrión no puede difundir una solicitud a todas las computadoras en la LIS porque ATM no ofrece hardware de difusión, sino que se contacta a un servidor para obtener la transformación. La comunicación entre el anfitrión y el servidor utiliza *ATMARP*, una variante del protocolo ARP descrito en el capítulo 5.

Como con una ARP convencional, un emisor forma una solicitud que incluye las direcciones de hardware y los emisores IP y ATM, así como la dirección IP de un destino para el que es necesaria una dirección de hardware ATM. El emisor transmite entonces la solicitud hacia el *servidor ATMARP* para la subred lógica. Si el servidor conoce la dirección de hardware ATM, envía una *réplica ATMARP*. De otra forma, el servidor envía una *réplica ATMARP negativa*.

18.16 Formato de los paquetes ATMARP

La figura 18.10 ilustra el formato de un paquete ATMARP. Como lo muestra la figura, ATMARP modifica ligeramente el formato del paquete ARP. El mayor cambio comprende campos de longitud y dirección adicional para adaptarse a las direcciones ATM. Para entender el cambio, se debe entender que han sido propuestas varias formas de direcciones para ATM, y que no aparece una sola forma que se defina como estándar. Las compañías telefónicas que ofrecen redes públicas ATM se valen de un formato de 8 octetos donde cada dirección es un número telefónico ISDN

[11] Como es usual una dirección del próximo salto es una dirección IP.

definido por el estándar ITU-TS en el documento *E.164*. En contraste, el ATM Forum[12] permite que cada computadora conectada con una red ATM privada sea asignada a 20 octetos una dirección *Network Service Access Point* (*NSAP*). Así, se necesita una dirección jerarquizada en dos niveles para especificar una dirección E.164 para una localidad remota y una dirección NSAP de un anfitrión en un conmutador local en la localidad.

Para adaptarse a varios formatos de dirección y a una jerarquía de dos niveles, un paquete ATMARP contiene dos campos de longitud para cada dirección ATM así como un campo de longitud para cada dirección de protocolo. Como se muestra en la figura 18.10, un paquete ATMARP comienza con campos de tamaños fijos que especifican longitudes de dirección. El primero de los dos campos sigue el mismo formato que una ARP convencional. El campo con el nombre *HARDWARE TYPE (TIPO DE HARDWARE)* contiene el valor hexadecimal *0x0013* para ATM, y el campo con el nombre *PROTOCOL TYPE* (*TIPO DE PROTOCOLO*) contiene el valor hexadecimal *0x0800* para IP.

Como el formato de las direcciones del emisor y el destino pueden diferir, cada dirección ATM requiere un campo de longitud. El campo *SEND HLEN* especifica la longitud de la dirección ATM del emisor y el campo *SEND HLEN2* especifica la longitud de la subdirección ATM del emisor. Los campos *TAR LEN* y *TAR LEN2* especifican la longitud de la dirección ATM del destino y de su subdirección. Por último, los campos *SEND PLEN* y *TAR PLEN* especifican la longitud de las direcciones de protocolo del emisor y el receptor.

0	8	16	24 31
TIPO DE HARDWARE (0x0013)		TIPO DE PROTOCOLO (0x0800)	
SEND. HLEN (20)	SEND. HLEN2 (0)	OPERACIÓN	
SEND. PLEN (4)	TAR.HLEN (20)	TAR.HLEN2 (0)	TAR. PLEN (4)
DIRECCIÓN ATM DEL EMISOR (octetos 0-3)			
DIRECCIÓN ATM DEL EMISOR (octetos 4-7)			
DIRECCIÓN ATM DEL EMISOR (octetos 8-11)			
DIRECCIÓN ATM DEL EMISOR (octetos 12-15)			
DIRECCIÓN ATM DEL EMISOR (octetos 16-19)			
DIRECCIÓN DEL PROTOCOLO DEL EMISOR			
DIRECCIÓN ATM DEL DESTINO (octetos 0-3)			
DIRECCIÓN ATM DEL DESTINO (octetos 4-7)			
DIRECCIÓN ATM DEL DESTINO (octetos 8-11)			
DIRECCIÓN ATM DEL DESTINO (octetos 12-15)			
DIRECCIÓN ATM DEL DESTINO (octetos 16-19)			
DIRECCIÓN DEL PROTOCOLO DEL DESTINO			

Figura 18.10 Formato de un paquete ATMARP en el que se utilizan 20 octetos para las direcciones ATM, como lo recomienda el ATM Forum.

[12] El Forum ATM es un consorcio de miembros industriales, incluidos usuarios y fabricantes, que han acordado un estándar para las redes ATM privadas.

Aparte de los campos de longitud en el encabezado, un paquete ATMARP contiene seis direcciones. Los primeros tres campos de dirección contienen la dirección ATM del emisor, la subdirección ATM y la dirección del protocolo. Los tres últimos campos contienen la dirección ATM del destino, la subdirección ATM y la dirección de protocolo. En el ejemplo de la figura 18.10, tanto los campos de longitud de subdirección del emisor como el destino contienen 0, y el paquete no contiene octetos para subdirecciones.

18.16.1 Formato de los campos de longitud de dirección ATM

Dado que ATMARP está diseñado para utilizarse con E.164 o direcciones NSAP de 20 octetos, el campo que contiene una longitud de dirección ATM incluye un bit que especifica el formato de dirección. La figura 18.11 ilustra cómo ATMARP codifica el tipo de dirección y longitud en un campo de 8 bits.

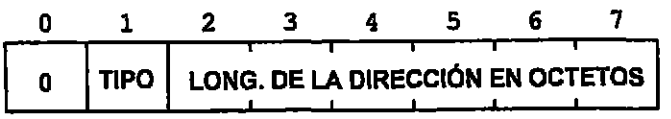


Figura 18.11 Codificación de un tipo de dirección ATM en un campo de 8 octetos. El bit *1* distingue los dos tipos de direcciones ATM.

Un solo bit codifica el tipo de dirección ATM pues sólo se dispone de dos formas posibles. Si el bit *1* contiene cero, la dirección tiene un formato NSAP recomendado por el ATM Forum. Si el bit *1* contiene el valor uno, la dirección está en el formato E.164 recomendado por la ITU-TS. Como cada campo de longitud de dirección ATM en un paquete ATMARP tiene la forma mostrada en la figura 18.11, un solo paquete puede contener varios tipos de direcciones ATM.

18.16.2 Códigos de operación utilizados con el protocolo ATMARP

El formato de paquete mostrado en la figura 18.10 se utiliza para solicitar una asignación de dirección, para solicitar la asignación de una dirección inversa. Cuando una computadora envía un paquete ATMARP, debe establecer el campo en *OPERATION* para especificar el tipo de asignación. La tabla en la figura 18.12 muestra los valores que pueden emplearse en los campos *OPERATION*

Código	Significado
1	Solicitud ATMARP
2	Réplica ATMARP
8	Solicitud ATMARP inversa
9	Réplica ATMARP inversa
10	Acuse de recibo negativo ATMARP

Figura 18.12 Valores que pueden aparecer en el campo *OPERATION* en un paquete ATMARP y su significado. Cuando es posible, los valores deben seleccionarse de acuerdo con los códigos de operación utilizados en la ARP convencional.

de un paquete ATMARP y se proporciona el significado de cada uno. En lo que resta de esta sección se explica cómo trabaja el protocolo.

18.17 Utilización de paquetes ATMARP para determinar una dirección

Realizar la asignación de direcciones para el hardward orientado a la conexión es ligeramente más complicado que para el hardware sin conexión. Dado que el hardware ATM soporta dos tipos de circuitos virtuales, se originan dos casos. En el primero, consideraremos el caso de los circuitos virtuales permanentes; en el segundo, el caso de los circuitos virtuales conmutados.

18.17.1 Circuitos virtuales permanentes

Para entender los problemas que introduce PVC, recordemos cómo opera el hardware ATM. Un administrador de red debe configurar cada PVC; los anfitriones por sí mismos no participan en la configuración de PVC. En particular, un anfitrión comienza la operación con PVC en su lugar y no recibe ninguna información desde el hardware acerca de las direcciones o de los puntos extremos remotos. Así, a menos que la información de direcciones haya sido configurada en el anfitrión (esto es, almacenada en disco), el anfitrión no tiene conocimiento de las direcciones IP o de las direcciones ATM de la computadora a la que se conecta un PVC.

El protocolo *Inverse ATMARP* (*InATMARP*) es el que resuelve el problema de encontrar una dirección cuando se emplea PVC. Para utilizar el protocolo, una computadora debe conocer cada uno de los circuitos virtuales permanentes que han sido configurados. Para determinar las direcciones IP y ATM de un punto remoto extremo, una computadora envía un paquete de solicitud Inverse ATMARP con el campo *OPERATION* puesto en *8*. Cada vez que llega una solicitud en un PVC el receptor genera una réplica Inverse ATMARP con el campo *OPERATION* puesto en *9*. Tanto la solicitud como la réplica contienen la dirección IP del emisor y la dirección ATM. Así, una computadora en cada extremo de la conexión aprende la asignación para la computadora ubicada en el otro extremo. En resumen:

> *Dos computadoras que se comunican a través de un circuito virtual permanente utilizan Inverse ATMARP para descubrir las direcciones IP y ATM de las otras. Una computadora envía una solicitud Inverse ATMARP para la que las otras envían una réplica.*

18.17.2 Circuitos virtuales conmutados

Dentro de una LIS, las computadoras crean circuitos virtuales conmutados en función de la demanda. Cuando una computadora *A* necesita enviar un datagrama a la computadora *B* y no existe en ese momento un circuito para *B*, *A* utiliza la señalización ATM para crear el circuito necesario. Así, *A* comienza con la dirección IP de *B*, la cual debe ser transformada en una dirección ATM equivalen-

te. Decimos que cada LIS tiene un servidor ATMARP y todas las computadoras en una LIS deben ser configuradas de manera que éstas tengan conocimiento acerca de como alcanzar al servidor (esto es, una computadora puede tener un PVC al servidor o la dirección ATM del servidor almacenada en disco). Un servidor no forma conexiones hacia otras computadoras; el servidor únicamente espera que las computadoras en la LIS se pongan en contacto. Para transformar direcciones *B* en direcciones ATM, la computadora *A* debe tener un circuito virtual abierto para el servidor ATMARP de la LIS. La computadora *A* forma un paquete de solicitud ATMARP y lo envía sobre la conexión hacia el servidor. El campo *OPERATION* en un paquete contiene *1*, y el campo de dirección de protocolo del destino contiene *B*.

Un servidor ATMARP mantiene una base de datos de las transformaciones de las direcciones IP en direcciones ATM. Si el servidor conoce las direcciones ATM de *B*, el protocolo ATMARP opera de manera similar a Proxy ARP. El servidor forma una réplica ATMARP al enviar el código *OPERATION* puesto en *2* y llenar la dirección ATM que corresponda a la dirección IP de destino. Como en un ARP convencional, el servidor intercambia las entradas de emisor y destino antes de regresar la réplica a la computadora que envió la solicitud.

Si el servidor no conoce la dirección ATM que corresponde a la dirección IP de destino en una solicitud, ATMARP difiere de la ARP convencional. En lugar de ignorar la solicitud, el servidor devuelve un acuse de recibo negativo (un paquete ATMARP con un campo *OPERATION* de *10*). Un acuse de recibo negativo distingue entre direcciones para las que un servidor no tiene una asignación y un servidor con falla de funcionamiento. Así, cuando un anfitrión envía una solicitud a un servidor ATMARP, determina una de tres posibilidades sin ambigüedad: la dirección ATM del destino, si el destino no está actualmente disponible en la LIS o si el servidor actualmente no está respondiendo.

18.18 Obtención de entradas para un servidor de base de datos

Un servidor ATMARP elabora y mantiene automáticamente su base de datos de asignaciones. Para hacerlo utiliza Inverse ATMARP. Cada vez que un anfitrión o un primer ruteador abren un circuito virtual hacia un servidor ATMARP, el servidor inmediatamente envía un paquete de solicitud Inverse ATMARP.[13] El anfitrión o el ruteador deben responder enviando un paquete de réplica Inverse ATMARP. Cuando reciben una réplica Inverse ATMARP, el servidor extrae las direcciones IP y ATM del emisor y almacena la asignación en su base de datos. Así, cada computadora en una LIS debe establecer una conexión hacia el servidor ATMARP, aun cuando la computadora no consulte las asignaciones.

> *Cada anfitrión o ruteador en una LIS debe registrar sus direcciones IP y sus correspondientes direcciones ATM con el servidor ATMARP para LIS. El registro se da automáticamente cada vez que una computadora establece un circuito virtual hacia un servidor ATMARP ya que el servidor envía un Inverse ATMARP al que la computadora debe responder.*

[13] El circuito debe utilizar AAL5 con una identificación de tipo LLC/SNAP.

18.19 Finalización del tiempo de la información ATMARP en un servidor

Como la asignación en una memoria provisional ARP convencional, la asignación obtenida por medio de ATMARP debe ser cronometrada y eliminada si es necesario. ¿Qué tanto debe persistir una entrada de información en un servidor? Una vez que una computadora registra sus asignaciones con un servidor ATMARP, el servidor conserva la entrada de información por un mínimo de 20 minutos. Después de este lapso, el servidor examina la entrada de información. Si no existe un circuito hacia la computadora que envió la entrada de información, el servidor borra la entrada.[14] Si la computadora que envía la entrada de información ha mantenido un circuito virtual abierto, el servidor intentará revalidar la entrada. El servidor envía una solicitud Inverse ATMARP y espera una respuesta. Si la respuesta verifica la información en la entrada, el servidor inicia un temporizador y espera otros 20 minutos. Si la respuesta Inverse ATMARP no concuerda con la información en la entrada, el servidor cierra el circuito y borra la entrada.

Para ayudar a reducir el tráfico, el estándar ATMARP permite una optimización. Permite a un anfitrión usar un solo circuito virtual para toda la comunicación con un servidor ATMARP. Cuando el anfitrión envía una solicitud ATMARP, la solicitud contiene las asignaciones del anfitrión en el campo Inverse ATMARP. El servidor puede extraer la asignación y utilizarla para revalidar su información almacenada. Así, si un anfitrión envía más de una solicitud ATMARP cada 20 minutos, el servidor no necesitará enviar al anfitrión una solicitud Inverse ATMARP.

18.20 Finalización del tiempo de información ATMARP en un anfitrión o en un ruteador

Un anfitrión o un ruteador deben también utilizar temporizadores para invalidar información obtenida desde un servidor ATMARP. En particular, el estándar especifica que una computadora puede tomar una asignación contenida de un servidor ATMARP por un máximo de 15 minutos. Cuando concluyen los 15 minutos, la entrada debe ser removida o revalidada. Si una asignación de dirección expira y el anfitrión no tiene un circuito virtual abierto para el destino, el anfitrión retirará la entrada desde su memoria intermedia ARP. Si un anfitrión tiene un circuito virtual abierto para el destino, el anfitrión intentará revalidar la asignación de direcciones. La finalización del tiempo de validez de una asignación de direcciones puede retrasar el tráfico debido a que:

> *Un anfitrión o ruteador debe dejar de enviar datos a cualquier destino para el que la asignación de direcciones ha expirado hasta que la asignación pueda revalidarse.*

El método que un anfitrión emplea para revalidar una asignación depende del tipo de circuito virtual que se esté utilizando. Si el anfitrión puede alcanzar el destino en un PVC, el anfitrión envía una solicitud Inverse ATMARP en el circuito y espera una réplica. Si el anfitrión tiene un SVC abierto hacia el destino, el anfitrión envía una solicitud ATMARP hacia el servidor ATMARP.

---

[14] El servidor no borra de manera automática una entrada cuando un circuito se cierra; por el contrario, espera durante un periodo determinado.

18.21 Resumen

ATM es una tecnología de red de alta velocidad en la que una red consiste en uno o más conmutadores interconectados para formar las instalaciones de conmutación. Lógicamente, una instalación de conmutación ATM opera como una sola y amplia red que permite a un anfitrión comunicarse con cualquier otro.

Como ATM es una tecnología orientada a la conexión, dos computadoras deben establecer el circuito virtual a través de la red antes de que puedan transferir datos. Un anfitrión puede seleccionar entre el circuito virtual de tipo permanente o conmutado. Los circuitos conmutados se crean según la demanda; los circuitos permanentes requieren de la configuración manual. En cada caso, ATM asigna a cada circuito abierto un identificador entero. Cada trama que envía un anfitrión y cada trama que entrega la red contiene un identificador de circuito; una trama no contiene una dirección de fuente o destino.

Aun cuando el nivel inferior de ATM se vale de celdas de 53 octetos para transferir información, ATM incluye un mecanismo adicional en su capa de adaptación que utiliza las aplicaciones. En particular, la capa 5 de adaptación ATM (AAL5) se utiliza para enviar datos a través de una red ATM. AAL5 ofrece una interfaz que acepta y entrega bloques de datos de tamaños variables, donde cada bloque puede ser mayor a 64K de octetos.

Para enviar un datagrama IP a través de una red ATM, el emisor debe formar una conexión de circuito virtual para el destino que utiliza AAL5, y enviar el datagrama hacia AAL5 como un solo bloque de datos. AAL5 añade un remolque, divide el datagrama y el remolque en celdas para su transmisión a través de la red, luego reensambla el datagrama antes de transferirlo hacia el sistema operativo en la computadora destino. Así, cuando se envía un datagrama a través de ATM, el IP no fragmenta el tamaño de la celda ATM. De hecho, el IP utiliza una MTU de 9,180 y permite a AAL5 segmentar el datagrama dentro de la celda.

Una subred IP lógica (LIS) consiste en un conjunto de computadoras que utilizan ATM en lugar de una LAN; las computadoras forman un circuito virtual entre ellas por medio del cual intercambian datagramas. Tener tanto los circuitos virtuales permanentes como los conmutados en una LIS complica el problema de la asignación de direcciones. Un protocolo ARP modificado y conocido como ATMARP maneja la asignación de direcciones para las computadoras en una LIS conectada por un circuito virtual conmutado. Las computadoras en una LIS dependen de un servidor ATMARP para asignar las direcciones IP de otras computadoras en la LIS con una dirección ATM equivalente. Cada computadora en la LIS debe registrarse con el servidor a fin de proporcionar sus direcciones IP y las direcciones ATM al servidor. Conforme es necesaria, otras computadoras pueden entonces contactar el servidor para obtener una asignación. Como en la ARP convencional, una asignación obtenida desde ATMARP tiene un periodo de vida válido, luego del cual, la asignación se debe revalidar o descartar. Hay un protocolo relacionado, Inverse ATMARP, que se utiliza para descubrir la dirección IP y ATM de una computadora remota conectada por un circuito virtual permanente.

PARA CONOCER MÁS

Laubach (RFC 1577) introduce el concepto de Subred IP Lógica y define el protocolo ATMARP. Heinanen (RFC 1483) describe el uso de los encabezados LLC/SNAP en la encapsulación IP en AAL5, y Ackinson (RFC 1626) especifica la MTU por omisión.

Partridge (1994) describe en general el trabajo en redes con gigabits y la importancia de la conmutación de celdas en particular. De Prycker (1993) considera varios de los fundamentos teóricos de ATM y analiza su relación con la red telefónica.

EJERCICIOS

18.1 Si su organización tiene un conmutador ATM o un servicio ATM, encuentre cuáles son sus especificaciones técnicas y económicas, asimismo compare el costo de usar ATM con el de otras tecnologías como Ethernet.

18.2 Lea acerca de la interfaz TAXI. ¿Cómo se plantea el estándar?

18.3 Póngase en contacto con un vendedor de conmutadores ATM para determinar el ancho de banda agregado de un conmutador y el número máximo de computadoras que se puede conectar en él. ¿Con qué velocidad debe generar datos cada computadora para saturar el conmutador?

18.4 Una conexión común entre un anfitrión y un conmutador ATM privado opera a 155 Mbps. Considere la velocidad en el bus de su computadora favorita. ¿Qué porcentaje de este bus se necesita para mantener una interfaz ATM ocupada?

18.5 Varios sistemas operativos seleccionan un tamaño de búfer de TCP de 8K octetos. Si el IP fragmenta los datagramas para una MTU de 9180 octetos, ¿de qué tamaño resultarán los fragmentos de un datagrama que transporta un segmento TCP de 16K octetos y de 24K octetos?

18.6 ATM es un sistema de entrega con el mejor esfuerzo cuyo hardware puede descartar celdas si la red se congestiona. ¿Cuál es la probabilidad de perder una sola celda si la probabilidad de pérdida de una celda es $1/P$ y el datagrama es de 576 octetos de largo? ¿Y para los casos en que el número de octetos es igual a 1500, 4500 y 9180?

18.7 Una sesión remota en línea común utiliza el TCP para generar datagramas de 41 octetos: 20 octetos de encabezado IP, 20 octetos de encabezado TCP y 1 octeto de datos. ¿Cuántas celdas ATM se requieren para enviar un datagrama utilizando la encapsulación IP por omisión en AAL5?

18.8 ¿Cuántas celdas, octetos y bits pueden estar presentes en una fibra conectada a un conmutador ATM si la fibra es de 3 metros de largo?, ¿y en el caso de que la fibra tenga una longitud de 100 y 3000 metros? Para encontrar la solución considere que un conmutador ATM transmite datos a 155 Mbps. Cada bit es un pulso de luz con una duración de $1/(155 \times 10^6)$ segundos. Suponiendo que el pulso viaja a la velocidad de la luz, calcule el tiempo para las longitudes de la fibra.

18.9 Un anfitrión puede especificar una dirección ATM de dos niveles cuando solicita un SVC. ¿Qué topología de red es apropiada para el esquema de direccionamiento de dos niveles? Defina situaciones en que son útiles niveles jerárquicos adicionales.

18.10 Lea acerca de las capas 3 y 4 de adaptación ATM que originalmente fueron proyectadas para utilizarse con protocolos de transporte sin conexión y orientados a la conexión. ¿Cuál es la diferencia mayor entre ambos?

18.11 Una red ATM garantiza la entrega de celdas en orden, pero debe desechar celdas si comienza a congestionarse. ¿Es posible modificar el TCP a fin de aprovechar el ordenamiento de las celdas de modo que se reduzca la sobrecarga del protocolo? ¿Por qué sí o por qué no?

18.12 Existen productos que agregan una emulación de interfaz LAN a ATM, con lo que es posible simular FDDI u otras redes de área local. ¿Cuál es la mayor ventaja de usar ATM para emular otras LAN? ¿Cuál es la mayor desventaja?

18.13 Una organización extensa que utiliza ATM para interconectar anfitriones IP debe dividir los anfitriones en subredes IP lógicas. Existen dos extremos: en uno, la organización puede colocar todos los anfitriones en una sola LIS extensa o, en otro, debe tener varias LIS (por ejemplo, que cada par de anfitriones forme una LIS). Explique por qué ninguno de los dos extremos es deseable.

18.14 ¿Cuántas celdas ATM se requieren para transferir un solo paquete ATMARP si cada dirección y subdirección ATM es de 20 octetos y cada dirección de protocolo es de 4 octetos?

18.15 ATM permite a un anfitrión establecer varios circuitos virtuales hacia un destino dado. ¿Cuál es la mayor ventaja de hacer esto?

18.16 Mida el desempeño y el retardo de un conmutador ATM cuando utiliza TCP. Si su sistema operativo lo permite, repita el experimento con el búfer de transmisión del TCP configurado para varios tamaños (si su sistema utiliza sockets, consulte el manual para obtener mayores detalles sobre cómo establecer el tamaño del búfer). ¿El resultado le sorprende?

18.17 El IP no tiene un mecanismo para asociar datagramas que viajan a través de una red ATM con un circuito virtual específico. ¿Bajo qué circunstancias puede ser útil ese mecanismo?

18.18 Observe la propuesta del IP de la próxima generación descrita en el capítulo 29. ¿Qué nuevo mecanismo se relaciona directamente con ATM?

18.19 Un servidor no suprime inmediatamente una entrada en su memoria intermedia cuando el anfitrión que envía la información cierra la conexión hacia el servidor. ¿Cuál es la mayor ventaja de este diseño? ¿Cuál es la mayor desventaja?

19

Modelo de interacción cliente-servidor

19.1 Introducción

En los primeros capítulos presentamos los detalles de la tecnología TCP/IP, incluyendo los protocolos que proporcionan los servicios básicos y la arquitectura de ruteo que provee la información necesaria de ruteo. Ahora que comprendemos la tecnología básica, podemos examinar los programas de aplicación que se aprovechan del uso cooperativo de una red de redes de TCP. Las aplicaciones de ejemplo son prácticas e interesantes pero no hacen el énfasis principal. De hecho, el enfoque descansa sobre los patrones de interacción de los programas de aplicación de comunicación. El patrón de interacción primario que se da entre las aplicaciones de cooperación se conoce como paradigma *cliente-servidor*. La interacción cliente-servidor forma la base de la mayor parte de la comunicación por redes y es fundamental ya que nos ayuda a comprender las bases sobre las que están construidos los algoritmos distribuidos. En este capítulo, se considera la relación entre cliente y servidor; en capítulos posteriores se ilustra el patrón cliente-servidor con más ejemplos.

19.2 Modelo cliente servidor

El término *servidor* se aplica a cualquier programa que ofrece un servicio que se puede obtener en una red. Un servidor acepta la petición desde la red, realiza el servicio y devuelve el resultado al solicitante. En el caso de los servicios más sencillos, cada petición llega en un solo datagrama IP y el servidor devuelve una respuesta en otro datagrama.

Un programa ejecutable se convierte en un *cliente* cuando manda una petición a un servidor y espera una respuesta. Debido a que el modelo cliente-servidor es de extensión conveniente y natural en la comunicación de interproceso en una sola máquina, es fácil construir programas que utilicen el modelo para interactuar.

Los servidores pueden ejecutar tareas simples o complejas. Por ejemplo, un *servidor hora del día* simplemente devuelve la hora actual cuando un cliente manda un paquete al servidor. Un *servidor de archivo* recibe las peticiones para realizar las operaciones de almacenaje o recuperación de datos de un archivo; el servidor realiza la operación y devuelve el resultado.

Los servidores se suelen implantar como aplicaciones de programas.[1] La ventaja de implantar los servidores como programas de aplicación es que pueden ejecutarse en cualquier sistema computacional que soporte la comunicación TCP/IP. De este modo, el servidor de un servicio en particular puede ejecutarse en un sistema de tiempo compartido junto con otros programas o en una computadora personal. Los servidores múltiples pueden ofrecer el mismo servicio y ejecutarse en la misma máquina o en múltiples máquinas. De hecho, los administradores comúnmente duplican copias de un servidor dado en máquinas físicamente independientes para incrementar la disponibilidad o mejorar la ejecución. Si el propósito principal de una computadora es apoyar un programa servidor en particular, el término "servidor" se puede aplicar tanto a la computadora como al programa servidor. De este modo, podemos escuchar frases como "la máquina A es nuestro servidor de archivos".

19.3 Un ejemplo simple: servidor de eco UDP

La manera más simple de interacción cliente-servidor se vale de un datagrama de entrega no confiable para transportar los mensajes de un cliente a un servidor y de regreso. Consideremos, por ejemplo un *servidor de eco UDP*. Como se muestra en la figura 19.1, el aspecto mecánico es directo. En el lugar del servidor comienza un *proceso de servidor de eco* UDP mediante la negociación, con su sistema operativo, de la obtención del permiso para utilizar la ID de puerto UDP reservada para el servicio de *eco*, el *puerto de eco* UDP. Una vez que se ha obtenido el permiso, el proceso de servidor de eco entra en un ciclo interminable que incluye tres pasos: (a) espera a que el datagrama llegue al puerto de eco, (b) se invierten las direcciones de fuente y destino[2] (incluyendo tanto las direcciones de IP de fuente y destino como las identificaciones de UDP), y (c) se devuelve el datagrama al emisor original. En algún otro lugar, un programa se convierte en un UDP *eco-cliente* cuando ubica un puerto de protocolo UDP no utilizado, manda un mensaje UDP al UDP eco-servidor y espera la respuesta. El cliente espera recibir exactamente los mismos datos que mandó.

En el servicio de eco de UDP se ilustran dos puntos importantes acerca de la interacción cliente-servidor que por lo general son ciertos. El primero se refiere a la diferencia entre el tiempo de vida de los servidores y los clientes:

> *Un servidor comienza la ejecución antes de que empiece la interacción y (usualmente) continúa aceptando las peticiones y mandando las respuestas sin termi-*

---

[1] Muchos de los sistemas operativos se refieren a programas de aplicación que están corriendo como a un *proceso* o a un *proceso de usuario*.

[2] En uno de los ejercicios sugeridos se considera este paso con mayor detalle

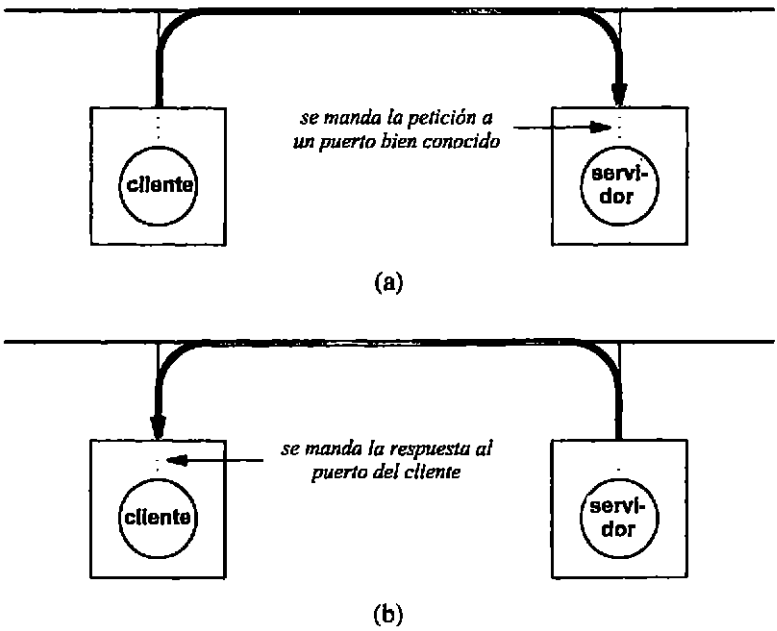


Figura 19.1 Eco UDP como ejemplo del modelo cliente-servidor. En el inciso (a) el cliente manda una petición al servidor a una dirección IP conocida y al puerto bien conocido UDP, y en el inciso (b) el servidor devuelve una respuesta. Los clientes utilizan cualquier puerto UDP que esté disponible.

nar nunca. Un cliente es cualquier programa que hace una petición, por lo general espera una respuesta y termina después de que ha utilizado un servidor un número finito de veces.

El segundo punto, mucho más técnico, se ocupa del uso de los identificadores de puerto reservados y no reservados.

Un servidor espera las peticiones en un puerto bien conocido que ha sido reservado para el servicio que ofrece. Un cliente ubica un puerto arbitrario no utilizado y no reservado para su comunicación.

En una interacción cliente-servidor se necesita reservar sólo uno de los dos puertos. Asignando un identificador único de puerto para cada servicio, se facilita la construcción de clientes y servidores.

¿Quién podrá utilizar un servicio eco? No todos los clientes promedio están interesados en este servicio. Sin embargo, los programadores que diseñan, implantan, miden o modifican el software de protocolo de la red o los gerentes de red que prueban las rutas y depuran los problemas de comunicación, a menudo, utilizan los servicios de eco en sus pruebas. Por ejemplo, un servicio de eco puede también emplearse para determinar si es posible conectarse con una máquina remota.

19.4 Servicio de fecha y hora

El servidor eco es muy sencillo y se requiere muy poca codificación para implantar ya sea el lado del servidor o el del cliente (siempre que el sistema operativo ofrezca una manera razonable de acceder a los protocolos UDP/IP implícitos). Nuestro segundo ejemplo, que es un servidor de hora, muestra que aun una sencilla interacción cliente-servidor puede proporcionar servicios útiles. El problema que un servidor de hora resuelve es el de definir el reloj de hora del día de una computadora. El reloj de hora del día es un dispositivo de hardware que mantiene la fecha y hora actuales, poniéndolos a disposición de los programas. Una vez que se ha definido, el reloj mantiene dicha hora del día tan precisa como un reloj de pulso.

Muchos sistemas resuelven el problema pidiéndole al programador que introduzca la hora y la fecha cuando se inicia el sistema. El sistema incrementa el reloj de manera periódica (es decir, cada segundo). Cuando un programa de aplicación pregunta por la fecha o la hora, el sistema consulta su reloj interno y da un formato a la hora del día en forma legible para cualquier persona. Podemos utilizar una interacción cliente-servidor para definir el sistema de reloj automáticamente cuando se inicia la máquina. Para hacerlo, el administrador configura una máquina que suele ser la máquina con el reloj de mayor exactitud, a fin de correr un servidor de hora del día. Cuando otras máquinas arrancan, se ponen en contacto con el servidor para obtener la hora actual.

19.4.1 Representación de la fecha y la hora

¿Cómo se supone que deberá mantener un sistema operativo la fecha y la hora del día? Una representación útil almacena la hora y la fecha como un conteo de segundos a partir de una fecha de época. Por ejemplo, el sistema operativo de UNIX utiliza el segundo cero del primero de enero de 1970 como su fecha de época. Los protocolos del TCP/IP también definen un fecha de época y reportan las horas conforme los segundos pasan la época. Para el TCP/IP, la época se define como el segundo cero del primero de enero de 1900 y la hora se mantiene en un entero de 32 bits, representación que se adaptará a todas las fechas de un futuro cercano.

Se mantiene a la fecha como a la hora en segundos pues la época hace que la representación se comparta y permite que se compare fácilmente. Enlaza a la fecha con la hora del día y hace posible que se mida el tiempo incrementando un simple entero binario.

19.4.2 Hora local y universal

Ya que se ha dado una fecha de época y una representación para la hora, ¿a la hora de qué zona se refiere el conteo? Cuando dos sistemas se comunican a través de grandes distancias geográficas, utilizar la zona de la hora local para una u otra se vuelve difícil; deben acordar una zona de hora estándar para mantener los valores de fecha y hora comparables. De este modo, además de definir una representación para la fecha y elegir una época, el servidor de tiempo TCP/IP estándar especifica que todos los valores se dan con respecto a una sola zona de tiempo. Originalmente se le llamaba tiempo medio de Greenwich, la zona de tiempo ahora se conoce como *tiempo coordinado universal* o *tiempo universal.*

La interacción entre un cliente y un servidor que ofrece servicio de tiempo funciona de manera muy parecida a un servidor eco. Del lado del servidor, la aplicación obtiene permiso para utilizar el puerto reservado asignado a los servidores de tiempo, espera el mensaje UDP dirigido a ese puerto y responde con un mensaje UDP que contiene la hora actual en un entero de 32 bits. Podemos resumir que:

El envío de un datagrama a un servidor de tiempo es equivalente a pedir la hora actual; el servidor responde con un mensaje UDP que contiene la hora actual.

19.5 La complejidad de los servidores

En nuestros ejemplos, los servidores son bastante sencillos debido a que son secuenciales. Esto quiere decir que el servidor procesa una petición a la vez. Después de aceptar una petición, el servidor forma una respuesta y la manda antes de volver a ver si ha llegado otra petición. Implícitamente asumimos que el sistema operativo hará una cola de peticiones que lleguen para un servidor mientras esté ocupado, y que dicha cola no será demasiado larga porque el servidor tiene sólo una pequeña cantidad de trabajo que realizar.

En la práctica, los servidores suelen ser mucho más difíciles de construir que los clientes, ya que necesitan acomodar varias peticiones concurrentes, aun cuando una sola petición se lleve una cantidad de tiempo considerable para ser procesada. Por ejemplo, consideremos que un servidor de transferencia de archivos es el responsable de copiar un archivo a otra máquina bajo pedido. En general, los servidores tiene dos partes. Un programa maestro sencillo, responsable de aceptar nuevas peticiones, y un conjunto de esclavos, los responsables de manejar las peticiones individuales. El servidor maestro ejecuta los cinco pasos siguientes:

Abrir puerto
El servidor maestro abre el puerto bien conocido al que se puede accesar.

En espera del cliente
El maestro espera a que un nuevo cliente mande una petición.

Elección de puerto
Si es necesario, el maestro ubica un nuevo puerto de protocolo local para esta petición e informa al cliente (veremos más adelante que este paso es innecesario con el TCP/IP).

Se inicia el esclavo
El maestro inicia un esclavo independiente y concurrente para que maneje esta petición (por ejemplo, en UNIX, se realiza una copia del proceso del servidor). Nótese que el esclavo maneja una petición y después termina, el esclavo no espera a que lleguen peticiones de otros clientes.

Continua
El maestro regresa al paso de *espera* y continúa aceptando nuevas peticiones mientras el esclavo recientemente creado maneja de manera concurrente las peticiones previas.

Como el maestro inicia un esclavo para cada nueva petición el procesamiento procede de manera concurrente. De este modo, las peticiones que requieren de poco tiempo para completarse se pueden terminar antes que las peticiones que se llevan más tiempo, independientemente del orden en que se hayan comenzado. Por ejemplo, supongamos que el primer cliente que contacta a un servidor de archivos pide la transferencia de un archivo grande que se llevará varios minutos. Si un segundo cliente se pone en contacto con el servidor para pedir una transferencia que se lleva solamente unos segundos, la segunda transferencia puede iniciarse y completarse mientras que la primera transferencia aún continúa.

Además de la complejidad que resulta de que los servidores manejen peticiones concurrentes, la complejidad también surge porque los servidores deben reforzar las reglas de autorización y protección. Los programas servidor suelen requerir una ejecución de alta prioridad pues tienen que leer archivos del sistema, mantenerse en línea y tener acceso a datos protegidos. El sistema operativo no restringirá un programa servidor si intenta tener acceso a los archivos del usuario. De este modo, los servidores no pueden cumplir a ciegas las peticiones de otras localidades. Por el contrario, cada servidor toma la responsabilidad para reforzar el acceso al sistema y las políticas de protección.

Por último, los servidores deben protegerse a sí mismos contra las peticiones formadas equivocadamente o contra las peticiones que causarán que el mismo programa servidor se aborte. A menudo, es difícil prever los problemas potenciales. Por ejemplo, en un proyecto, en la Universidad de Purdue, se diseñó un servidor de archivos que permitió que los sistemas operativos de los estudiantes accesaran archivos en un sistema UNIX de tiempo compartido. Los estudiantes descubrieron que la petición al servidor de que abriera un archivo llamado */dev/tty* ocasionaba que el servidor abortara el proceso pues UNIX asocia ese nombre con la terminal de control a la que está unida un programa. El servidor que fue creado por una iniciación de sistema no tenía dicha terminal. Una vez que se abortaba el proceso, ningún cliente podía accesar archivos hasta que un programador de sistemas reiniciaba el servidor.

Hubo casos más serios de la vulnerabilidad del servidor en el verano de 1988, cuando un estudiante de la Universidad Cornell diseñó un programa *gusano* que atacó a las computadoras en toda la red Internet. Una vez que el gusano comenzó a correr en una máquina, buscó el acceso a Internet para llegar a computadoras con servidores que sabía cómo explotar y usó a los servidores para crear más copias de sí mismo. En uno de los ataques, el gusano aprovechó un bug[3] en el servidor *fingerd* de UNIX. Debido a que el servidor no revisó las peticiones que entraban, el gusano fue capaz de mandar una cadena de entrada ilegal que causó que el servidor sobrescribiera partes de sus áreas internas de datos. El servidor, que se ejecutaba con el privilegio más alto, no se comportó debidamente y permitió que el gusano creara copias de sí mismo.

Podemos resumir nuestro análisis sobre servidores de la siguiente forma:

> *Los servidores suelen ser más difíciles de construir que los clientes porque, aun cuando pueden implantarse con programas de aplicación, los servidores deben reforzar todas los procedimientos de acceso y protección del sistema computacional en el que corren, además tienen que protegerse contra todos los errores posibles.*

---

[3] N. del T.: textualmente, bicho o piojo. Se refiere a un desperfecto en el programa.

19.6 Servidor RARP

Hasta ahora, todos nuestros ejemplos de interacción cliente-servidor requieren que el cliente sepa la dirección completa del servidor. El protocolo RARP, del capítulo 6, proporciona un ejemplo de interacción cliente-servidor con un cambio levemente diferente. Recordemos que cuando una máquina sin disco se reinicia, utiliza RARP para encontrar su dirección IP. En lugar de tener al cliente comunicado directamente con el servidor, los clientes de RARP difunden sus peticiones. Una o más máquinas ejecutan los procesos de respuesta del servidor RARP y cada una de ellas devuelve un paquete que responde a la búsqueda.

Hay dos diferencias significativas entre el servidor RARP y un eco UDP o servidor de tiempo. En primer lugar, los paquetes RARP viajan a través de una red física directamente en las estructuras del hardware y no en los datagramas del IP. De este modo, a diferencia del servidor UDP de eco que permite al cliente ponerse en contacto con un servidor en cualquier lugar de la red de redes, el servidor RARP requiere que el cliente esté en la misma red física. En segundo lugar, RARP no puede implantarse mediante un programa de aplicación. Los servidores de eco y tiempo se pueden construir como programas de aplicación porque utilizan el UDP. En contraste, un servidor RARP necesita tener acceso a los paquetes de hardware primarios.

19.7 Alternativas al modelo cliente-servidor

¿Cuáles son las alternativas para la interacción cliente-servidor y cuándo podrían éstas ser atractivas? En esta sección se ofrece al menos una respuesta a estas preguntas.

En el modelo cliente servidor, los programas suelen actuar como clientes cuando necesitan información, pero algunas veces es importante minimizar dichas interacciones. El protocolo ARP del capítulo 5 nos brinda un ejemplo. Utiliza una forma modificada de la interacción cliente-servidor para obtener transformaciones de las direcciones físicas. Las máquinas que se valen de ARP mantienen una memoria intermedia (caché) de respuestas para mejorar la eficiencia de las búsquedas que surjan después. El proceso de memoria intermedia (caching) mejora el desempeño de la interacción cliente-servidor en casos en los que la historia reciente de búsquedas ha sido un buen indicador de lo que será su uso futuro.

Aunque el proceso de memoria intermedia mejora el desempeño, no cambia la esencia de la interacción cliente-servidor. La esencia descansa en que suponemos que el procesamiento debe ester dirigido por la demanda. Todos hemos asumido que un programa se ejecuta hasta que se necesita información y, entonces, actúa como un cliente para obtener la información que necesita. Adoptar una opinión del mundo enfocada a la demanda es natural y surge de la experiencia. El proceso de memoria intermedia ayuda a aliviar el costo de la obtención de información, bajando los costos de recuperación para todo a excepción del primer proceso en el que se hace una petición.

¿Cómo podemos disminuir el costo de recuperación de información en la primera petición? En un sistema distribuido, es posible tener actividades de respaldo concurrentes que recolecten y difundan la información *antes* de que algún programa en particular la requiera, logrando que los costos de recuperación sean aún más bajos para la petición inicial. Lo más importante es que la prerrecolección de información permite que un sistema dado continúe ejecutándose aun cuando otras máquinas o redes conectadas a ella hayan fallado.

La prerrecolección es la base para el comando *ruptime* de UNIX 4BSD. Cuando se invoca *ruptime*, éste reporta la carga de CPU e indica desde hace cuánto tiempo se inició el sistema de cada máquina en la red local. Un programa de respaldo que corre en cada máquina emplea el UDP para difundir la información acerca de la máquina de manera periódica. El mismo programa también recolecta la información de entrada y la coloca en un archivo. Debido a que las máquinas difunden información continuamente, cada máquina tiene una copia de la última información a la mano; el cliente que busque información, nunca necesitará acceder a la red. De hecho, puede leer la información de un almacén secundario e imprimirla para su lectura.

La ventaja principal de tener la información reunida localmente, antes de que el cliente la necesite, es la velocidad. El comando *ruptime* responde de manera inmediata cuando se invoca sin esperar los mensajes que atraviesan la red. El segundo beneficio ocurre cuando el cliente puede encontrar algo acerca de las máquinas que ya no están operando. En particular, si una máquina deja de radiotransmitir información, el cliente puede reportar el tiempo que ha transcurrido desde la última radiotransmisión (es decir, puede reportar cuánto tiempo ha estado la máquina fuera de línea).

La prerrecolección tiene una gran desventaja: utiliza tiempo del procesador y ancho de banda de la red, aun cuando a nadie le importen los datos que se están recolectando. Por ejemplo, la difusión de ruptime y la recolección continúan corriendo durante toda la noche, aunque nadie esté presente para leer la información. Si sólo algunas máquinas están conectadas a una red dada, el costo de prerrecolección es insignificante. Se puede pensar como una actividad de respaldo inocua. Sin embargo, para las redes con muchos anfitriones, el gran volumen de tráfico de difusión generado por la prerrecolección, se hace muy caro. En particular, el costo de leer y procesar los mensajes de radiodifusión se vuelve alto. De este modo, la prerrecolección no está entre las alternativas más populares para cliente-servidor.

19.8 Resumen

Los programas distribuidos requieren comunicación en red. Dichos programas a menudo caen dentro de un patrón de uso conocido como interacción cliente-servidor. Un proceso de servidor espera una petición y ejecuta una acción basada en la petición. La acción suele incluir el envío de una respuesta. Un programa cliente formula una petición, la manda al servidor y después espera una respuesta.

Hemos visto ejemplos de clientes, de servidores y hemos encontrado que algunos clientes mandan peticiones directamente, mientras que otros difunden las peticiones. La radiotransmisión es útil en especial en un área local cuando una máquina no sabe la dirección de un servidor.

También, notamos que, si los servidores utilizan protocolos de la red de redes como el UDP, pueden aceptar y responder a las peticiones a través de una red de redes. Si se comunican mediante estructuras físicas y direcciones de harware físicas, se les restringe a una red física única.

Por último, consideramos una alternativa para el paradigma cliente-servidor que utiliza a la prerrecolección de información para evitar demoras. Un ejemplo de prerrecolección se obtiene a partir de un servicio de estado de máquina.

PARA CONOCER MÁS

El servicio UDP de eco se define en Postel (RFC 862). En el *Manual de programadores de UNIX* se describe el comando *ruptime* (ver también *rwho* que es una descripción relacionada). Feinler *et. al.* (1985) especifica muchos protocolos de servidor estándar que no se trataron aquí, incluyendo el descarte (*discard*), la generación de caracteres (*character generation*), día y hora (*day and time*), usuarios activos (*active users*), y cita del día (*quote of the day*). En el siguiente capítulo se consideran otros.

EJERCICIOS

19.1 Construir un cliente UDP eco que mande un datagrama a un servidor eco específico, esperar una respuesta y compararla con el mensaje original.

19.2 Considere cuidadosamente la manipulación de las direcciones IP en un servidor UDP de eco. ¿Bajo qué condiciones es incorrecto crear nuevas direcciones IP mediante la inversión de las direcciones IP de fuente y destino?

19.3 Como hemos visto, los servidores se pueden implantar por medio de programas de aplicación separados o mediante el código de construcción de servidor dentro del software de protocolo en un sistema operativo. ¿Cuáles son las ventajas y las desventajas de tener un programa de aplicación (proceso del usuario) por servidor?

19.4 Supongamos que usted no conoce la dirección IP de una máquina local que corre un servidor UDP de eco, pero sabe que responde a las peticiones mandadas al puerto 7. ¿Existe alguna dirección IP que pueda utilizar para conectarse?

19.5 Construya un cliente para un servicio de tiempo UDP.

19.6 Caracterice situaciones en las que un servidor se pueda ubicar en una red física separada de su cliente. ¿Puede un servidor RARP ubicarse en algún momento en una red física separada de sus clientes? ¿Por qué sí o por qué no?

19.7 ¿Cuál es la desventaja principal de que todas las máquinas difundan su estado periódicamente?

19.8 Examine el formato de datos difundidos por los servidores que implanta el comando *ruptime* de UNIX 4BSD. ¿Qué información está disponible para el cliente además del estado de la máquina?

19.9 ¿Qué servidores están corriendo en las computadoras donde usted se ubica? Si no tiene acceso a los archivos de configuración del sistema que listan los servidores iniciados para una computadora dada, vea si su sistema tiene un comando que imprima una lista de puertos TCP y UDP abiertos (por ejemplo, el comando *netstat* de UNIX).

19.10 Algunos servidores permiten que el administrador los apague o los reinicie. ¿Cuál es la ventaja de ello?

20

La interfaz socket

20.1 Introducción

Hasta aquí, nos hemos concentrado en tratar los principios y conceptos que sustentan los protocolos TCP/IP sin especificar la interfaz que existe entre los programas de aplicación y el software de protocolo. En este capítulo, veremos el ejemplo de una interfaz entre programas de aplicación y protocolos TCP/IP. Existen dos razones para posponer el análisis acerca de las interfaces. En primer lugar, debemos distinguir entre los protocolos de interfaz y el TCP/IP debido a que los estándares no especifican exactamente cómo es que interactúan los programas de aplicación con el software de protocolo. Por ello, la arquitectura de interfaz no está estandarizada; su diseño descansa fuera del campo de lo relacionado con el protocolo. En segundo lugar, en la práctica, es inapropiado unir a los protocolos con una interfaz en particular pues ninguna arquitectura de interfaz funciona bien en todos los sistemas. En particular, como el software de protocolo reside en el sistema operativo de una computadora, los detalles de la interfaz dependen del sistema operativo.

A pesar de la carencia de un estándar, la revisión de un ejemplo nos ayudará a comprender cómo es que emplean los programadores el TCP/IP. Aunque el ejemplo que hemos escogido es del sistema operativo de BSD de UNIX, se ha aceptado ampliamente y se usa en muchos otros sistemas. En particular, la interfaz *Winsock* proporciona la funcionalidad socket para Microsoft Windows. El lector deberá recordar que nuestra meta es tan sólo dar un ejemplo concreto y no prescribir cómo es que deberían estar diseñadas las intefaces. El lector no deberá olvidar tampoco que las operaciones listadas aquí no comprenden una estandarización en ningún sentido.

20.2 El paradigma E/S de UNIX y la E/S de la red

Unix fue desarrollado a finales de los años sesenta y principios de los setenta, y se diseñó originalmente como un sistema de tiempo compartido para computadoras de un solo procesador. Se trata de un sistema operativo orientado al proceso, en el que cada programa de aplicación se ejecuta como un proceso de nivel de usuario. Un programa de aplicación interactúa con el sistema operativo haciendo *llamadas de sistema*. Desde el punto de vista del programador, las llamadas de sistema se ven y comportan exactamente igual que las demás llamadas de procedimiento. Toman argumentos y devuelven uno o más resultados. Los argumentos pueden ser valores (por ejemplo, una operación de enteros) o punteros a objetos en el programa de aplicación (como un búfer que ha de ser llenado con caracteres).

Derivados de los Multics y los sistemas anteriores, los sistemas primitivos de entrada y salida (E/S, ENTRADA/SALIDA) de UNIX siguen un paradigma que algunas veces se denomina *open-read-write-close* (*abrir-leer-escribir-cerrar*). Antes de que un proceso de usuario pueda ejecutar operaciones de E/S, llama a *open* para especificar el archivo o dispositivo que se va a usar y obtiene el permiso. La llamada a *open* devuelve un pequeño entero *descriptor de archivo*[1] que el proceso utiliza cuando ejecuta las operaciones de E/S en el archivo o dispositivo abierto. Una vez que se ha abierto un objeto, el proceso de usuario hace una o más llamadas a *read* o *write* para transferir datos. *Read* transfiere datos dentro del proceso de usuario; *write* transfiere datos del proceso de usuario al archivo o dispositivo. Tanto *read* como *write* toman tres argumentos que especifican el descriptor de archivo que se ha de usar, la dirección de un búfer y el número de octetos que se han de transferir. Luego de completar todas las operaciones de transferencia, el proceso de usuario llama a *close* para informar al sistema operativo que ha terminado de usar el objeto (el sistema operativo cierra de manera automática todos los descriptores abiertos si ningún proceso llama a *close*).

20.3 Adición de la red E/S a UNIX

Originalmente, los diseñadores de UNIX agruparon todas las operaciones de E/S en el paradigma open-read-write-close descrito arriba. El esquema incluyó E/S para los dispositivos orientados a caracteres como los teclados y para los dispositivos orientados a bloques como los discos y los archivos de datos. Una de las primeras implantaciones del TCP/IP bajo UNIX también utilizó el paradigma open-read-write-close con un nombre de archivo especial, */dev/tcp*.

El grupo que añadió los protocolos de red a BSD de UNIX decidió que, como los protocolos de la red eran más complejos que los dispositivos convencionales de E/S, la interacción entre los procesos del usuario y los protocolos de red debía ser más compleja que las interacciones entre los procesos de usuario y las instalaciones convencionales de E/S. En particular, la interfaz de protocolo debía permitir a los programadores crear un código de servidor que esperara las conexiones pasivamente, así como también un código cliente que formara activamente las conexiones. Además, los programas de aplicación que mandaban datagramas podían especificar la dirección de destino

[1] El término "descriptor de archivo" surge porque, en UNIX, todos los dispositivos son transformados en el archivo de espacio de nombre de sistema. En la mayor parte de los casos, las operaciones de E/S en los archivos y dispositivos no se pueden distinguir.

junto con cada datagrama en lugar de destinos enlazados en el momento en que llamaban a *open*. Para manejar todos estos casos, los diseñadores eligieron abandonar el paradigma tradicional de UNIX open-read-write-close y añadir varias llamadas nuevas del sistema operativo, así como también una nueva biblioteca de rutinas. La adición de protocolos de red a UNIX incrementó sustancialmente la complejidad de la interfaz de E/S.

Posteriormente surgió una mayor complejidad en la interfaz de protocolos de UNIX pues los diseñadores intentaron construir un mecanismo general para adaptar muchos protocolos. Por ejemplo, la generalidad hace posible, para el sistema operativo, incluir software para otros conjuntos de protocolos así como también el TCP/IP y permitir que un programa de aplicación utilice uno o más de ellos a la vez. Como consecuencia, el programa de aplicación no sólo puede proporcionar una dirección de 32 bits y esperar a que el sistema operativo la interprete de manera correcta. La aplicación debe especificar explícitamente que el número de 32 bits representa una dirección IP.

20.4 La abstracción de socket

La base para la E/S de red en BSD de UNIX se centra en una abstracción conocida como *socket*.[2] Pensamos al socket como una generalización del mecanismo de acceso a archivos de UNIX que proporciona un punto final para la comunicación. Al igual que con el acceso a archivos, los programas de aplicación requieren que el sistema operativo cree un socket cuando se necesita. El sistema devuelve un entero pequeño que utiliza el programa de aplicación para hacer referencia al socket recientemente creado. La diferencia principal entre los descriptores de archivos y los descriptores de socket es que el sistema operativo enlaza un descriptor de archivo a un archivo o dispositivo específico cuando la aplicación llama a *open*, pero puede crear sockets sin enlazarlos a direcciones de destino específicas. La aplicación puede elegir abastecer una dirección de destino cada vez que utiliza el socket (es decir, cuando se envían datagramas) o elegir enlazar la dirección de destino a un socket y evadir la especificación de destino repetidamente (es decir, cuando se hace una conexión TCP).

Cada vez que tenga sentido, los sockets hacen ejecuciones exactamente iguales a los archivos o dispositivos de UNIX, de manera que pueden utilizarse con operaciones tradicionales, como *read* y *write*. Por ejemplo, una vez que un programa de aplicación crea un socket y una conexión TCP del socket a un destino externo, el programa puede hacer uso de *write* para mandar un flujo de datos a través de la conexión (el programa de aplicación en el otro extremo puede utilizar *read* para recibirlo). Para hacer posible que se utilicen las primitivas *read* y *write* con archivos y sockets, el sistema operativo ubica los descriptores de socket y los descriptores de archivo del mismo conjunto de enteros y se asegura de que, si un entero dado se ha ubicado como descriptor de archivo, no será ubicado también como descriptor de socket.

---

[2] Por ahora describiremos a los sockets como parte del sistema operativo pues es la manera en que BSD de UNIX los proporciona; en secciones posteriores se describe cómo es que otros sistemas operativos utilizan las bibliotecas de rutinas para proporcionar una interfaz de socket.

20.5 Creación de un socket

El sistema de llamada *socket* crea sockets cuando se le pide. Toma tres argumentos enteros y devuelve un resultado entero:

resultado = socket (pf, tipo, protocolo)

El argumento *pf* especifica la familia del protocolo que se va a utilizar con el socket. Esto quiere decir que especifica cómo interpretar la dirección cuando se suministra. Las familias actuales incluyen la red de redes TCP/IP (PF_INET), la Red de redes PUP de Xerox Corporation (PF_PUP), la red Appletalk de Apple Computer Incorporated (PF_APPLETALK) y el sistema de archivos UNIX (PF_UNIX) así como también muchos otros.[3]

El argumento *tipo* especifica el tipo de comunicación que se desea. Los tipos posibles incluyen el servicio de entrega confiable de flujo (SOCK_STREAM) y el servicio de entrega de datagramas sin conexión (SOCK_DGRAM), así como también el tipo creado o no procesado (SOCK_RAW) que permite a los programas de privilegio accesar a los protocolos de bajo nivel o a las interfaces de red. Otros dos tipos adicionales ya se han planeado pero no han sido implantados.

Aunque el acercamiento general de separación de las familias y los tipos de protocolo puede parecer suficiente para manejar todos los casos con facilidad, no es así. En primer lugar puede ser que una familia de protocolos dada no soporte uno o mas de los tipos de servicio posibles. Por ejemplo, la familia UNIX tiene un mecanismo de comunicación de interproceso, llamado *pipe*, que utiliza un servicio de entrega de flujo confiable, pero no posee mecanismo de entrega de paquetes secuenciados. De este modo, no todas las combinaciones de familias de protocolos y tipos de servicio tienen sentido. En segundo lugar, algunas familias de protocolos poseen protocolos diversos que soportan un tipo de servicio. Por ejemplo podría ser que una sola familia de protocolos tuviera dos servicios de entrega de datagramas sin conexión. Para acomodar los diversos protocolos dentro de una familia, la llamada de *socket* tiene un tercer argumento, que se puede emplear para seleccionar un protocolo específico. Para usar el tercer argumento, el programador debe conocer la familia de protocolos lo suficientemente bien como para saber el tipo de servicio que cada protocolo brinda.

Como los diseñadores trataron de capturar muchas de las operaciones convencionales de UNIX en su diseño socket, necesitaban una manera de simular el mecanismo pipe. No es necesario comprender los detalles de los pipes; sólo una de las características es importante: los pipes difieren de las operaciones de red estándar pues el proceso de llamado crea ambos puntos de terminación para la comunicación de manera simultánea. Para acomodar los pipes, los diseñadores añadieron un sistema de llamado *socketpair* (*apareamiento de sockets*) que toma la siguiente forma:

socketpair(pf, tipo, protocolo, sarray)

Socketpair tiene un argumento más que en el procedimiento *socket*, se trata de *sarray*. Dicho argumento adicional da la dirección de un arreglo entero de dos elementos. *Socketpair* crea dos sockets de manera simultánea y coloca dos descriptores de socket en los dos elementos de *sarray*.

---

[3] En UNIX, los programas de aplicación contienen nombres simbólicos como *PF_INET*; los archivos de sistema contienen las definiciones que especifican valores numéricos para cada nombre.

Los lectores deberán comprender que *socketpair* no es significativo cuando se aplica a la familia de protocolos TCP/IP (lo incluimos aquí sólo para completar nuestra descripción de la interfaz).

20.6 Herencia y finalización del socket

UNIX utiliza las llamadas del sistema *fork* y *exec* para comenzar nuevos programas de aplicación. Se trata de un procedimiento de dos pasos. En el primer paso, *fork* crea una copia separada del programa de aplicación que se está ejecutando en ese momento. En el segundo paso, la nueva copia se reemplaza a sí misma con el programa de aplicación deseado. Cuando una programa llama a *fork*, la copia que se acaba de crear le hereda el acceso a todos los sockets abiertos, justo como si heredara el acceso a todos los archivos abiertos. Cuando el programa llama a *exec*, la nueva aplicación retiene el acceso para todos los sockets abiertos. Veremos que los servidores maestros utilizan la herencia socket cuando crean servidores esclavos o manejan una conexión específica. Internamente, el sistema operativo mantiene una cuenta de referencia asociada con cada socket de manera que se sabe cuántos programas de aplicación (procesos) han accedido a él.

Tanto los procesos viejos como los nuevos tienen los mismos derechos de acceso para los sockets existentes, y ambos tipos pueden accesar a los sockets. Luego pues, es responsabilidad del programador asegurar que los dos procesos empleen el significado del socket compartido.

Cuando un proceso termina de utilizar un socket, éste llama a *close* (*cerrar*). *Close* tiene la forma:

close(socket)

donde el argumento *socket* especifica al descriptor de un socket para que cierre. Cuando un proceso se termina por cualquier razón, el sistema cierra todos los sockets que permanecen abiertos. Internamente, una llamada a *close* (*cerrar*) disminuye la cuenta de referencia para un socket y destruye al socket si la cuenta llega a cero.

20.7 Especificación de una dirección local

Inicialmente, un socket se crea sin ninguna asociación hacia direcciones locales o de destino. Para los protocolos TCP/IP, esto significa que ningún número de puerto de protocolo local se ha asignado y que ningún puerto de destino o dirección IP se ha especificado. En muchos casos, los programas de aplicación no se preocupan por las direcciones locales que utilizan, ni están dispuestos a permitir que el software de protocolo elija una para ellos. Sin embargo los procesos del servidor que operan en un puerto bien conocido deben ser capaces de especificar dicho puerto para el sistema. Una vez que se ha creado un socket, el servidor utiliza una llamada del sistema *bind (enlace)* para establecer una dirección local para ello. *Bind* tiene la siguiente forma:

bind(socket, localaddr, addrlen)

El argumento *socket* es el descriptor de enteros del socket que ha de ser enlazado. El argumento *localaddr* es una estructura que especifica la dirección local a la que el socket deberá enlazarse, y el argumento *addrlen* es un entero que especifica la longitud de las direcciones medidas en octetos. En lugar de dar la dirección, solamente como una secuencia de octetos, los diseñadores eligieron utilizar una estructura para las direcciones, como se ilustra en la figura 20.1.

0	16 31
FAMILIA DE DIRECCIONES	OCTETOS DE DIRECCIÓN 0-1
OCTETOS DE DIRECCIÓN 2-5	
OCTETOS DE DIRECCIÓN 6-9	
OCTETOS DE DIRECCIÓN 10-13	

Figura 20.1 La estructura *sockaddr* que se utiliza cuando se pasa una dirección TCP/IP para la interfaz socket.

La estructura, que genéricamente se denomina *sockaddr*, comienza como un campo *ADDRESS FAMILY* (*FAMILIA DE DIRECCIONES*) de 16 bits que identifica el conjunto de protocolos al que pertenece la dirección. Va seguido por una dirección de hasta *14* octetos. Cuando se declara en C, la dirección de la estructura de socket es una unión de estructuras para todas las familias de direcciones posibles.

El valor en el campo *ADDRESS FAMILY* determina el formato de los octetos de las direcciones restantes. Por ejemplo, el valor 2[4] en el campo *ADDRESS FAMILY* significa que los octetos de direcciones restantes contienen una dirección TCP/IP. Cada familia de protocolos define cómo se usarán los octetos en el campo de dirección. Para las direcciones TCP/IP, la dirección de socket se conoce como *sockaddr_in*. Esto incluye una dirección del IP y un número de puerto de protocolo (es decir, una dirección de socket de red de redes cuya estructura puede contener direcciones IP y puertos de protocolo en la dirección). En la figura 20.2 se muestra el formato exacto de una dirección de socket TCP/IP.

Aunque es posible especificar valores arbitrarios en la estructura de la dirección cuando se llama a *bind*, no todos los enlaces posibles son válidos. Por ejemplo, quien llame podría pedir un puerto de protocolo local que ya esté en uso para otro programa, o podría pedir una dirección IP no válida. En tales casos, la llamada *bind* falla y se devuelve un código de error.

20.8 Conexión de socket con direcciones de destino

Inicialmente, un socket se crea en el *unconnected state* (*estado no conectado*), lo que significa que el socket no está asociado con ningún destino externo. La llamada de sistema *connect* (*conectar*)

[4] UNIX utiliza el nombre simbólico PF_INET para denotar direcciones TCP/IP.

0	16 31
FAMILIA DE DIRECCIONES (2)	PUERTO DE PROTOCOLO
DIRECCIÓN IP	
NO UTILIZADA (CERO)	
NO UTILIZADA (CERO)	

Figura 20.2 El formato de una estructura de dirección de socket (*sockaddr_in*) cuando se usa con una dirección TCP/IP. La estructura incluye ambas, una dirección IP y un puerto de protocolo en la dirección.

enlaza un destino permanente a un socket, colocándolo en el *connected state* (*estado conectado*). Un programa de aplicación debe llamar a *connect* para establecer una conexión antes de que pueda transferir datos a través de un socket de flujo confiable. Los sockets utilizados con servicios de datagrama sin conexión necesitan no estar conectados antes de usarse, pero haciéndolo así, es posible transferir datos sin especificar el destino en cada ocasión.

La llamada de sistema *connect* tiene la forma:

connect(socket, destaddr, addrlen)

El argumento *socket* es el descriptor entero del socket que ha de conectarse. El argumento *destaddr* es una estructura de dirección socket en la que se especifica la dirección de destino a la que deberá enlazarse el socket. El argumento *addrlen* especifica la longitud de la dirección de destino medida en octetos.

El significado de *connect* depende de los protocolos subyacentes. La selección del servicio de entrega de flujo confiable en la familia PF_INET significa elegir al TCP. En tales casos, *connect* construye una conexión con el destino y devuelve un error si no puede. En el caso del servicio sin conexión, *connect* no hace nada más que almacenar localmente la dirección de destino.

20.9 Envío de datos a través de un socket

Una vez que el programa de aplicación ha establecido un socket, puede usar el socket para transmitir datos. Hay cinco llamadas posibles de sistemas operativos de entre las que se puede escoger: *send* (*mandar*), *sendto* (*mandar a*), *sendmsg* (*mandar mensaje*), *write* (*escribir*) y *writev* (*vector escribir*). *Send*, *write* y *writev* sólo trabajan con sockets conectados pues no permiten que quien llama especifique una dirección de destino. Las diferencias entre los tres son menores. *Write* toma tres argumentos:

write(socket, buffer, length)

El argumento *socket* contiene un descriptor de socket entero (*write* puede también usarse con otros tipos de descriptores). El argumento *buffer* contiene la dirección de datos que se han de man-

dar y el argumento *length* especifica el número de octetos que se han de mandar. La llamada a *write* se bloquea hasta que los datos se pueden transferir (es decir, se bloquea si los búfers del sistema interno para el socket están llenos). Como en la mayor parte de las llamadas de sistema en UNIX, *write* devuelve un código de error para la aplicación que llama, lo cual permite al programador saber si la operación tuvo éxito.

La llamada de sistema *writev* funciona como *write* salvo porque utiliza la forma "gather write (escritura de recolección)", la cual hace posible que el programa de aplicación escriba un mensaje sin copiar el mensaje en los octetos de memoria contiguos. *Writev* tiene la forma:

writev(socket, iovector, vectorlen)

El argumento *iovector* da la dirección de un arreglo de tipo *iovec*, el cual contiene una secuencia de apuntadores de los bloques de octetos que forman el mensaje. Como se muestra en la figura 20.3, a cada apuntador lo acompaña una longitud. El argumento *vectorlen* especifica el número de entradas en *iovector*.

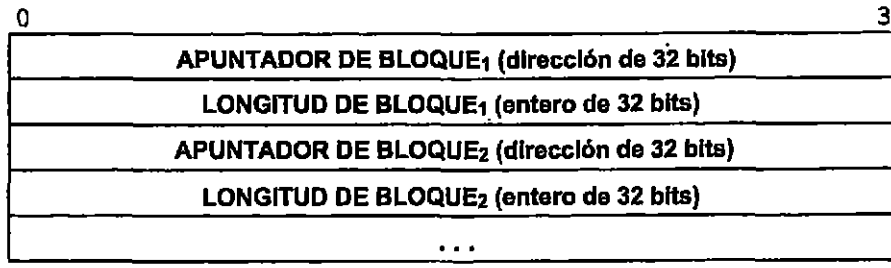


Figura 20.3 Formato de un iovector (vector e/s) de tipo *iovec* que se usa con *writev (vector escribir)* y *readv* (*vector leer*).

La llamada de sistema *send* tiene la forma:

send(socket, message, length, flags)

donde el argumento *socket* especifica el socket que se ha de usar, el argumento *message* (*mensaje*) da la dirección de datos a ser mandados, el argumento *length* (*longitud*) especifica el número de octetos que se va a enviar y el argumento *flags* (*banderas*) controla la transmisión. Un valor para *flags* permite que quien envía especifique que el mensaje se deberá enviar fuera de banda en los sockets que soporten tal noción. Por ejemplo, recordemos que, en el capítulo 13, se decía que los mensajes fuera de banda corresponden a la noción de datos urgentes del TCP. Otro valor para *flags* permite que quien llama pida que el mensaje se envíe sin necesidad de usar tablas de ruteo local. La intención es permitir a quien llama que tome el control del ruteo, haciendo posible que se escriba en el software de depuración de red. Por supuesto, no todos los sockets soportan todas las peticiones de cualquier programa. Algunas peticiones requieren que el programa tenga privilegios especiales y hay otros programas que sencillamente no se soportan en todos los sockets.

Las llamadas de sistema *sendto* y *sendmsg* permiten que quien llama envíe un mensaje a través de un socket no conectado, pues ambos requieren que quien llama especifique un destino. *Sendto*, que toma la dirección de destino como argumento, tiene la forma:

sendto(socket, message, length, flags, destaddr, addrlen)

Los primeros cuatro argumentos son exactamente los mismos que se usaron con la llamada de sistema *send*. Los dos argumentos finales especifican una dirección de destino y dan la longitud de dicha dirección. El argumento *destaddr* especifica la dirección de destino utilizando la estructura *sockaddr_in* como se define en la figura 20.2

Un programador puede elegir usar la llamada de sistema *sendmsg* en casos en los que la larga lista de argumentos requeridos para *sendto* haga que el programa sea ineficaz o difícil de leer. *Sendmsg* tiene la forma:

sendmsg(socket, messagestruct, flags)

donde el argumento *messagestruct* (*estructura de mensaje*) es una estructura de la forma ilustrada en la figura 20.4. La estructura contiene información acerca del mensaje que se ha de mandar, su longitud, la dirección de destino y la longitud de la dirección. Esta llamada es especialmente útil porque no hay una operación de entrada correspondiente (se describe abajo) que produzca una estructura de mensaje exactamente en el mismo formato.

0 … 31
APUNTADOR A SOCKETADDR
TAMAÑO DE SOCKETADDR
APUNTADOR DE LISTA IOVEC
LONGITUD DE LISTA IOVEC
APUNTADOR DE LISTA DE DERECHOS DE ACCESO
LONGITUD DE LISTA DE DERECHOS DE ACCESO

Figura 20.4 Formato de la estructura de mensaje *messagestruct* utilizado por *sendmsg*.

20.10 Recepción de datos a través de un socket

BSD de UNIX ofrece cinco llamadas de sistema, análogas a las cinco diferentes operaciones de salida, que un proceso puede utilizar para recibir datos a través de un socket: *read* (*leer*), *ready* (*listo*), *recv* (*recibir*), *recvfrom* (*recibir de*) y *recvmsg (recibir mensaje)*. La operación de entrada convencional de UNIX, *read*, sólo se puede usar cuando un socket se conecta. Este tiene la forma:

read(descriptor, buffer, length)

donde el *descriptor* da el descriptor entero de un socket o el descriptor de archivo del que se puede leer los datos, el *buffer* especifica la dirección en memoria en la que se almacenan los datos y *length* especifica el número máximo de octetos que puedan leerse.

Una forma alternativa, *readv*, permite que quien llama utilice un estilo de interfaz de "scatter read" (lectura de diseminación) que coloque los datos que entran en ubicaciones no contiguas. *Readv* tiene la forma:

readv(descriptor, iovector, vectorlen)

El argumento *iovector* da la dirección de una estructura de tipo *iovec* (ver figura 20.3) que contiene una secuencia de apuntadores de bloques de memoria dentro de la que deberán almacenarse los datos que entran. El argumento *vectorlen* especifica el número de entradas en *iovector*.

Además de las operaciones de entrada convencionales, hay tres llamadas de sistema adicionales para la entrada de un mensaje a la red. Las llamadas de procesos *recv* son para recibir datos de un socket conectado. Tienen la forma:

recv(socket, buffer, length, flags)

El argumento *socket* especifica un descriptor de socket del que serán recibidos los datos. El argumento *buffer* especifica la dirección en memoria dentro de la que deberá colocarse el mensaje, y el argumento *length* especifica la longitud del área del búfer. Por último, el argumento *flags* permite que quien llame controle la recepción. Entre los valores posibles para el argumento *flags*, hay uno que permite que quien llame se adelante y extraiga una copia del siguiente mensaje que viene sin retirar el mensaje del socket.

La llamada de sistema *recvfrom* permite que quien llame especifique la entrada de un socket no conectado. Incluye argumentos adicionales que permiten a quien llame especificar dónde registrar la dirección de quien envía. La forma es:

recvfrom(socket, buffer, length, flags, fromaddr, addrlen)

Los dos argumentos adicionales, *fromaddr* y *addrlen*, son apuntadores de estructura de dirección de socket y un entero. El sistema operativo utiliza *fromaddr* para registrar las direcciones de quien envía el mensaje y utiliza *fromlen* para registrar la longitud de la dirección de quien lo envía. Obsérvese que la operación de salida *sendto*, que se analizó arriba, toma una dirección en forma exactamente igual a la que *recvfrom* genera. Así, enviar respuestas es fácil.

La última llamada de sistema utilizada para entrada, *recvmsg*, es análoga a la operación de salida *sendmsg*. *Recvmsg* opera como *recvfrom*, pero requiere de menos argumentos. Su forma es:

recvmsg(socket, messagestruct, flags)

donde el argumento *messagestruct* da la dirección de una estructura que sostiene la dirección para el mensaje que entra así como también las ubicaciones para la dirección de quien envía. La estruc-

tura producida por *recvmsg* es exactamente la misma que la estructura utilizada por *sendmsg*, lo que las hace que operen bien en conjunto.

20.11 Obtención de direcciones socket locales y remotas

Dijimos que los procesos que se crearon recientemente heredaron el conjunto de sockets abiertos del proceso que los creó. En algunas ocasiones, un proceso recién creado necesita determinar la dirección de destino a la que se conecta el socket. Un proceso puede también determinar la dirección local para un socket. Dos llamadas de sistema proporcionan información como: *getpeername* (*obtener nombre de pareja*) y *getsockname* (*obtener nombre de socket*) (sin importar sus nombres, ambos tratan con lo que pensamos como "direcciones").

Un proceso llama a *getpeername* para determinar la dirección de la pareja (es decir, el extremo remoto) al que se conecta el socket. Tiene la forma:

getpeername(socket, destaddr, addrlen)

El argumento *socket* especifica el socket para el que se desea la dirección. El argumento *destaddr* es un apuntador de estructura de tipo *sockaddr* (ver figura 20.1) que recibirá la dirección de socket. Por último, el argumento *addrlen* es un apuntador de entero que recibirá la longitud de la dirección. *Getpeername* sólo trabaja con sockets conectados.

La llamada de sistema *getsockname* devuelve la dirección local asociada con un socket. Tiene la forma:

getsockname(socket, localaddr, addrlen)

Como se esperaba, el argumento *socket* especifica el socket para el que se desea la dirección local. El argumento *localaddr* es un apuntador a una estructura de tipo *sockaddr* que contendrá la dirección y el argumento *addrlen* es un apuntador a entero que contendrá la longitud de dirección.

20.12 Obtención y definición de opciones de socket

Además de enlazar un socket a una dirección local o conectarla a una dirección de destino, surge la necesidad de un mecanismo que permita a los programas de aplicación controlar el socket. Por ejemplo, cuando se usan protocolos que emplean tiempos límite y retransmisión, el programa de aplicación podría obtener o designar los parámetros del tiempo límite. También, podría controlar la ubicación de espacio de búfer, determinar si el socket permite la transmisión de difusión o controlar el procesamiento de datos fuera de banda. En lugar de añadir nuevas llamadas de sistema para cada nueva operación de control, los diseñadores decidieron construir un mecanismo sencillo. El mecanismo tiene dos operaciones: *getsockopt* (*obtener opción socket*) y *setsockopt* (*definir opción socket*).

La llamada de sistema *getsockopt* permite que la aplicación solicite información sobre el socket. Quien llama especifica el socket, que es la opción de interés, y una ubicación en la que se almacena la información requerida. El sistema operativo examina su estructura de datos interna para el socket y transmite la información requerida a quien llama. La llamada tiene la forma:

getsockopt(socket, level, optionid, optionval, length)

El argumento *socket* especifica el socket para el que se necesita la información. El argumento *level* identifica si la operación se aplica al socket mismo o a los protocolos subyacentes que se están usando. El argumento *optionid* especifica una sola opción a la que la petición se aplica. El par de argumentos *optionval* y *length* especifican dos apuntadores. El primero nos da la dirección de un búfer dentro del cual el sistema coloca el valor requerido, y el segundo nos da la dirección de un entero dentro del que el sistema coloca la longitud del valor de opción.

La llamada de sistema *setsockopt* permite que un programa de aplicación configure una opción de socket mediante el conjunto de valores obtenidos con *getsockopt*. Quien llama especifica un socket para el que la opción deberá designarse, la opción a ser cambiada y un valor para tal opción. La llamada a *setsockopt* tiene la forma:

setsockopt(socket, level, optionid, optionval, length)

donde los argumentos como *getsockopt*, a excepción del argumento *length* contienen la longitud de la opción que se está transmitiendo al sistema. Quien llama debe proporcionar un valor legal para la opción así como también la longitud correcta para ese valor. Por supuesto, no todas las opciones se aplican a todos los sockets. La corrección y el significado de las peticiones individuales depende del estado actual del socket y de los protocolos subyacentes que se están usando.

20.13 Especificación de una longitud de cola para un servidor

Una de las opciones que se aplica a los sockets se utiliza con tanta frecuencia que se ha tenido que dedicar una llamada de sistema por separado a ella. Para comprender cómo surge, consideremos un servidor. El servidor crea un socket, lo enlaza a un puerto de protocolo bien conocido y espera las peticiones. Si el servidor emplea una entrega de flujo confiable, o si la computación de una respuesta se lleva cantidades no triviales de tiempo, puede suceder que una nueva petición llegue antes de que el servidor termine de responder a una petición anterior. Para evitar el rechazo de protocolos o la eliminación de las peticiones entrantes, el servidor debe indicar al software de protocolo subyacente que desea tener dichas peticiones en cola de espera hasta que haya tiempo para procesarlas.

La llamada de sistema *listen* (*escuchar*) permite que los servidores preparen un socket para las conexiones que vienen. En términos de protocolos subyacentes, *listen* pone al socket en modo pasivo listo para aceptar las conexiones. Cuando el servidor invoca a *listen*, también informa al sistema operativo que el software de protocolo deberá colocar en cola de espera las diversas peticiones simultáneas que llegaron al socket. La forma es:

listen(socket, qlength)

El argumento *socket* indica al descriptor de un socket que debe estar preparado para que lo use un servidor y el argumento *qlength* especifica la longitud de la cola de espera para ese socket. Después de la llamada, el sistema establece la cola de espera para que *qlength* solicite las conexiones. Si la cola de espera está llena cuando llega una petición, el sistema operativo rechaza la conexión descartando la petición. *Listen* se aplica sólo a los sockets que han seleccionado el servicio de entrega de flujo confiable.

20.14 Cómo acepta conexiones un servidor

Como hemos visto, un proceso de servidor utiliza las llamadas de sistema *socket, bind* y *listen* para crear un socket, enlazarlo a un puerto de protocolo bien conocido y especificar la longitud de cola para las peticiones de conexión. Obsérvese que la llamada a *bind* asocia al socket con un puerto de protocolo bien conocido, pero el socket no está conectado a un destino exterior específico. De hecho, el destino exterior debe especificar un *wildcard* (*comodín*), lo cual permite que el socket reciba peticiones de conexión de cualquier cliente.

Una vez que se ha establecido un socket, el servidor necesita esperar la conexión. Para hacerlo, se vale de la llamada de sistema *accept* (*aceptar*). Una llamada *accept* se bloquea hasta que la petición de conexión llega. Tiene la forma:

newsock = accept(socket, addr, addrlen)

El argumento *socket* especifica el descriptor del socket en el que va a esperar. El argumento *addr* es un apuntador a estructura de tipo *sockaddr* y *addrlen* es un apuntador a entero. Cuando llega una petición, el sistema llena un argumento *addr* con la dirección del cliente que ha colocado la petición y configura *addrlen* a la longitud de la dirección. Por último, el sistema crea un nuevo socket que tiene su destino conectado hacia el cliente que pide, y devuelve el nuevo descriptor de socket a quien llama. El socket original todavía tiene un comodín de destino externo y aún permanece abierto. De este modo, el servidor maestro puede continuar aceptando las peticiones adicionales en el socket original.

Cuando llega una petición de conexión la llamada a *accept* reaparece. El servidor puede manejar las peticiones de manera concurrente o iterativa. Desde un enfoque iterativo, el servidor mismo maneja la petición, cierra el nuevo socket y, después, llama a *accept* para obtener la siguiente petición de conexión. Desde el enfoque concurrente, después de que la llamada a *accept* reapareció, el servidor maestro crea un esclavo para manejar la petición (en terminología UNIX el proceso se bifurca en el proceso hijo para manejar la petición). El proceso esclavo hereda una copia del nuevo socket, de modo que puede proceder a servir a la petición. Cuando termina, el esclavo cierra el socket y termina. El proceso del servidor original (maestro) cierra su copia de un nuevo socket después de iniciar al esclavo. Después, llama a *accept* para obtener la siguiente petición de conexión.

El diseño concurrente para servidores podría parecer confuso debido a que los diversos procesos usarán el mismo número local de puerto de protocolo. La clave para comprender el mecanis-

mo descansa en la manera en que los protocolos subyacentes tratan a los puertos de protocolos. Recordemos que en el TCP, un par de extremos definen una conexión. De este modo, no importa cuántos procesos se usen en un número local de puerto de protocolo, mientras se conecten a diferentes destinos. En el caso de un servidor concurrente, hay un proceso por cliente y un proceso adicional por destino externo, lo cual permite que se conecte con cualquier localidad externa. Cada proceso restante tiene un destino externo específico. Cuando llega un segmento TCP, es enviado al socket conectado a la fuente de segmento. Si no existe tal socket, el segmento se mandará al socket que tenga un comodín para su destino externo. Además, como el socket con un comodín de destino externo como número de puerto no tiene una conexión abierta, sólo respetará a los segmentos TCP que pidan una nueva conexión.

20.15 Servidores que manejan varios servicios

La interfaz BSD de UNIX proporciona otra posibilidad interesante para el diseño de servidores porque permite que un solo proceso espere las conexiones de varios sockets. La llamada de sistema que hace que el diseño sea posible se denomina *select* (*seleccionar*) y se aplica en general a la E/S, no sólo para comunicación en los sockets. *Select* tiene la forma:

nready = select(ndesc, indesc, outdesc, excdesc, timeout)

En general, una llamada a *select* se bloquea y espera que uno de los conjuntos de descriptores de archivos se encuentre listo. El argumento *ndesc* especifica cuántos descriptores se deben examinar (los descriptores revisados son siempre *0* a *ndesc*-1). El argumento *indesc* es un apuntador a una máscara de bits que especifica los descriptores de archivo que se han de revisar para la entrada, el argumento *outdesc* es un apuntador a una máscara de bits que especifica los descriptores de archivo que se han de revisar para la salida, y el argumento *excdesc* es un apuntador a una máscara de bits que especifica los descriptores de archivo que se han de revisar en cuanto a condiciones de excepción. Por último, si el argumento *timeout* (*tiempo límite*) no es cero, será la dirección de un entero la que especifique cuánto esperará una conexión antes de regresarla a quien llama. Un valor de cero para el tiempo límite bloquea a quien llama hasta que el descriptor está listo. Debido a que el argumento *timeout* contiene la dirección del entero de tiempo límite y no al entero mismo, se puede pedir un proceso de demora cero, transmitiendo la dirección de un entero que contiene cero (es decir, un proceso puede ver si la E/S está lista).

Una llamada a *select* devuelve el número de descriptores del conjunto especificado que está listo para la E/S. También cambia las máscaras de bit especificadas por *indesc, outdesc* y *excdesc* para informar a la aplicación qué descriptores de archivo seleccionados están listos. De este modo, antes de llamar a *select*, quien llama debe activar los bits que correspondan a los descriptores que se han de revisar. Siguiendo la llamada, todos los bits que permanezcan con valor *1* corresponderán a un descriptor de archivo ya listo.

Para comunicarse con más de un socket a la vez mediante un proceso, primero se crean todos los sockets que se necesitan y después se usa *select* para determinar cuáles de ellos están listos primero para E/S. Una vez que encuentra un socket listo, el proceso se vale de los procedimientos de entrada o salida definidos arriba para la comunicación.

20.16 Obtención y especificación de nombres de anfitrión

El sistema operativo BSD de UNIX mantiene un nombre de anfitrión interno. Para las máquinas de Internet, el nombre interno suele elegirse como el nombre de dominio para la interfaz principal de la red de la máquina. La llamada de sistema *gethostname* (*conseguir nombre de anfitrión*) permite a los procesos del usuario que accedan al nombre de anfitrión, y la llamada de sistema *sethostname* (*definir nombre de anfitrión*) permite a los procesos de privilegio definir el nombre de anfitrión. *Gethostname* tiene la forma:

gethostname(name, length)

El argumento *name* (*nombre*) da la dirección de un arreglo de octetos en la que se ha de almacenar el nombre, y el argumento *length* (*longitud*) es un entero que especifica la longitud del arreglo *name*. Para definir el nombre de anfitrión un proceso privilegiado hace una llamada de la forma:

sethostname(name, length)

El argumento *name* da la dirección de un arreglo en la que se almacena el nombre, y el argumento *length* es un entero que da la longitud del arreglo del nombre.

20.17 Obtención y especificación del dominio de anfitrión interno

El sistema operativo mantiene una cadena que especifica el nombre de dominio al que pertenece la máquina. Cuando una localidad obtiene autoridad por parte de un espacio de nombre de dominio, implanta una cadena que identifica su porción de espacio y usa una cadena como el nombre del dominio. Por ejemplo, las máquinas del dominio

cs.purdue.edu

tienen nombres tomados de la leyenda del rey Arturo. De este modo, uno puede encontrar máquinas llamadas *merlín, arturo, guenevere* y *lancelot*. El dominio en sí se llama *camelot*, de manera que el sistema operativo en cada anfitrión del grupo debe estar informado de que reside en el dominio *camelot*. Para hacerlo, un proceso privilegiado utiliza la llamada de sistema *setdomainname*, que tiene la forma:

setdomainname(name, length)

El argumento *name* da la dirección de un arreglo de octetos que contiene el nombre de un dominio y el argumento *length* es un entero que da la longitud del nombre.

Los procesos del usuario llaman a *getdomainname* para recuperar el nombre del dominio del sistema. Getdomainname tiene la forma:

getdomainname(name, length)

donde el argumento *name* especifica la dirección de un arreglo en el que el nombre debe estar almacenado, y el argumento *length* es un entero que especifica la longitud del arreglo.

20.18 Llamadas de la biblioteca de red BSD de UNIX

Además de las llamadas de sistema descritas arriba, BSD de UNIX ofrece un conjunto de rutinas de biblioteca que ejecutan funciones útiles, relacionadas con el trabajo en red. En la figura 20.5 se ilustra la diferencia entre las llamadas de sistema y las rutinas de biblioteca. Las llamadas de sistema transmiten el control al sistema operativo, mientras que las rutinas de biblioteca son como otros procedimientos que el programador enlaza dentro de un programa.

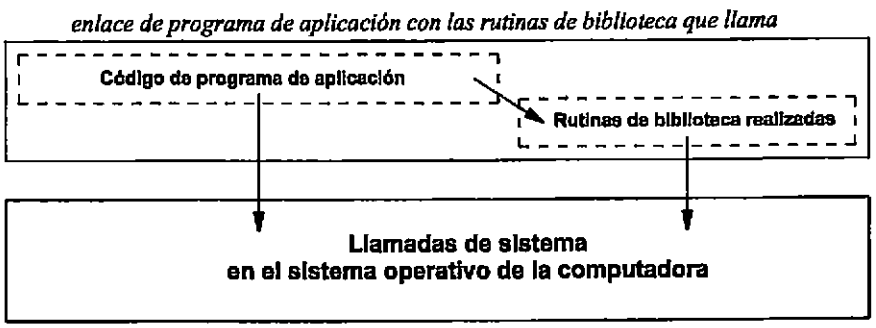


Figura 20.5 Diferencia entre la biblioteca de rutinas, que están enlazadas dentro de un programa de aplicación y las llamadas de sistema, que son parte de un sistema operativo. Un programa puede llamar a ambos; una biblioteca de rutina puede llamar a otras bibliotecas de rutinas o a llamadas de sistema.

Muchas de las rutinas de biblioteca de BSD de UNIX proporcionan servicios de base de datos que permiten que un proceso determine los nombres de las máquinas y los servicios de red, los números de puertos de protocolos y demás información relacionada. Por ejemplo, un conjunto de rutinas de biblioteca proporciona acceso para la base de datos de los servicios de red. Pensamos en las entradas de la base de datos de los servicios como triadas (formadas por tres partes), donde cada triada contiene el nombre (legible para las personas) de un servicio de red de trabajo, el protocolo que soporta el servicio y un número de puerto de protocolo para el servicio. Existen rutinas de biblioteca que permiten que un proceso obtenga información de una entrada dada.

En la siguientes secciones se examinan grupos de rutinas de biblioteca, se explica sus propósitos y se proporciona información acerca de cómo pueden usarse. Como veremos, los conjuntos de rutinas de biblioteca que proporcionan acceso a bases de datos secuenciales siguen un patrón. Cada conjunto permite la aplicación para: establecer una conexión hacia la base de datos, obtener entradas una a la vez y cerrar la conexión. Las rutinas empleadas para estas tres operaciones se de-

nominan *setXent, getXent* y *endXent*, donde *X* es el nombre de la base de datos. Por ejemplo, las rutinas de biblioteca para la base de datos anfitrión se llaman *sethostent, gethostent* y *endhostent*. En las secciones en las que se describe estas rutinas, se resume las llamadas sin repetir los detalles de su uso.

20.19 Rutinas de conversión del orden de red de los octetos

Recordemos que las máquinas se diferencian por la forma en que almacenan las cantidades de enteros y que los protocolos TPC/IP definen un estándar independiente de máquina para orden de octetos. BSD de UNIX proporciona cuatro funciones de biblioteca que convierten el orden de octetos de la máquina local y el orden estándar de octetos de red. Para hacer que los programas sean portátiles deben escribirse para que llamen a rutinas de conversión cada vez que se copia un valor entero de la máquina local a un paquete de red de trabajo o cuando copian un valor de un paquete de red de trabajo a una máquina local.

Las cuatro rutinas de conversión son funciones que toman un valor como un argumento y devuelven un nuevo valor con los octetos reordenados. Por ejemplo, para convertir un entero pequeño (2 octetos) de un orden de octetos de red a un orden de octetos de anfitrión local, un programador llama a *ntohs* (*network to host short*). El formato es:

localshort = ntohs(netshort)

El argumento *netshort* es un entero de 2 octetos (16 bits) en el orden de octetos de la red de trabajo estándar y el resultado, *localshort*, es un orden de octetos de anfitrión local.

UNIX llama *longs* a los enteros de 4 octetos (32 bits). La función *ntohl* (network to host long) convierte los enteros largos de 4 octetos de un orden de octetos de red estándar en un orden de octetos del anfitrión local. Los programas invocan a *ntohl* como función dando un entero largo de un orden de octetos de red como argumento:

locallong = ntohl(netlong)

Dos funciones análogas son las que le permiten al programador convertir el orden de octetos del anfitrión local al orden de octetos de la red. La función *htons* convierte un entero (corto) de 2 octetos del orden de octetos del anfitrión local en un entero de 2 octetos de un orden de octetos de red estándar. Los programas invocan a *htons* como una función:

netshort = htons(localshort)

La rutina final de conversión, *htonl*, convierte a los enteros grandes de orden de octetos del anfitrión en orden de octetos de red. Como los otros, *htonl* es una función:

netlong = htonl(locallong)

Es obvio que las rutinas de conversión preservan las siguientes relaciones matemáticas:

netshort = htons(ntohs(netshort))

y

localshort = ntohs(htons(localshort))

Se sostienen relaciones similares para las rutinas de conversión de enteros grandes.

20.20 Rutinas de manipulación de direcciones IP

Debido a que muchos programas traducen de direcciones IP de 32 bits y la notación decimal con puntos correspondiente, la biblioteca BSD de UNIX incluye rutinas de herramientas que realizan la traducción. Los procedimientos *inet_addr* y *inet_network* traducen de formato decimal con puntos a direcciones IP de 32 bits en orden de octetos de la red. *Inet_addr* forma una dirección IP de anfitrión de 32 bits; *inet_network* forma la dirección de red con ceros para la parte anfitriona. Tienen la forma:

address = inet_addr(string)

y

address = inet_network(string)

donde el argumento *string* da la dirección de una cadena ASCII que contiene el número expresado en formato decimal con puntos. La forma decimal con puntos puede tener de 1 a 4 segmentos de dígitos separados por puntos. Si aparecen los cuatro, cada uno corresponde a un solo octeto del entero de 32 bits resultante. Si aparecen menos de 4, el último segmento se expande para llenar los octetos restantes.

El procedimiento *inet_ntoa* ejecuta lo inverso a *inet_addr*, pues transforman un entero de 32 bits en una cadena ASCII en formato decimal con puntos. Esto tiene la forma:

str = inet_ntoa(internetaddr)

donde el argumento *internetaddr* es una dirección IP de 32 bits en el orden de octetos de la red y *str* es la dirección de la versión ASCII resultante.

A menudo, los programas que manipulan direcciones IP deben combinar una dirección de red de trabajo con la dirección local de un anfitrión en una red de trabajo. El procedimiento *inet_makeaddr* realiza dicha combinación. Tiene la forma:

internetaddr = inet_makeaddr(net,local)

El argumento *net* es una dirección IP de red de 32 bits en el orden de octetos del anfitrión y el argumento *local* es el entero que representa una dirección de anfitrión local en la red, también en el orden de octetos para el anfitrión local.

Los procedimientos *inet-netof* y *inet-lnaof* proporcionan lo inverso a *inet_makeaddr* pues separan la red y las porciones locales de una dirección IP. Tienen la forma:

net = inet_netof(internetaddr)

y

local = inet_lnaof(internetaddr)

donde el argumento *internetaddr* es una dirección IP de 32 bits en el orden de octetos de la red y los resultados se devuelven en el orden de octetos del anfitrión.

20.21 Acceso al sistema de nomenclatura de dominios ANS[5]

Hay un conjunto de cinco procedimientos de biblioteca que comprende la interfaz BSD de UNIX para el sistema de nombre de dominio TCP/IP. Los programas de aplicación que llaman a estas rutinas se convierten en clientes de un sistema de nombre de dominio, mandando una o más peticiones de servicio y recibiendo respuestas.

La idea general es que un programa hace una solicitud, la manda a un servidor y espera una respuesta. Como existen muchas opciones, las rutinas tienen sólo unos cuantos parámetros básicos y utilizan una estructura global: *res*, para sostener a otras. Por ejemplo, un campo en la estructura *res* permite la depuración de mensajes mientras que otro controla si el código usa UDP o TCP para las solicitudes. La mayor parte de los campos en *res* comienza con datos por omisión razonables, de manera que las rutinas se pueden utilizar sin cambiar la estructura *res*.

Un programa llama a *res_init* antes de utilizar otros procedimientos. La llamada no toma argumentos:

res_init()

Res_init lee un archivo que contiene información como el nombre de la máquina que corre el servidor de nombre de dominio y almacena los resultados en la estructura global *res*.

El procedimiento *res_mkquery* forma una averiguación de nombre de dominio y la coloca en un búfer en la memoria. La forma de la llamada es:

res_mkquery(op, dname, class, type, data, datalen, newrr, buffer, buflen)

Los primeros siete argumentos corresponden directamente a los campos de la solicitud de nombre de dominio. El argumento *op* especifica la operación requerida, *dname* da la dirección de un arreglo de caracteres que contiene el nombre de dominio, *class* es un entero que da la clase de solicitud, *type* es un entero que da el tipo de solicitud, *data* da la dirección de un arreglo de datos que han de ser incluidos en la solicitud y *datalen* es un entero que da la longitud de los datos. Además de los procedimientos de biblioteca, UNIX proporciona programas de aplicación con definiciones de constantes simbólicas para valores importantes. De este modo, los programadores pueden utilizar el sistema de nombre de dominio sin comprender los detalles del protocolo. Los últimos dos argumentos, *buffer* y *buflen*, especifican la dirección de un área dentro de la que deberá colocarse

[5] En el capítulo 22, se considera detalladamente el Sistema de Nombre de Dominio.

la solicitud y la longitud del entero del área de búfer, respectivamente. Por último, en la implantación actual, el argumento *newrr* no se utiliza.

Una vez que el programa ha formado una búsqueda, llama a *res_send* para enviar a un nombre de servidor y obtener una respuesta. La forma es:

res_send(buffer, buflen, answer, anslen)

El argumento *buffer* es un apuntador a memoria que guarda el mensaje que se ha de enviar (presumiblemente, la aplicación llama al procedimiento *res_mkquery* para formar el mensaje). El argumento *buflen* es un entero que especifica la longitud. El argumento *answer* da la dirección en memoria dentro de la que se deberá escribir una respuesta, y el argumento entero *anslen* especifica la longitud de un área de respuesta.

Además de las rutinas que hacen y envían solicitudes, la biblioteca BSD de UNIX contiene dos rutinas que traducen nombres de dominio de ASCII convencional y del formato comprimido utilizado en las solicitudes. El procedimiento *dn_expand* expande un nombre de dominio comprimido convirtiéndolo en una versión completa en ASCII. Tiene la forma:

dn_expand(msg, eom, compressed, full, fullen)

El argumento *msg* da la dirección de un mensaje de nombre de dominio que contiene el nombre que se ha de expandir, con *eom* especificando el límite de fin de mensaje más allá del cual la expansión no puede ir. El argumento *compressed* es un apuntador para el primer octeto del nombre comprimido. El argumento *full* es un apuntador para un arreglo dentro del cual el nombre expandido deberá estar escrito, y el argumento *fullen* es un entero que especifica la longitud del arreglo.

Generar un nombre comprimido es más complejo que expandir un nombre comprimido porque la compresión comprende la eliminación de los sufijos comunes. Cuando se comprimen nombres, el cliente debe mantener un registro de los sufijos que han aparecido previamente. El procedimiento *dn_comp* comprime un nombre de dominio completo comparando para ello los sufijos con una lista de sufijos previamente utilizados y eliminando los sufijos más grandes posibles. Una llamada tiene la forma:

dn_comp(full, compressed, cmprlen, prevptrs, lastptr)

El argumento *full* da la dirección de un nombre de dominio completo. El argumento *compressed* apunta a un arreglo de octetos que mantendrá un nombre comprimido, con el argumento *cmprlen* especificando la longitud del arreglo. El argumento *prevptrs* es la dirección de un arreglo de apuntadores para los sufijos previamente comprimidos, con *lastptr* que apunta al extremo del arreglo. Normalmente, *dn_comp* comprime el nombre y actualiza *prevptrs* si se ha utilizado un nuevo sufijo.

El procedimiento *dn_comp* puede también usarse para traducir un nombre de dominio de ASCII a la forma interna sin compresión (es decir, sin quitar sufijos). Para hacerlo, el proceso invoca a *dn_comp* con el argumento *prevptrs* definido como *NULL* (es decir, cero).

20.22 Obtención de información sobre anfitriones

Existen procedimientos de biblioteca que permiten que un proceso recupere información de un anfitrión dado, ya sea que se tenga un nombre de dominio o una dirección IP. Cuando se emplea en una máquina que tiene acceso a un servidor de nombre de dominio, los procedimientos de la biblioteca realizan el proceso para el cliente del sistema de nombre de dominio enviando una petición a un servidor y esperando la respuesta. Cuando se utiliza en sistemas que no tienen acceso al sistema de nombre de dominio (es decir, un anfitrión que no está en Internet), las rutinas obtienen la información deseada de una base de datos que se mantiene en almacenamiento secundario.

La función *gethostbyname* (*obtener anfitrión por nombre*) toma un nombre de dominio y devuelve un apuntador a una estructura de información para ese anfitrión. Una llamada toma la forma:

ptr = gethostbyname(namestr)

El argumento *namestr* es un apuntador a una cadena de caracteres que contiene un nombre de dominio para el anfitrión. El valor devuelto, *ptr*, apunta a una estructura que contiene la siguiente información: el nombre oficial del anfitrión, una lista de alias que se han registrado para el anfitrión, el tipo de dirección del anfitrión (es decir, si la dirección es IP), la longitud de la dirección y una lista de una o más direcciones del anfitrión. Se pueden encontrar más detalles en el Manual del Programador de UNIX.

La función *gethostbyaddr* produce la misma información que *gethostbyname*. La diferencia entre las dos es que *gethostbyaddr* acepta la dirección de un anfitrión como un argumento:

ptr = gethostbhyaddr(addr, len, type)

El argumento *addr* es un apuntador a una secuencia de octetos que contiene una dirección de anfitrión. El argumento *len* es un entero que da la longitud de la dirección y el argumento *type* es un entero que especifica el tipo de la dirección (es decir, que es una dirección IP).

Como se mencionó al principio, los procedimientos *sethostent, gethostent* y *endhostent* proporcionan un acceso secuencial a la base de datos anfitrión.

20.23 Obtención de información sobre redes

Los anfitriones que utilizan BSD de UNIX o bien emplean el sistema de nombre de dominio o mantienen una base de datos sencilla de redes en su red de redes. Las rutinas de la biblioteca de la red incluyen cinco rutinas que permiten que un proceso acceda a la base de datos de la red. El procedimiento *getnetbyname* obtiene y da formato al contenido de una entrada de la base de datos una vez dado el nombre de dominio de una red de trabajo. Una llamada tiene la forma:

ptr = getnetbyname(name)

donde el argumento *name* es un apuntador a una cadena que contiene el nombre de la red de trabajo para la que se desea la información. El valor devuelto es un apuntador a una estructura que con-

tiene los campos para el nombre oficial de la red de trabajo, una lista de los alias registrados, una dirección de tipo entero y una dirección de red de 32 bits (es decir, una dirección IP con la porción de anfitrión puesta en cero).

Un proceso llama a la rutina de biblioteca *getnetbyaddr* cuando necesita buscar información acerca de una red de trabajo una vez dada su dirección. La llamada tiene la forma:

ptr = getnetbyaddr(netaddr, addrtype)

El argumento *netaddr* es una dirección de red de trabajo de 32 bits, y el argumento *addrtype* es un entero que especifica el tipo de *netaddr*. Los procedimientos *setnetent, getnetent* y *endnetent* proporcionan un acceso secuencial a una base de datos de la red.

20.24 Obtención de información sobre protocolos

Hay cinco rutinas de biblioteca que proporcionan el acceso a la base de datos de protocolos disponibles en una máquina. Cada protocolo tiene un nombre oficial, alias registrados y un número de protocolo oficial. El procedimiento *getprotobyname* permite que quien llama obtenga información acerca de un protocolo dando su nombre:

ptr = getprotobyname (name)

El argumento *name* es un apuntador a una cadena ASCII que contiene el nombre del protocolo para el que se desea la información. La función devuelve un apuntador a una estructura que tiene campos para el nombre oficial de protocolo, una lista de alias y un valor entero único asignado al protocolo.

El procedimiento *getprotobynumber* permite que un proceso busque la información del protocolo utilizando el número de protocolo como una clave:

ptr = getprotobynumber(number)

Finalmente, los procedimientos *getprotoent, setprotoente* y *endprotoent* proporcionan un acceso secuencial a la base de datos de protocolos.

20.25 Obtención de información sobre servicios de red

Recordemos que, en los capítulos 12 y 13, se mencionó que algunos números de puerto de protocolo de UDP y TCP están reservados para los servicios bien conocidos. Por ejemplo, el puerto *43* del TCP está reservado para el servicio *whois*. *Whois* permite a un cliente en una máquina ponerse en contacto con un servidor en otra y obtener información acerca de un usuario que tiene una cuenta en la máquina del servidor. La entrada para *whois* en la base de datos de servicios especifica el nombre de servicio, *whois*, el protocolo, *TCP*, y el número de puerto de protocolo *43*. Existen cin-

co rutinas de biblioteca que obtienen información acerca de los servicios y los puertos de protocolo que usan.

El procedimiento *getservbyname* transforma un servicio nombrado en un número de puerto:

ptr = getservbyname(name, proto)

El argumento *name* especifica la dirección de una cadena que contiene el nombre del servicio deseado, un argumento entero, *proto*, especifica el protocolo con el que el servicio se ha de utilizar. Por lo general, los protocolos están limitados a TCP y UDP. El valor devuelto es una apuntador hacia una estructura que contiene campos para el nombre del servicio, una lista de alias, una identificación del protocolo con el que se usa el servicio y un número entero de puerto de protocolo asignado para ese servicio.

El procedimiento *getservbyport* permite a quien llama obtener una entrada de la base de datos de servicios con sólo dar el número de puerto asignado para ello. Una llamada tiene la forma:

ptr = getservbyport(port, proto)

El argumento *port* es el número entero de puerto de protocolo asignado al servicio y el argumento *proto* especifica el protocolo para el que se desea el servicio. Al igual que con las otras bases de datos, un proceso puede accesar a la base de datos de servicios de manera secuencial utilizando *setservent, getservent* y *endservent.*

20.26 Ejemplo de un cliente

El siguiente ejemplo del programa C ilustra la interfaz entre el sistema operativo BSD de UNIX y TCP/IP. Se trata de una implantación sencilla de un *whois* cliente y servidor. Como se define en RFC 954, el servicio *whois* permite que un cliente en una máquina obtenga información acerca de un usuario en un sistema remoto. En esta implantación, el cliente es un programa de aplicación que un usuario invoca mediante dos argumentos: el nombre de una máquina remota y el nombre del usuario en esta máquina acerca de quien se desea la información. El cliente llama a *gethostbyname* para transformar el nombre de la máquina remota en una dirección IP y llama a *getservbyname* para encontrar el puerto bien conocido para el servicio *whois*. Una vez que ha transformado los nombres de anfitrión y servicio, el cliente crea un socket, especificando que el socket usará una cadena de entrega confiable (es decir, TCP). El cliente entonces enlaza el socket con el puerto de protocolo *whois* en la máquina de destino especificada.

```
/* whoisclient.c - main */

#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h
#include <netinet/in.h>
#include <netdb.h>
/*-----------------------------------------------------------------
```

```
 * Programa:      whoisclient
 *
 * Propósito:     Programa de aplicación de UNIX que se convierte en
 *                cliente para el servicio "whois" de Internet.
 *
 * Usa:           whois hostname username
 *
 * Autor:         Barry Shein, Boston University
 *
 * Fecha:         Enero de 1987
 *
 *-----------------------------------------------------------------
 */
main(argc, argv)
int argc;          /* declaraciones de argumento estándar de UNIX*/
char *argv[];
{
  int s;                      /* descriptor de socket*/
  int len;                    /* longitud de datos recibidos*/
  struct sockaddr_in sa;      /* estruc. direc. socket Internet*/
  struct hosten *hp;          /* resultado búsqueda nombre anfi.*/
  struct servent *sp;         /* resultado servicio búsqueda*/
  char buf[BUFSIZ+1];         /* búfer que lee inf. whois*/
  char *myname;               /* apuntador a nombre este prog.*/
  char *host;                 /* apuntador a nombre afi. remot.*/
  char *user;                 /* apuntador a nombre usuario rem.*/

  myname = argv[0];
  /*
   * Revisar que haya dos argumentos de línea de comandos
   */
  if(argc !=3) {
     fprintf(stderr, "usage: %s host username\n", myname);
     exit(1);
  }
  host = argv[1];
  user = argv[2];
  /*
   * Búsqueda del nombre de anfitrión especificado
   */
  if((hp = gethostbyname(host)) == NULL) {
     fprint(stderr,"%s: %s: no such host?\n", myname, host);
     exit(1);
  }
  /*
   • Poner la dirección del anfitrión en el tipo de anfitrión dentro de
     la estructura socket
   */
  bcopy((char *)hp-h_addr, (char *) &sa.sin_addr, hp-h_length);
  sa.sin_family = hp-h_addrtype;
```

```
/*
 * Búsqueda del número de socket para el servicio WHOIS
 */
if((sp = getservbhyname("whois","tcp")) == NULL) {
   fprintf(stderr, "%s: No existe servicio whois en este anfitrión\n",
   myname);
   exit(1);
}
/*
 * Poner el número de socket whois en la estructura de socket.
 */
sa.sin_port = sp-s_port;
/*
 * Ubicar un socket abierto
 */
if((s = socket(hp-h_addrtype, SOCK_STREAM, 0))  0) {
   perror("socket");
   exit(1);
}
/*
 * Conectar al servidor remoto
 */
if(connect(s, &sa, sizeof sa)  0) {
   perror("connect");
   exit(1);
}
/*
 * Enviar la petición
 */
if(write(s, user, strlen(user)) 1= strlen(user)) {
   fprintf(stderr, "%s: error de escritura\n", myname);
                exit(1);
}
/*
 * Leer la respuesta y ponerla a la salida del usuario
 */
while( (len = read(s, buf, BUFSIZ))  0)
   write(1, buf, len);
close(s);
exit(0);
}
```

20.27 Ejemplo de un servidor

El servidor de ejemplo es sólo un poco más complejo que el de cliente. El servidor escucha en el puerto "whois" bien conocido y devuelve la información requerida en respuesta a una petición de

cualquier cliente. La información se toma del archivo de claves de acceso de UNIX de la máquina del servidor.

```
/* whoisserver.c - main */

#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <pwd.h>

/*-----------------------------------------------------------------
 * Programa:      whoisserver
 *
 * Propósito:     Programa de aplicación de UNIX que actúa como un
 *                servidor para el servicio "whois" de la máquina
 *                local. Escucha en el puerto WHOIS bien conocido (43)
 *                y contesta las solicitudes de los clientes. Este
 *                programa requiere un privilegio de super usuario para
 *                correrse.
 *
 * Usa:           whois hostname username
 *
 * Autor:         Barry Shein, Boston University
 *
 * Fecha:                    Enero de 1987
 *
 *-----------------------------------------------------------------
*/

#define BACKLOG        5       /* # peticiones dispuestas para poner
                               en cola*/
#define MAXHOSTNAME    32      /* longitud máxima tolerable del nombre
                               de anfitrión*/

main(argc, argv)
int argc;                      /* declaraciones de argumento estándar
                               de UNIX*/
char *argv[];
{
  int s, t;                    /* descriptores de socket*/
  int I;                       /* entero de propósito general*/
  struct sockaddr_in sa, isa;/* estructura de la dirección de socket
                               de Internet*/
  struct hostent *hp;          /* resultado de la búsqueda de nombre
                               de anfitrión*/
  char *myname;                /* apuntador al nombre de este programa*/
  struct servent *sp;          /* resultado del servicio de búsqueda*/
```

```
char localhost[ MAXHOSTNAME+1] ;/* nombre del anfitrión local como
                                    cadena de caracteres*/

myname = argv[ 0] ;
/*
 • Búsqueda de la entrada al servicio WHOIS
 */
if((sp = getservbyname("whois","tcp")) == NULL)  {
   fprintf(stderr, "%s: No existe servicio whois en este anfitrión\n",
   myname);
   exit(1);
}
/*
 * Obtener nuestra propia información de anfitrión
 */
gethostname(localhost, MAXHOSTNAME);
if((hp = gethostbyname(localhost)) == NULL) {
   fprintf(stderr, "%s: no se puede obtener información del anfitrión
   local?\n", myname);
   exit(1);
}
/*
 * Poner el número de socket WHOIS y nuestra información de
 * dirección dentro de la estructura de socket
 */
sa.sin_port = sp-s_port;
bcopy((char *)hp-h_addr, (char *)&sa.sin_addr, hp-_length);
sa.sin_family = hp-h_addrtype;
/*
 * Ubicar un socket abierto para las conexiones que vienen
 */
si((s = socket(hp-h_addrtype, SOCK_STREAM, 0))  0) {
   perror("socket");
  exit(1);
}
/*
 * Enlazar el socket al puerto de servicio
 * de modo que escuchemos las conexiones entrantes
 */
if(bind(s, &sa, sizeof sa)  0) {
   perror("bind");
   exit(1);
}
/*
 * Definir las conexiones máximas que dejaremos atrás
 */
listen(s, BACKLOG);
/*
 * Caer en un ciclo infinito de espera de las nuevas conexiones
 */
```

```
  while(1) {
     1 = sizeof isa;
     /*
      * Esperamos la aceptación () mientras se espera a nuevos clientes
      */
     if((t = accept(s, &isa, &I))  0) {
       perror("accept");
       exit(1);
     }
     whois(t);                /* ejecuta el servicio WHOIS actual */
     cerrar(t);
  }
  }
  /*
   * Obtener la petición WHOIS del anfitrión remoto y dar formato
     a la respuesta.
   */
  whois(sock)
  int sock;
  {
  struct passwd *p;
  char buf[ BUFSIZ+1] ;
     int I;

  /*
   * Obtener una petición de línea
   */
  if( (I = read(sock, buf, BUFSIZ)) < = 0)
     return;
  buf[ i] = '\0' ;             /* Termina estado nulo */
  /*
   *Búsqueda del usuario requerido y formato a respuesta
   */
  if((p = getpwnam(buf)) == NULL)
     strcpy(buf,"Wser not found\n");
  else
     sprintf(buf, "%s: %s\n", p-pw_name, p-pw_gecos);
  /*
   * Se devuelve respuesta
   */
  write(sock, buf, strien(buf));
  return;
}
```

20.28 Resumen

Debido a que el software de protocolo TCP/IP reside dentro de un sistema operativo, la interfaz exacta entre un programa de aplicación y los protocolos TCP/IP dependen de los detalles del sistema operativo; no está especificada por el estándar de protocolo de TCP/IP. Examinamos la interfaz socket originalmente diseñada para BSD de UNIX y vimos que adoptó el paradigma open-read-write-close de UNIX. Para utilizar el TCP, un programa debe crear un socket, enlazar direcciones a éste, aceptar las conexiones que llegan y después comunicarse por medio de las primitivas *read* o *write*. Por último, cuando se termina de usar un socket, el programa debe cerrarlo. Además de la abstracción del socket y las llamadas de sistema que operan en los sockets, BSD de UNIX incluye rutinas de biblioteca que ayudan a los programadores a crear y manipular direcciones IP, convertir enteros entre el formato de la máquina local y el orden de octetos del estándar de red, y buscar información como direcciones de redes.

La interfaz socket se ha convertido en algo muy popular y es soportada ampliamente por muchos vendedores. Los vendedores que no ofrecen facilidades de socket en sus sistemas operativos a menudo proporcionan una biblioteca socket que les facilita a los programadores escribir aplicaciones mediante las llamadas de socket aun cuando el sistema operativo subyacente utilice un conjunto diferente de llamadas de sistema.

PARA CONOCER MÁS

Se puede encontrar información detallada sobre las llamadas del sistema socket en la sección 2 del *UNIX Programmer's Manual* (*Manual del programador de Unix*), la cual contiene una descripción de cada llamada del sistema UNIX; en la sección 3 aparece una descripción de cada procedimiento de biblioteca. UNIX también proporciona copias en línea de las páginas del manual vía el comando *man*. Leffler, McKusick, Karels y Quarterman (1989) exploran con mayor detalle el sistema UNIX.

Los vendedores de sistemas operativos a menudo proporcionan bibliotecas de procedimientos que emulan a los sockets en sus sistemas. Además, Microsoft y otras compañías han cooperado para definir la interfaz *Winsock* que permite a los programas de aplicación que hacen llamadas socket que corran con Microsoft Windows; varios vendedores ofrecen productos compatibles con Winsock. Si requiere mayores detalles, consulte los manuales de programación de los vendedores.

La versión socket del volumen 3 de esta obra describe cómo están estructurados los programas de cliente y servidor y cómo utilizan la interfaz socket. En el volumen 3 podemos encontrar la versión TLI que nos proporciona una introducción a la *Transport Layer Interface (Interfaz de Capa de Transporte)*, que es una alternativa a los sockets utilizados en el Sistema V de UNIX.

EJERCICIOS

20.1 Trate de correr el ejemplo de *whois* cliente y servidor en su sistema local.

20.2 Construya un servidor sencillo que acepte diversas conexiones concurrentes (para probarlo haga que el proceso maneje una conexión de impresión de un pequeño mensaje, demore un tiempo aleatorio, imprima otro mensaje y salga).

20.3 ¿Cuándo es importante la llamada *listen* (*escuchar*)?

20.4 ¿Qué procedimientos proporciona su sistema local para acceder al sistema de nombre de dominio?

20.5 Diseñe un servidor que utilice un proceso UNIX sencillo, pero que maneje diversas conexiones concurrentes TCP. Sugerencia: piensa en *select* (*seleccionar*) (*poll* en SISTEMA V).

20.6 Infórmese sobre la Interfaz de Biblioteca de Transporte (TLI) del Sistema V de AT&T y compárela con la interfaz socket. ¿Cuáles son las principales diferencias conceptuales?

20.7 Cada sistema operativo limita el número de sockets que un programa dado puede usar en cualquier momento. ¿Cuántos sockets puede crear un programa en su sistema local?

20.8 El mecanismo descriptor de socket/archivo y las operaciones asociadas *read* (*leer*) y *write* (*escribir*) se pueden considerar como una forma de objeto orientada al diseño. Explique por qué.

20.9 Considere un diseño de interfaz alternativo que proporcione una interfaz para todas las capas del software de protocolo (es decir, que el sistema permita a un programa de aplicación enviar y recibir paquetes sin procesar, sin usar el IP, o enviar y recibir datagramas IP sin emplear el UDP o el TCP). ¿Cuáles son las ventajas y las desventajas de tener dicha interfaz?

20.10 Un cliente y un servidor pueden correr en la misma computadora y valerse de un socket TCP para comunicarse. Explique de qué manera es posible construir un cliente-servidor que pueda comunicarse con una sola máquina sin enterarse de la dirección IP del anfitrión.

20.11 Experimente con el servidor de muestra de este capítulo para ver si puede generar conexiones TCP que sean lo suficientemente rápidas como para exceder el trabajo atrasado que especifica el servidor. ¿Espera que las peticiones de conexión entrantes excedan el trabajo atrasado más rápido si el servidor opera en una computadora que tiene *1* procesador o en una computadora que tiene *5* procesadores? Explíquelo.

21

Arranque y autoconfiguración (BOOTP, DHCP)

21.1 Introducción

Este capítulo muestra cómo se utiliza el paradigma cliente-servidor para el proceso de arranque. Cada computadora conectada a una red de redes TCP/IP necesita saber su dirección IP antes de que pueda enviar o recibir datagramas. Además, una computadora requiere información adicional como la dirección de un ruteador, la mascara de red en uso y la dirección de un servidor de nombres. En el capítulo 6, se describió cómo puede utilizar una computadora el protocolo RARP en el inicio de un sistema para determinar su dirección IP. En este capítulo, se analiza una alternativa: dos protocolos del proceso de arranque muy relacionados que permiten a un anfitrión determinar su dirección IP sin utilizar RARP. Sorprendentemente, el cliente y el servidor se comunican mediante el UDP, el protocolo de datagrama usuario descrito en el capítulo 12.

Lo que es sorprendente del procedimiento de arranque es que el UDP depende del protocolo IP para transferir mensajes, y podría parecer imposible que una computadora pudiera utilizar el UDP para localizar una dirección IP que utiliza ésta cuando se comunica. Examinaremos los protocolos que nos ayudarán a entender cómo puede utilizar una computadora la dirección IP especial mencionada en el capítulo 4 y la flexibilidad del mecanismo de transporte UDP/IP. También veremos de qué manera asigna un servidor una dirección IP a una computadora en forma automática. Esta asignación es especialmente importante en ambientes que permiten conexiones de red de redes temporales en los que las computadoras se transfieren de una red a otra (por ejemplo, un empleado con una computadora portátil puede moverse de una localidad hacia otra en una compañía).

21.2 La necesidad de una alternativa a RARP

El capítulo 6 presenta el problema de las computadoras sin disco durante el arranque de sistema. Estas máquinas por lo general contienen un programa de arranque en un medio de almacenamiento no volátil (por ejemplo, en ROM). Para minimizar costos y conservar partes intercambiables, el vendedor coloca exactamente el mismo protocolo en todas las máquinas. Como las computadoras con diferentes direcciones IP corren el mismo programa de arranque, el código no puede contener una dirección IP. Así, una máquina sin disco debe obtener su dirección IP desde otra fuente. De hecho, una computadora sin disco necesita conocer mucho más que su dirección IP. En general, la memoria ROM sólo contiene un pequeño programa de arranque, de manera que las computadoras sin disco deben también obtener una imagen de memoria inicial para su ejecución. Además, cada máquina sin disco puede determinar la dirección de un servidor de archivos en el que pueda almacenar y recuperar datos, así como la dirección del ruteador IP más cercano.

El protocolo RARP, descrito en el capítulo 6, tiene tres inconvenientes. En primer lugar, dado que RARP opera en un nivel bajo, su uso requiere de un acceso directo hacia el hardware de red. Así pues, puede resultar difícil o imposible para un programador de aplicaciones construir un servidor. En segundo lugar, aun cuando RARP necesita un intercambio de paquetes entre una máquina cliente y una computadora que responda a las solicitudes, la réplica contiene sólo una pequeña parte de información: la dirección IP del cliente de 4 octetos. Estos inconvenientes son especialmente molestos en redes como Ethernet que imponen un tamaño de paquete mínimo, ya que la información adicional puede enviarse en la respuesta sin costo adicional. En tercer lugar, como RARP emplea una dirección de hardware de computadora para identificar una máquina, no puede utilizarse en redes con una asignación dinámica de direcciones de hardware.

Para sortear algunas de las dificultades de RARP, los investigadores desarrollaron el *BOOTstrap Protocol* (*BOOTP*). Más recientemente, el *Dynamic Host Configuration Protocol* (*DHCP*) ha sido propuesto como sucesor del BOOTP. Dado que los dos protocolos se encuentran estrechamente relacionados, la mayor parte de las descripciones en este capítulo se aplica a ambos. Para simplificar el texto, describiremos primero BOOTP y luego veremos cómo extiende el DHCP su funcionalidad a fin de proporcionar una asignación de direcciones dinámica.

Dado que utiliza al UDP y al IP, BOOTP debe implantarse con un programa de aplicación. Como RARP, BOOTP opera dentro de un paradigma cliente-servidor y requiere sólo de un intercambio de paquetes. No obstante, BOOTP es más eficiente que RARP pues un sólo mensaje BOOTP especifica muchos aspectos necesarios para arranque, incluyendo una dirección IP de computadora, la dirección de un ruteador y la dirección de un servidor. BOOTP también incluye un campo de vendedor específico en la respuesta, que permite a los vendedores de hardware enviar información adicional utilizada sólo en sus computadoras.[1]

---

[1] *Como veremos, el término "vendedor específico" es un nombre equivocado pues las especificaciones actuales* también recomiendan utilizar el área vendedor-específico para información de propósito general, como máscaras de subred; el DHCP cambia el nombre del campo a *opciones*.

21.3 Utilización de IP para determinar una dirección IP

Dijimos que BOOTP se vale del UDP para transportar mensajes y que los mensajes UDP están encapsulados en los datagramas IP para su entrega. A fin de entender cómo puede enviar una computadora a BOOTP en un datagrama IP antes de que la computadora conozca su dirección IP, recordemos, del capítulo 4, que hay varias direcciones IP de casos especiales. En particular, cuando se usa como una dirección de destino, la dirección IP está formada sólo por *unos* (255.255.255.255), que especifican el límite para la difusión. El software IP puede aceptar y difundir datagramas que especifican la dirección de difusión límite, incluso antes de que el software haya descubierto la información de su dirección IP local. El punto es el siguiente:

> *Un programa de aplicación puede utilizar la dirección IP de difusión límite para obligar al IP a difundir un datagrama en la red local, antes de que el IP haya descubierto la dirección IP de la red local o la dirección IP de la máquina.*

Supongamos que la máquina cliente *A* desea utilizar BOOTP para localizar información de arranque (incluyendo su dirección IP) y que *B* es el servidor en la misma red física que responderá a la solicitud. Dado que *A* no conoce la dirección IP de *B* o la dirección IP de la red, debe difundir en su BOOTP inicial la solicitud para utilizar la dirección IP de difusión límite. ¿Cual será la réplica? ¿Puede *B* enviar una réplica directamente? Por lo general no. Aun cuando pueda no ser obvio, *B* quizás necesite utilizar la dirección de difusión límite para su réplica, aunque conozca la dirección IP de *A*. Para entender por qué, considere qué sucedería si un programa de aplicación en *B* logra enviar un datagrama utilizando la dirección IP de *A*. Después de rutear el datagrama, el software IP en *B* pasará el datagrama al software de interfaz de red. El software de interfaz debe transformar la siguiente dirección IP de salto hacia la dirección de hardware correspondiente, presumiblemente utilizando ARP como se describe en el capítulo 5. Sin embargo, debido a que *A* no ha recibido la réplica BOOTP, no reconocerá su dirección IP y no podrá responder a la solicitud ARP de *B*. Además, *B* tiene sólo dos alternativas: difundir la réplica o utilizar información del paquete de solicitud para añadir de manera manual una entrada a su memoria intermedia ARP. En los sistemas que no permiten a los programas de aplicación modificar la memoria intermedia ARP, la difusión es la única solución.

21.4 Política de retransmisión BOOTP

BOOTP confiere toda la responsabilidad de la confiabilidad de la comunicación al cliente. Sabemos que como el UDP utiliza al IP para la entrega, los mensajes pueden retrasarse, perderse, entregarse fuera de orden o duplicarse. Además, dado que el IP no proporciona una suma de verificación para los datos, el datagrama UDP puede llegar con algunos bits alterados. Para protegerse contra la alteración de datos, BOOTP requiere que el UDP utilice sumas de verificación. También, especifica que las solicitudes y las réplicas deben enviarse con el bit de *no fragmentar* activado a fin de adaptarse a los clientes que tengan una memoria pequeña para reensamblar datagramas. BOOTP también está construido para permitir réplicas múltiples; las acepta y procesa primero.

Para manejar datagramas perdidos, BOOTP utiliza la técnica convencional de *tiempo límite* (*time out*) y *retransmisión* (*retransmission*). Cuando el cliente transmite una solicitud, inicia un temporizador. Si no llega ninguna réplica antes de que el tiempo expire, el cliente debe retransmitir la solicitud. Por supuesto, después de una falla en el suministro de alimentación, todas las máquinas en la red deben arrancar de nuevo de manera simultánea, posiblemente sobrecargando el servidor BOOTP con solicitudes. Si todos los clientes emplean exactamente el mismo tiempo límite de retransmisión, muchos de ellos o todos retransmitirán simultáneamente. Para evitar las colisiones resultantes, las especificaciones BOOTP recomiendan utilizar un retardo aleatorio. Además, las especificaciones aconsejan comenzar con un tiempo límite aleatorio de entre *0* y *4* segundos y duplicar el temporizador después de cada retransmisión. Luego de que el temporizador alcanza un valor alto, *60* segundos, el cliente no incrementa el temporizador, pero continúa utilizando el procedimiento de establecer un valor aleatoriamente. Duplicar el tiempo límite después de cada retransmisión evita que BOOTP añada un tráfico excesivo y congestione la red; el procedimiento aleatorio ayuda a evitar las transmisiones simultáneas.

21.5 Formato de los mensajes BOOTP

Para mantener a una implantación tan simple como sea posible, BOOTP los mensajes tienen campos de longitud fija y las réplicas poseen el mismo formato que las solicitudes. Aun cuando dijimos que los clientes y los servidores son programas, el protocolo BOOTP emplea los términos con cierta vaguedad al referirse a la máquina que envía una solicitud BOOTP como el *cliente* y a cualquier máquina que envía una réplica como el *servidor*. La figura 21.1 muestra el formato del mensaje BOOTP.

El campo *OP* especifica si el mensaje es una solicitud (*valor 1*) o una réplica (*valor 2*). Como en ARP, los campos *HTYPE* y *HLEN* especifican el tipo de hardware de red y la longitud de la dirección de hardware (por ejemplo, Ethernet tiene definido un tipo *1* y una dirección de una longitud de 6).[2] El cliente coloca *0* en el campo *HOPS*. Si recibe la solicitud y decide transferir la solicitud hacia otra máquina (por ejemplo, permitir el arranque a través de varios ruteadores), el servidor BOOTP incrementará el contador *HOPS*. El campo *TRANSACTION ID (ID DE TRANSACCIÓN)* contiene un entero que la máquina sin disco utiliza para cotejar las respuestas con las solicitudes. El campo *SECONDS* (*SEGUNDOS*) reporta el número de segundos desde que el cliente comenzó el arranque.

El campo *CLIENT IP ADDRESS* (*DIRECCIÓN IP DEL CLIENTE*) y todos los campos que le siguen contienen la mayor parte de la información importante. Para permitir una flexibilidad creciente, los clientes llenan los campos con toda la información con la que cuentan y dejan los campos restantes puestos en cero. Por ejemplo, si un cliente conoce el nombre o la dirección de un servidor específico desde el que espera información, puede llenar los campos *SERVER IP ADDRESS* (*DIRECCIÓN IP DEL SERVIDOR*) o *SERVER HOST NAME* (*NOMBRE DEL ANFITRIÓN SERVIDOR*). Si estos campos no son iguales a cero, sólo el servidor con el nombre-dirección que concuerde responderá a la solicitud. Si los campos están puestos en cero, responderá cualquier servidor que reciba la solicitud.

BOOTP puede utilizarse desde un cliente que ya conozca su dirección IP (por ejemplo, para obtener información del archivo de arranque). Un cliente que conozca su dirección IP la colocará

[2] Los valores para el campo *HTYPE* pueden encontrarse en el último Assigned Numbers RFC.

0	8	16	24 31
OP	HTYPE	HLEN	HOPS
ID DE TRANSACCIÓN			
SEGUNDOS		SIN USO	
DIRECCIÓN IP DE CLIENTE			
SU DIRECCIÓN IP			
DIRECCIÓN IP DEL SERVIDOR			
DIRECCIÓN IP DEL RUTEADOR			
DIRECCIÓN DE HARDWARE DE CLIENTE (16 OCTETOS) ⋮			
NOMBRE DE ANFITRIÓN SERVIDOR (64 OCTETOS) ⋮			
NOMBRE DE ARCHIVO DE ARRANQUE (128 OCTETOS) ⋮			
ÁREA DE VENDEDOR ESPECÍFICO (64 OCTETOS) ⋮			

Figura 21.1 Formato de un mensaje BOOTP. Para mantener las implementaciones lo suficientemente pequeñas como para ajustarse a una memoria ROM, todos los campos tienen longitudes fijas.

en el campo *CLIENT IP ADDRESS* (*DIRECCIÓN IP DEL CLIENTE*); otros clientes utilizarán cero en este campo. Si la dirección IP del cliente es cero en la solicitud, un servidor devolverá la dirección IP del cliente en el campo *YOUR IP ADDRESS* (*SU DIRECCIÓN IP*).

21.6 Procedimiento de arranque de dos pasos

BOOTP utiliza un procedimiento de arranque de dos pasos. No proporciona una imagen de memoria a los clientes —sólo proporciona al cliente información necesaria para obtener una imagen. El cliente entonces utiliza un segundo protocolo (por ejemplo, el TFTP considerado en el capítulo 24) para obtener la imagen de memoria. Aunque el procedimiento de dos pasos muchas veces parece innecesario, permite una clara separación de configuración y almacenamiento. Un servidor BOOTP no necesita correr en la misma máquina que almacena las imágenes de memoria. De hecho, el servidor BOOTP opera desde una simple base de datos que sólo conoce los nombres de las imágenes de memoria.

Mantener la configuración separada del almacenamiento es importante pues permite al administrador configurar conjuntos de máquinas para que éstas actúen en forma idéntica o de manera independiente. El campo *BOOT FILE NAME* de un mensaje BOOTP ilustra el concepto. Supongamos que un administrador tiene varias estaciones de trabajo con diferentes arquitecturas de hardware, y supongamos que cuando el usuario arranca una de las estaciones de trabajo, éstas seleccionan correr en UNIX o en un sistema operativo local. Debido a que el conjunto de estaciones de trabajo incluye múltiples arquitecturas de hardware, no operará en todas las máquinas una sola imagen de memoria. Para adaptarse a esta diversidad, BOOTP permite que el campo *BOOT FILE NAME* en una solicitud contenga un nombre genérico como "unix", lo cual significa "quiero arrancar el sistema operativo UNIX para esta máquina". El servidor BOOTP consulta su base de datos de configuración para transformar el nombre genérico en un nombre de archivo específico que contiene la imagen de memoria UNIX apropiada para el hardware del cliente y devuelve el nombre específico (es decir, completamente definido) en su réplica. Por supuesto, la base de datos de configuración también permite un arranque completamente automático mediante el cual el cliente coloca ceros en el campo *BOOT FILE NAME* y, con ello, BOOTP selecciona una imagen de memoria para la máquina. La ventaja del método automático es que permite al usuario especificar nombres genéricos que trabajen en cualquier máquina. No es necesario recordar nombres de archivos específicos o arquitecturas de hardware.

21.7 Campo área de vendedor específico

El campo *VENDOR-SPECIFIC AREA* (*ÁREA DE VENDEDOR ESPECÍFICO*) contiene información opcional para su transferencia del servidor al cliente. Aun cuando la sintaxis resulta intrincada, no es difícil. Los primeros 4 octetos del campo se llaman *magic cookie* y definen el formato de los temas restantes; el formato estándar descrito aquí utiliza un magic cookie con valor de 99.130.83.99 (notación decimal con puntos). A continuación de este campo, sigue una lista de términos, en la que cada aspecto contiene un octeto *type* (*tipo*), un octeto *length* (*longitud*) opcional y varios octetos *value* (*valor*). El estándar define los siguientes tipos que tienen longitudes de valores fijos predeterminados:

Tipo	Código	Valor de Longitud	Valor de Contenidos
Relleno	0	-	Cero — utilizado sólo como relleno
Máscara de subred	1	4	Máscara de subred para red local
Hora del día	2	4	Hora del día en tiempo universal
Fin	255	-	Fin de la lista de aspectos

Figura 21.2 Contenido de la información de vendedor. Los campos fijos deben existir para los tipos *1* y *2*, pero no para los tipos *0* y *255*.

Aun cuando una computadora puede obtener información de máscara de su red mediante una solicitud ICMP, el estándar recomienda ahora que los servidores BOOTP proporcionen la máscara de su red en cada réplica para suprimir mensajes ICMP innecesarios.

Algunos aspectos adicionales en *VENDOR-SPECIFIC AREA* tienen un octeto *type*, un octeto *length* y uno *value*, como se muestra en la figura 21.3.

21.8 La necesidad de una configuración dinámica

Como RARP, BOOTP fue diseñado para un ambiente relativamente estático en el que cada anfitrión tiene una conexión de red permanente. Un administrador crea un archivo de configuración BOOTP que especifica un conjunto de parámetros BOOTP para cada anfitrión. El archivo no cambia con frecuencia pues la configuración generalmente se mantiene estable. Por lo común, una configuración no registra cambios durante semanas.

Con la llegada de redes inalámbricas y computadoras portátiles como las laptop y las notebook, se ha vuelto posible transportar a las computadoras de una localidad a otra rápida y fácilmente. BOOTP no se adapta a esta situación pues la información de configuración no puede cambiar rápidamente. Así pues, sólo proporciona una transformación estática desde un identificador de anfitrión hacia parámetros para el anfitrión. Además, un administrador debe introducir un conjunto de parámetros para cada anfitrión y luego almacenar la información en un archivo de configuración de servidor BOOTP —BOOTP no incluye una forma para asignar dinámicamente valores a máquinas individuales. En particular, un administrador debe asignar cada anfitrión a una dirección IP y configurar el servidor para entender la transformación del identificador de anfitrión a la dirección IP.

Los parámetros de asignación estática trabajan bien si las computadoras se mantienen en localidades fijas y el administrador tiene suficientes direcciones IP para asignar a cada computadora

Tipo	Código	Longitud en octetos	Contenidos
Ruteadores	3	N	Direcciones IP de N/4 ruteadores
Servidor de hora	4	N	Direcciones IP de N/4 servidores de hora
Servidor IEN116	5	N	Direcciones IP de N/4 servidores IEN116
Servidor de dominio	6	N	Direcciones IP de N/4 servidores DNS
Servidor Log	7	N	Direcciones IP de N/4 servidores log
Servidor de citas	8	N	Direcciones IP de N/4 servidores de citas
Servidor Lpr	9	N	Direcciones IP de N/4 servidores lpr
Impress	10	N	Direcciones IP de N/4 servidores Impress
Servidor RLP	11	N	Direcciones IP de N/4 servidores RLP
Hostname	12	N	N bytes de nombre de anfitrión cliente
Tamaño de arranque	13	2	entero de 2 octetos para tamaño del archivo de arranque
RESERVADO	128-254	-	Reservado para usos específicos de la localidad

Figura 21.3 Tipo y contenido de aspecto del *VENDOR-SPECIFIC AREA* de una réplica BOOTP que tiene longitudes variables.

una dirección IP única. Sin embargo, en los casos en los que las computadoras se muevan con frecuencia o que el número de computadoras físicas exceda el de direcciones de anfitrión IP disponibles, la asignación estática generará sobrecargas excesivas.

Para entender cómo puede exceder el número de computadoras el de direcciones IP disponibles, consideremos una LAN, en el laboratorio de un colegio que ha sido asignado a direcciones clase C o a una subred de direcciones clase B con 255 direcciones. Supongamos que como el laboratorio sólo tiene sillas para 30 estudiantes, la cédula del laboratorio en 10 diferentes momentos durante una semana le da cabida a más de 300 estudiantes. Además, supongamos que cada estudiante transporta una computadora notebook personal que se utiliza en el laboratorio. En cualquier momento, la red tiene más de 30 computadoras activas. Sin embargo, ya que las direcciones de red pueden dar cabida a más de 255 anfitriones, un anfitrión no puede asignar una dirección única a cada computadora. Así, aunque recursos como las conexiones físicas limitan el número de conexiones simultáneas, el número de computadoras potencial que puede utilizar la instalación es elevado. Está claro que un sistema es inadecuado si requiere que el administrador cambie el archivo de configuración del servidor antes de que se añada una nueva computadora a la red y comience a comunicarse; se necesita pues un mecanismo automatizado.

21.9 Configuración dinámica de anfitrión

Para manejar la asignación de direcciones de manera automática, el IETF ha diseñado un nuevo protocolo. Conocido como *Dynamic Host Configuration Protocol* (*Protocolo de configuración dinámica de anfitrión* o *DHCP*), el nuevo protocolo extiende BOOTP de dos formas. En primer lugar, el DHCP permite que una computadora adquiera toda la información que necesita en un solo mensaje. Por ejemplo, además de una dirección IP, un mensaje DHCP puede tener una máscara de subred. En segundo lugar, el DHCP permite que una computadora posea una dirección IP en forma rápida y dinámica. Para utilizar el mecanismo de asignación de direcciones dinámico DHCP, un administrador debe configurar un servidor DHCP supliendo un conjunto de direcciones IP. Cada vez que una computadora nueva se conecta a la red, la computadora contacta al servidor y solicita una dirección. El servidor selecciona una de las direcciones especificadas por el administrador y la asigna a la computadora.

Para ser completamente general, el DHCP permite tres tipos de asignación de direcciones; un administrador selecciona cómo responderá el DHCP a cada red o a cada anfitrión. Como BOOTP, el DHCP permite la *configuración manual,* mediante la cual un administrador puede configurar una dirección específica para una computadora específica. El DHCP también permite la *configuración automática,* por medio de la cual el administrador permite a un servidor DHCP asignar una dirección permanente cuando una computadora es conectada por primera vez a la red. Por último, el DHCP permite una *configuración dinámica* completa, con la cual el servidor "presta" una dirección para una computadora por tiempo limitado.

Como en BOOTP, el DHCP utiliza la identidad del cliente para decidir cómo proceder. Cuando un cliente contacta un servidor DHCP, envía un identificador, por lo general, la dirección de hardware del cliente. El servidor utiliza el identificador del cliente y la red a la que el cliente se ha conectado para determinar cómo asignar el cliente y la dirección IP. Así, el administrador tiene un control completo sobre la forma en que se asignan las direcciones. Un servidor puede configu-

rarse para asignar direcciones a computadoras específicas de manera estática (como BOOTP), mientras permite a otras computadoras obtener dinámicamente direcciones de manera permanente o temporal.

21.10 Asignación dinámica de direcciones IP

La asignación dinámica de direcciones es el más significativo y novedoso aspecto del DHCP. A diferencia de la asignación de direcciones estática, utilizada en BOOTP, la asignación de direcciones dinámica no es una transformación uno a uno, y el servidor no necesita conocer la identidad de un cliente *a priori*. En particular, un servidor DHCP puede ser configurado para permitir que una computadora arbitraria obtenga una dirección IP y comience la comunicación. Así, el DHCP permite diseñar sistemas que se autoconfiguren. Luego de que una computadora ha sido conectada a la red, la computadora se vale del DHCP para obtener una dirección IP y entonces configura su software TCP/IP a fin de utilizar la dirección. Por supuesto, la autoconfiguración está sujeta a restricciones administrativas —es el administrador el que decide qué servidor DHCP puede realizar la autoconfiguración. En resumen:

> *Como el DHCP permite a un anfitrión obtener todos los parámetros necesarios para la comunicación, sin la intervención manual, también permite la autoconfiguración. Ésta se encuentra sujeta, por supuesto, a restricciones administrativas.*

Para hacer posible la autoconfiguración, un servidor del DHCP comienza con un conjunto de direcciones IP que el administrador de red asigna al servidor para su manejo. El administrador especifica las reglas bajo las que opera el servidor. Un cliente DHCP negocia el uso de una dirección intercambiando mensajes con un servidor. En el intercambio, el servidor proporciona una dirección para el cliente y el cliente verifica que la dirección sea aceptable. Una vez que el cliente ha aceptado una dirección, puede comenzar a utilizar la dirección para comunicarse.

A diferencia de la asignación de direcciones estática, que asigna permanentemente cada dirección IP a un anfitrión específico, la asignación de direcciones dinámica es temporal. Decimos que un servidor DHCP *arrienda* una dirección a un cliente por un período de tiempo finito. El servidor especifica el período de arrendamiento cuando asigna la dirección. Durante el período de arrendamiento, el servidor no arrendará la misma dirección a ningún otro cliente. Al final del período de arrendamiento, sin embargo, el cliente debe renovar el arrendamiento o dejar de usar la dirección.

¿Cuánto debe durar un arrendamiento DHCP? El tiempo óptimo de arrendamiento depende en particular de la red y de las necesidades de un anfitrión. Por ejemplo, para garantizar que las direcciones puedan reciclarse con rapidez, las computadoras en una red utilizadas por estudiantes en un laboratorio universitario deben tener un corto período de arrendamiento (por ejemplo, una hora). En contraste, la red de una compañía podría utilizar un período de arrendamiento de un día o de una semana. Para adaptarse a todos los posibles ambientes, el DHCP no especifica un período de arrendamiento fijo y constante. De hecho, el protocolo permite que un cliente solicite un período de arrendamiento específico y permite a un servidor informar al cliente que el período de arrenda-

miento está garantizado. Así, un administrador puede decidir durante cuánto tiempo podrá asignar cada servidor una dirección a un cliente. En el caso extremo, el DHCP reserva un valor *infinito* para permitir un arrendamiento por un período de tiempo indeterminadamente largo, como la asignación de direcciones permanente utilizada en BOOTP.

21.11 Obtención de direcciones múltiples

Una computadora multianfitriona (*multi-homed*) está conectada a más de una red. Cuando una computadora como ésta arranca, puede necesitar información de configuración para cada una de sus interfaces. Como un mensaje BOOTP, un mensaje DHCP suele proporcionar información acerca de una interfaz. Una computadora con varias interfaces debe manejar cada interfaz por separado. Así, aun cuando describimos del DHCP que una computadora necesita sólo una dirección, el lector debe recordar que cada interfaz de una computadora multianfitriona puede ser un punto diferente en el protocolo.

BOOTP y DHCP utilizan la noción *relay agent* (agente relevador) para permitir que una computadora contacte un servidor en una red no local. Cuando un agente relevador recibe una solicitud de difusión desde un cliente, envía la solicitud hacia un servidor y luego devuelve la réplica del servidor al anfitrión. Los agentes relevadores pueden complicar la configuración multianfitriona puesto que un servidor puede recibir varias solicitudes desde una misma computadora. Sin embargo, aun cuando BOOTP y DHCP utilizan el término *identificador de cliente*, asumimos que un cliente multianfitrión envía un valor que identifica a una interfaz muy particular (por ejemplo, a una dirección de hardware única). Así, un servidor siempre será capaz de distinguir entre solicitudes de un multianfitrión, aunque el servidor reciba tal solicitud por medio de un agente relevador.

21.12 Estados de adquisición de direcciones

Cuando se utiliza el DHCP para obtener una dirección IP, el cliente se encuentra en 1 de 6 estados. El diagrama de estados de transición en la figura 21.4 muestra los eventos y los mensajes que ocasionan que un cliente cambie de estado.

Cuando un cliente arranca por primera vez, entra en el estado *INITIALIZE* (*INICIALIZAR*). Para comenzar a adquirir una dirección IP, el cliente primero contacta a todos los servidores DHCP en la red local. Para hacerlo, el cliente difunde un mensaje *DHCPDISCOVER* y cambia al estado con el nombre *SELECT*. Dado que el protocolo es una extensión de BOOTP, el cliente envía el mensaje *DHCPDISCOVER* en un datagrama UDP con el puerto de destino activado para el puerto BOOTP (es decir, el puerto *67*). Todos los servidores DHCP de la red local reciben el mensaje y los servidores que hayan sido programados para responder a un cliente en particular enviarán un mensaje *DHCPOFFER*. Así, un cliente puede recibir ceros o más respuestas.

Mientras permanece en el estado *SELECT* (*SELECCIONADO*), el cliente reúne respuestas *DHCPOFFER* desde los servidores DHCP. Cada oferta contiene información de configuración para el cliente junto con una dirección IP que el servidor ofrece para arrendar al cliente. El cliente debe seleccionar una de las respuestas (por ejemplo, la primera en llegar) y negociar con el servi-

dor un arrendamiento. Para ello, el cliente envía al servidor un mensaje *DHCPREQUEST* y entra al estado *REQUEST*. A fin de enviar un acuse de recibo de la recepción de la solicitud y comenzar el arrendamiento, el servidor responde con el envío de un *DHCPACK*. El arribo y el acuse de recibo hacen que el cliente cambie al estado *BOUND*, en el cual el cliente procede a utilizar la dirección. En resumen:

Para utilizar el DHCP, un anfitrión debe volverse cliente y difundir un mensaje a todos los servidores de la red local. El anfitrión entonces reunirá los ofrecimientos de los servidores, seleccionará uno de ellos y verificará su aceptación por parte del servidor.

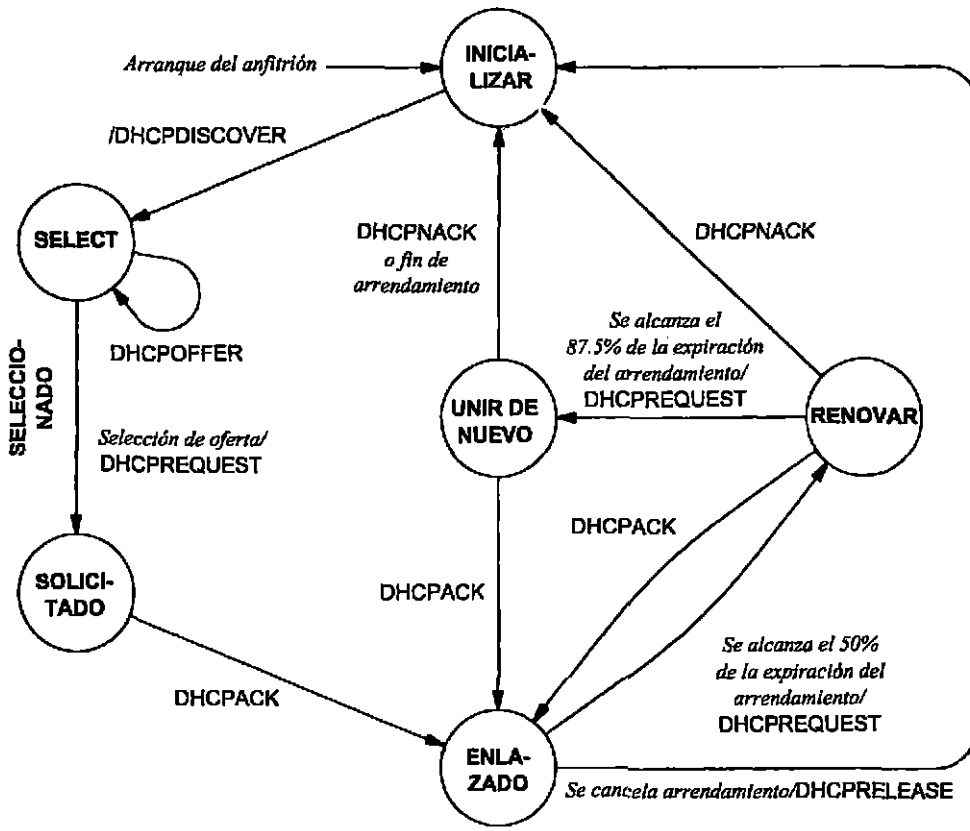


Figura 21.4 Los seis estados principales de un cliente DHCP y las transiciones entre éstos. Cada nombre en una transición lista el mensaje entrante o el evento que ocasiona la transición, seguido por una diagonal y el mensaje que envía el cliente.

21.13 Terminación temprana de arrendamiento

Pensemos el estado *BOUND* como el estado normal de operación; un cliente por lo general se mantiene en un estado *BOUND* mientras utiliza la dirección IP que ha adquirido. Si un cliente tiene almacenamiento secundario (por ejemplo, un disco local), puede almacenar la dirección IP que le fue asignada y solicitar la misma dirección cuando arranque de nuevo. En algunos casos, sin embargo, un cliente en el estado *BOUND* puede descubrir que no necesita por más tiempo una dirección IP. Por ejemplo, supongamos que un usuario conecta una computadora portátil a una red, utiliza el DHCP para adquirir una dirección IP y, luego, se vale del TCP/IP para leer correo electrónico. El usuario puede no saber por cuánto tiempo leerá su correspondencia, o bien, la computadora portátil podría permitir al servidor seleccionar el período arrendado. En cualquier caso, el DHCP especifica un periodo de arrendamiento mínimo de 1 hora. Si después de obtener una dirección IP, el usuario descubre que no tiene mensajes de correo electrónico para leer podría optar por desconectar la computadora portátil y cambiar hacia otra localidad.

Cuando no es necesario un arrendamiento por más tiempo, el DHCP permite que el cliente lo termine sin esperar a que su tiempo expire. Tal terminación es muy útil en los casos en los que ni el cliente ni el servidor pueden determinar una terminación de arrendamiento apropiada, al mismo tiempo que se garantiza el arrendamiento pues le es posible a un servidor seleccionar un periodo de arrendamiento razonablemente largo. Una finalización temprana es importante en especial si el número de dirección IP que un servidor tiene disponible es mucho más pequeño que el número de computadoras que están conectadas a la red. Si cada cliente termina su arrendamiento en cuanto la dirección IP deja de ser necesaria, el servidor será capaz de asignar la dirección a otro cliente.

Para terminar un arrendamiento de manera temprana, el cliente envía un mensaje *DHCPRELEASE* al servidor. Liberar una dirección es una acción final que previene que el cliente continúe utilizando la dirección. Así, luego de transmitir el mensaje de liberación, el cliente no debe enviar ningún otro datagrama que utilice la dirección. En términos del diagrama de transición de estados de la figura 21.4, un anfitrión que envía una *DHCPRELEASE* deja el estado *BOUND* y debe comenzar de nuevo en el estado *INITIALIZE* antes de utilizar el IP.

21.14 Estado de renovación de arrendamiento

Dijimos que cuando se adquiere una dirección, un cliente DHCP cambia al estado *BOUND*. Al entrar al estado *BOUND*, el cliente instala tres temporizadores que controlan la renovación de arrendamiento, la reasignación y la expiración. Un servidor DHCP puede especificar valores explícitos para los temporizadores cuando asigna una dirección al cliente; si el servidor no especifica valores para el temporizador, el cliente empleará valores por omisión. El valor por omisión para el primer temporizador es la mitad del tiempo que tarda el arrendamiento. Cuando el primer temporizador expira, el cliente debe lograr la renovación de su arrendamiento. Para solicitar una renovación, el cliente envía un mensaje *DHCPREQUEST* hacia el servidor desde el que fue obtenido el arrendamiento. El cliente entonces cambia al estado *RENEW* (*RENOVAR*) en espera de una respuesta. *DHCPREQUEST* contiene la dirección IP del cliente que está utilizando actualmente e interroga al servidor para extender el arrendamiento en esa dirección. Como en la negociación inicial de arrendamiento, un cliente puede solicitar un periodo para la extensión, pero el servidor controla, en últi-

re

0	8	16	24 31
OP	HTYPE	HLEN	HOPS
ID DE TRANSACCIÓN			
SEGUNDOS		BANDERAS	
DIRECCIÓN IP DE CLIENTE			
SU DIRECCIÓN IP			
DIRECCIÓN IP DEL SERVIDOR			
DIRECCIÓN IP DEL RUTEADOR			
DIRECCIÓN DE HARDWARE DE CLIENTE (16 OCTETOS) ⋮			
NOMBRE DE ANFITRIÓN SERVIDOR (64 OCTETOS) ⋮			
NOMBRE DEL ARCHIVO DE ARRANQUE (128 OCTETOS) ⋮			
OPCIONES (VARIABLE) ⋮			

Figura 21.5 Formato del mensaje DHCP, el cual es una extensión del mensaje BOOTP. El campo de opciones tiene una longitud variable; un cliente debe estar preparado para aceptar cuando menos 312 octetos por opción.

ma instancia, la renovación. Un servidor puede responder a la solicitud de renovación de un cliente de una de dos formas: puede instruir al cliente para que deje de usar la dirección o aprobar que la continúe utilizando. Si se aprueba esto último, el servidor envía un *DHCPACK*, el cual hace que el cliente regrese al estado *BOUND* y continúe utilizando la dirección. El *DHCPACK* puede también contener valores nuevos para los temporizadores del cliente. Si un servidor desaprueba que se continúe utilizando la dirección, el servidor enviará una *DHCPNACK* (acuse de recibo negativo), el cual hace que el cliente deje de utilizar la dirección inmediatamente y regrese al estado *INITIALIZE*.

Luego de enviar un mensaje *DHCPREQUEST* en el que solicite una extensión de su arrendamiento, el cliente se mantiene en el estado *RENEW* en espera de una respuesta. Si no se obtiene ninguna respuesta, el servidor que garantiza el arrendamiento se considera inactivo o inaccesible. Para manejar la situación, el DHCP libera un segundo temporizador, el cual fue instalado cuando el cliente entró al estado *BOUND*. El segundo temporizador expira luego de que se cumple el 87.5% del periodo de arrendamiento y hace que el cliente pase del estado *RENEW* al estado *REBIND* (*UNIR DE NUEVO*). Cuando se realiza la transición, el cliente asume que el anterior servidor DHCP no está disponible y comienza a difundir un mensaje *DHCPREQUEST* hacia cualquier servidor en la red local. Cualquier servidor configurado para proporcionar servicio a un cliente puede responder de manera positiva (por ejemplo, para extender el arrendamiento) o negativamente (esto

es, para no permitir que se siga usando la dirección IP). Si recibe una respuesta positiva, el cliente vuelve al estado *BOUND* y reinicializa los dos temporizadores. Si recibe una respuesta negativa, debe cambiar al estado *INITIALIZE*, dejar de usar inmediatamente la dirección IP y adquirir una nueva dirección IP antes de continuar utilizando el IP.

Luego de cambiar al estado REBIND, el cliente tendrá que interrogar al servidor original y a todos los servidores en la red local para una extensión del arrendamiento. En dado caso de que el cliente no reciba una respuesta de ningún servidor antes de que expire su tercer temporizador, el arrendamiento expirará. El cliente debe dejar de utilizar la dirección IP, regresar al estado *INITIALIZE* y comenzar la adquisición de una nueva dirección.

21.15 Formato de los mensajes DHCP

Como se muestra en la figura 21.5, el DHCP se vale del formato de mensaje BOOTP, pero modifica el contenido y el significado de algunos campos.

Como se muestra en la figura, casi todos los campos en un mensaje DHCP son idénticos a los de un mensaje BOOTP. De hecho, los dos protocolos son compatibles; un servidor DHCP puede ser programado para responder solicitudes BOOTP. Sin embargo, el DHCP cambia el significado de dos campos. Primero DHCP, interpreta el campo *UNUSED* (*SIN USO*) de BOOTP como un campo *FLAGS* (*BANDERAS*) de 16 bits. La figura 21.6 muestra que sólo el bit de orden superior del campo *FLAGS* tiene asignado un significado.

Como el mensaje de solicitud DHCP contiene la dirección de hardware del cliente, un servidor DHCP normalmente envía sus respuestas al cliente mediante una difusión de hardware. El cliente activa el bit de orden superior en el campo *FLAGS* para solicitar que el servidor responda por medio de la difusión y no de la unidifusión de hardware. Para entender por qué un cliente debe seleccionar una respuesta de difusión, recordemos que, cuando un cliente se comunica con un servidor DHCP, ya no tiene una dirección IP. Si un datagrama llega por medio de la unidifusión de hardware, y la dirección de destino no concuerda con la dirección de la computadora, el IP puede descartar el datagrama. Sin embargo, se requiere el IP para aceptar y manejar cualquier datagrama enviado hacia la dirección de difusión IP. Para asegurar que el software IP acepte y entregue mensajes DHCP que lleguen antes de que la dirección IP de la máquina se haya configurado, el cliente DHCP puede solicitar que el servidor envíe respuestas mediante la difusión IP.

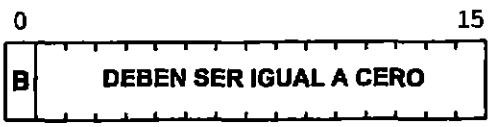


Figura 21.6 Formato del campo *FLAGS* (*BANDERAS*) de 16 bits en un mensaje DHCP. El bit de la extremo izquierdo se interpreta como una solicitud de difusión; todos los otros bits deben ser cero.

21.16 Opciones y tipos de mensajes DHCP

Sorprendentemente, el DHCP no añade nuevos campos fijos para el formato de los mensajes BOOTP, ni cambia el significado de la mayor parte de los campos. Por ejemplo, el campo *OP* en un mensaje DHCP contiene los mismos valores que el campo *OP* en un mensaje BOOTP: el mensaje es una solicitud de arranque (valor *1*) o una réplica de arranque (valor *2*). Para codificar información como la duración del arrendamiento, el DHCP utiliza *opciones*. En particular, la figura 21.7 ilustra la opción *tipo de mensaje DHCP* utilizada para especificar qué mensaje DHCP se está enviando.

El campo opciones tiene el mismo formato que la *VENDOR SPECIFIC AREA*, asimismo el DHCP acepta todos los temas de información de vendedores específicos definidos para BOOTP. Como en BOOTP, cada opción consiste en un campo de código y de un campo de longitud de 1 octeto respectivamente, seguidos por los octetos de datos que comprenden la opción. Como se muestra en la figura, la opción utilizada para especificar el tipo de mensaje DHCP consiste exactamente en 3 octetos. El primer octeto contiene el código *53*, el segundo la longitud *1* y el tercero un valor utilizado para identificar uno de los posibles mensajes DHCP.

CAMPO DE TIPO	Tipo de mensaje DHCP correspondiente
1	DHCPDISCOVER
2	DHCPOFFER
3	DHCPREQUEST
4	DHCPDECLINE
5	DHCPPACK
6	DHCPNACK
7	DHCPRELEASE

Figura 21.7 Formato de una opción de tipo de mensaje del DHCP utilizado para especificar el mensaje DHCP que se está enviando. La tabla lista posibles valores del tercer octeto y sus significados.

21.17 Opción Overload

Los campos *SERVER HOST NAME* y *BOOT FILE NAME* en el encabezado del mensaje DHCP ocupan muchos octetos. Si un mensaje dado no contiene información, en ninguno de estos campos, el espacio se desperdicia. Para permitir que un servidor DHCP utilice los dos campos para otras secciones, el DHCP define una opción *Option Overload*. Cuando está presente, la opción de sobrecarga informa al receptor que debe ignorar el significado usual de los campos *SERVER HOST NAME* y *BOOT FILE NAME*, y que debe considerar las opciones que están en lugar de los campos.

21.18 DHCP y nombres de dominios[3]

Aun cuando puede asignar direcciones IP a una computadora que lo demande, el DHCP no automatiza por completo todo el procedimiento requerido para conectar un anfitrión permanente a una red de redes. En particular, el DHCP no interactúa con el sistema de nombre de dominio. Así, la asignación entre un nombre de anfitrión y la asignación DHCP de la dirección IP del anfitrión se deben manejar de manera independiente.

¿Qué nombre deberá recibir el anfitrión cuando tenga una dirección IP desde DHCP? Conceptualmente, hay tres posibilidades. En la primera, el anfitrión no recibe un nombre. Aun cuando es posible correr software del cliente en un anfitrión sin un nombre, utilizar una computadora sin nombre puede ser inconveniente. En segundo lugar, el anfitrión está asignado de manera automática a un nombre junto con una dirección IP. Este método es muy popular en la actualidad ya que los nombres pueden ser preasignados y no se requieren cambios para DNS. Por ejemplo, un administrador de sistema puede configurar el servidor de nombre de dominios local a fin de tener un nombre de anfitrión para cada dirección IP manejada por DHCP. Una vez que ha sido instalado en DNS, la asignación nombre-a-dirección se mantiene estática. La mayor desventaja de la asignación estática es que al anfitrión se le da un nombre nuevo cada vez que recibe una nueva dirección (por ejemplo, cuando un anfitrión cambia de una red física a otra). En tercer lugar, el anfitrión puede ser *asignado a un nombre permanente que se mantiene sin cambios.* Conservar un nombre de anfitrión de manera permanente es conveniente pues la computadora puede ser accesada siempre por medio de un solo nombre, independientemente de la localización actual de la computadora.

Se necesitan mecanismos adicionales para soportar nombres de anfitrión permanentemente. En particular, los nombres de anfitrión permanentes requieren de una coordinación entre DHCP y DNS. Un servidor DNS debe cambiar la asignación, nombre-a-dirección cada vez que un anfitrión reciba una dirección IP y retirar la asignación cuando expire el arrendamiento. No obstante, un grupo de trabajo IETF está considerando actualmente cómo hacer que interactúe DHCP con el sistema de nombre de dominios. De momento, no hay protocolos para actualizaciones DNS dinámicos. Así pues, hasta que se desarrolle un mecanismo de actualización dinámico, no habrá protocolo que mantenga nombres de anfitrión de manera permanente y que permita a al DHCP cambiar direcciones IP.

21.19 Resumen

El protocolo de arranque BOOTP proporciona una alternativa a RARP para computadoras que necesitan determinar su dirección IP. BOOTP es más general que RARP pues utiliza el UDP, lo que hace posible extender el proceso de arranque a través de un ruteador. BOOTP también permite a una máquina determinar una dirección de ruteador, una dirección (archivo) de servidor y el nombre de un programa que la computadora deberá correr. Finalmente, BOOTP permite a los administradores establecer la configuración de una base de datos que transforma un nombre genérico como

---

[3] En el capítulo 22, se considera el Sistema de Nombres de Dominio en detalle.

"unix" en un nombre de archivo completamente caracterizado que contiene la imagen de memoria apropiada para el hardware del cliente.

BOOTP está diseñado para ser lo suficientemente pequeño y simple como para residir en un arranque localizado en ROM. El cliente utiliza la dirección de difusión limitada para comunicarse con el servidor y tiene la responsabilidad de retransmitir solicitudes si el servidor no responde. La retransmisión emplea un procedimiento de retroceso exponencial similar a Ethernet para evitar el congestionamiento.

Diseñado como sucesor de BOOTP, el Dynamic Host Configuration Protocol (DHCP) amplía BOOTP de varias formas. Lo más importante es que el DHCP permite que un servidor localice direcciones IP automática o dinámicamente. La asignación dinámica es necesaria para ambientes como las redes inalámbricas cuyas computadoras pueden conectarse y desconectarse rápidamente. Para utilizar el DHCP, una computadora debe convertirse en cliente. La computadora difunde una solicitud para los servidores DHCP, selecciona una de las ofertas recibidas e intercambia mensajes con el servidor a fin de obtener un arrendamiento de la dirección IP anunciada.

Cuando un cliente obtiene una dirección IP, arranca 3 temporizadores. Luego de que el primer temporizador expira, el cliente debe intentar la renovación de su arrendamiento. Si un segundo temporizador expira antes de que se complete la renovación, el cliente debe tratar de reasignar su dirección a cualquier servidor. Si el último temporizador expira antes de que se haya obtenido un arrendamiento, el cliente deja de utilizar la dirección IP y regresa al estado inicial para adquirir una nueva dirección. Una máquina de estado finito explica la adquisición de arrendamiento y su renovación.

PARA CONOCER MÁS

BOOTP es un protocolo estándar en la serie del TCP/IP. Se pueden obtener mayores detalles en Croft y Gilmore (RFC 951), en el cual se compara BOOTP con RARP y sirve como estándar oficial. Reynolds (RFC 1084) explica cómo interpretar el área de vendedor específico y Braden (RFC 1123) recomienda utilizar el área de vendedor específico para transferir la máscara de subred.

Droms (RFC 1541) presenta la última especificación para el DHCP, así como una descripción detallada de las transiciones de estado; se espera pronto otra revisión. En un documento relacionado, Alexander (RFC 1533) especifica la codificación de las opciones DHCP y la extensión de vendedor BOOTP. Por último, Droms (RFC 1534) analiza la interoperabilidad entre BOOTP y DHCP.

EJERCICIOS

21.1 BOOTP no contiene un campo explícito para volver a poner la hora del día del servidor al cliente, pero lo hace parte (opcional) de la información del vendedor específico. ¿La hora debería ser incluida en los campos requeridos? Explique por qué sí o por qué no.

21.2 Exponga qué separación de configuración y almacenamiento *no* es buena. (Sugerencia: consulte el RFC 951).

21.3 *El formato de mensaje BOOTP es inconsistente pues tiene dos campos para la dirección IP de cliente* y uno para el nombre de la imagen de arranque. Si el cliente deja su campo de dirección IP vacío, el servidor devuelve la dirección IP del cliente en el segundo campo. Si el cliente deja el campo de nombre del archivo de arranque vacío, el servidor lo *reemplaza* con un nombre explícito. ¿Por qué?

21.4 Lea el estándar para encontrar cómo utilizan clientes y servidores el campo *HOPS*.

21.5 Cuando un cliente BOOTP recibe una réplica por medio de la difusión de hardware, ¿cómo sabe si la réplica está dirigida a otro cliente BOOTP en la misma red física?

21.6 Cuando una máquina obtiene una máscara de subred con BOOTP en lugar de ICMP, coloca la carga menor en *otras* computadoras anfitrión. Explíquelo.

21.7 Lea el estándar para encontrar cómo pueden acordar un cliente DHCP y un servidor la duración de arrendamiento sin tener relojes sincronizados.

21.8 Considere un anfitrión que tiene un disco y utiliza DHCP para obtener una dirección IP. Si el anfitrión almacena su dirección en un disco junto con la fecha en que expira el arrendamiento y luego se reinicializa dentro del periodo de arrendamiento, ¿puede utilizar la dirección? ¿Por qué sí o por qué no?

21.9 El DHCP establece un arrendamiento de dirección mínimo de una hora. ¿Puede usted imaginar una situación en la que el arrendamiento mínimo del DHCP provoque inconvenientes? Explíquelo.

21.10 *Lea el RFC para encontrar cómo especifica* el DHCP la renovación y la reasignación de temporizadores. ¿Un servidor debe establecer siempre uno sin el otro? ¿Por qué sí o por qué no?

21.11 El diagrama de transición de estado no muestra la retransmisión. Lea el estándar para encontrar cómo muchas veces debe retransmitir un cliente una solicitud.

21.12 ¿Puede el DHCP garantizar que un cliente no es "engañado" (es decir, puede el DHCP garantizar que no está enviando información de configuración del anfitrión *A* al anfitrión *B*)? ¿La respuesta difiere para BOOTP? Explique por qué sí o por qué no.

21.13 El DHCP especifica que un anfitrión debe prepararse para manejar por lo menos *312* octetos de opciones. ¿Cómo se obtiene el número *312*?

21.14 ¿Puede una computadora que utiliza al DHCP obtener una dirección IP operando un servidor? Si así es, ¿cómo accede un cliente al servidor?

22

Sistema de nombre de dominio (DNS)

22.1 Introducción

Los protocolos descritos en los primeros capítulos utilizan enteros de 32 bits, llamados direcciones de Protocolo Internet (direcciones IP) para identificar máquinas. Aun cuando cada dirección proporciona una representación compacta y conveniente para identificar la fuente y el destino en paquetes enviados a través de una red de redes, los usuarios prefieren asignar a las máquinas nombres fáciles de pronunciar y recordar.

En este capítulo, consideraremos un esquema para asignar nombres significativos de alto nivel a grandes conjuntos de máquinas y direcciones IP. Veremos la traducción de un nombre de alto nivel a una dirección IP y la traducción des una dirección IP a un nombre de alto nivel para una máquina. El esquema de nombres es interesante por dos razones. En primer lugar, ha sido utilizado para asignar nombres de máquinas a través de la red global de Internet. En segundo, dado que utiliza un conjunto de servidores distribuidos geográficamente para transformar nombres en direcciones, la implantación del mecanismo de transformación de nombres proporciona un ejemplo a gran escala del paradigma cliente-servidor descrito en el capítulo 19.

22.2 Nombres para las máquinas

Los primeros sistemas de computadoras forzaban a los usuarios a entender direcciones numéricas para objetos como tablas de sistema y dispositivos periféricos. Los sistemas de tiempo compartido mejoraron el cómputo al permitir que los usuarios inventaran nombres simbólicos y significativos

para objetos físicos (por ejemplo, dispositivos periféricos) y objetos abstractos (por ejemplo, archivos). Un modelo similar ha aparecido en las redes de computadoras. Los primeros sistemas soportaban conexiones punto a punto entre computadoras y utilizaban direcciones de hardware de bajo nivel para especificar máquinas. El enlace de redes introduce el direccionamiento universal así como el software de protocolo para transformar direcciones universales en direcciones de hardware de bajo nivel. Como en la mayor parte de los ambientes de computación hay varias máquinas, los usuarios necesitan nombres simbólicos y significativos para nombrarlas.

Los primeros nombres de máquinas reflejan los ambientes pequeños en los que se seleccionaron. Era muy común, para localidades con un puñado de máquinas, elegir los nombres en base al propósito de las máquinas. Por ejemplo, las máquinas a menudo tenían nombres como *acceso, producción, contabilidad* y *desarrollo*. Los usuarios preferían estos nombres a las incómodas direcciones de hardware.

Aun cuando la diferencia entre *dirección* y *nombre* es significativa intuitivamente, resulta artificial. Cualquier *nombre* es sólo un identificador que consiste en una secuencia de caracteres seleccionados de un alfabeto finito. Los nombres sólo son útiles si el sistema puede transformarlos de manera eficiente para referirse al objeto que denotan. Así, pensamos en una dirección IP como en un *nombre de bajo nivel* y decimos que el usuario prefiere utilizar *nombres de alto nivel* para las máquinas.

La forma de los nombres de alto nivel es importante pues determina cómo son traducidos los nombres a nombres de bajo nivel o cómo conducen a objetos, también determina la forma en que se autoriza la asignación de nombres. Cuando sólo se tiene unas cuantas máquinas interconectadas, la selección de nombres es fácil y cualquier forma será suficiente. En Internet, donde hay alrededor de cuatro millones de máquinas conectadas, la selección de nombres se vuelve difícil. Por ejemplo, cuando el departamento principal de computadoras fue conectado a Internet en 1980, el Departamento de Ciencias Computacionales de la Universidad de Purdue seleccionó el nombre *purdue* para identificar a la máquina conectada. La lista de conflictos potenciales contenía sólo una docena de nombres. A mediados de 1986, la lista oficial de anfitriones en Internet contenía 3100 nombres registrados y 6500 alias.[1] A pesar de que la lista fue creciendo rápidamente en los años ochenta, la mayor parte de las localidades tiene máquinas adicionales (por ejemplo, computadoras personales) que no están registradas.

22.3 Espacio de nombre plano

El conjunto original de nombres utilizados a través de Internet formaba un *espacio de nombre plano* en el que cada nombre consistía en una secuencia de caracteres sin ninguna estructura adicional. En el esquema original, una localidad central, la Network Information Center (NIC), administraba el espacio de nombres y determinaba si un nombre nuevo era apropiado (esto es, se prohibían nombres obscenos o nombres nuevos que crearan conflictos con nombres ya existentes). Más adelante, el NIC fue reemplazado por el INTERnet Network Information Center (*INTERNIC*).

[1] Hacia 1990, más de 137,000 anfitriones de Internet tenían nombres y en 1995 el número rebasaba los 4 millones.

La mayor ventaja del espacio de nombres plano es que los nombres eran convenientes y cortos; la mayor desventaja es que el espacio de nombres plano no podía generalizarse para grandes conjuntos de máquinas por razones técnicas y administrativas. En primer lugar, como los nombres se construyen desde un sólo conjunto de identificadores, la posibilidad de conflictos se incrementa conforme crece el número de localidades. En segundo lugar, dado que la autoridad para añadir nombres nuevos debe residir en una sola localidad, la sobrecarga de trabajo administrativo en la localidad central se incrementa con el número de localidades. Para entender la seriedad del problema, imagine el crecimiento rápido de una red de redes con cientos de localidades, donde cada una tiene cientos de computadoras personales individuales y estaciones de trabajo. Cada vez que alguien adquiere y conecta una nueva computadora personal, su nombre debe ser aprobado por la autoridad central. En tercer lugar, como los nombres de dirección cambian con frecuencia, es elevado el costo de mantener copias correctas de las listas completas en cada localidad, y este costo se incrementa conforme crece el número de localidades. Además, si la base de datos de nombres está localizada en una sola localidad, el tráfico de red hacia dicha localidad se incrementará junto con el número de localidades.

22.4 Nombres jerárquicos

¿Cómo puede un sistema de nombres adaptarse al crecimiento rápido y extenso del conjunto de nombres sin requerir una localidad central que lo administre? La respuesta está en la descentralización del mecanismo de asignación de nombres, mediante el cual se delega la autoridad de partes del espacio de los nombres y se reparte la responsabilidad de la traducción de nombres y direcciones. Las redes de redes TCP/IP utilizan dicho esquema. Antes de examinar los detalles del esquema de TCP/IP consideraremos la motivación y la intuición subyacentes.

La partición del espacio de nombre debe definirse de forma que soporte la transformación eficiente de nombres y que garantice un control autónomo de la asignación de los mismos. Optimizar sólo la eficiencia de la transformación puede conducir a soluciones que conserven un espacio de nombres plano y reduzca el tráfico al dividir los nombres entre varias máquinas de transformación. Optimizar sólo el aspecto administrativo puede llevar a soluciones que facilitan la delegación de autoridad pero que hacen la transformación de nombres costosa o compleja.

Para entender cómo sería dividido el espacio de nombres, considere la estructura de una amplia organización. En el nivel superior, el ejecutivo principal tiene una responsabilidad general. Como éste no puede mirar hacia todas partes, la organización debe repartirse en divisiones, con un ejecutivo a cargo de cada división. El ejecutivo principal garantiza la autonomía de cada división dentro de límites específicos. Más aún, los ejecutivos a cargo de una división en particular pueden contratar o suspender a empleados, asignar oficinas y delegar autoridad, sin necesidad de obtener un permiso directo del ejecutivo principal.

Además de facilitar la delegación de autoridad, la jerarquía de una amplía organización introduce la autonomía de operaciones. Por ejemplo, cuando un oficinista necesita información, como el número telefónico de un nuevo empleado, comenzará por preguntar a los empleados de la oficina local (quienes pueden contactar a los trabajadores de oficina de otro local). El punto es que, aun cuando la autoridad siempre está bajo la jerarquía corporativa, la información puede fluir a través de la jerarquía desde una oficina a otra.

22.5 Delegar autoridad para los nombres

Un esquema de nombres jerárquico funciona como una administración extensa. El espacio de nombre es *particionado* en el nivel superior y la autoridad para los nombres de subdivisiones pasa a agentes designados. Por ejemplo, se debe seleccionar el espacio de nombres para particionarlo en base a un *nombre de localidad* y delegar a cada localidad la responsabilidad de mantener los nombres dentro de esta partición. El nivel superior de la jerarquía divide el espacio de nombres y delega la autoridad a cada división; es necesario que esto no sea modificado por los cambios que suceden dentro de una división.

La sintaxis de asignación jerárquica de nombres casi siempre refleja la delegación jerárquica de la autoridad utilizada para asignarlos. Como un ejemplo, consideremos un espacio de nombre con la forma:

local . *localidad*

donde *localidad* es el nombre de la localidad autorizada por la autoridad central, *local* es la parte del nombre controlado por la localidad y, el punto (".")[2] es un delimitador empleado para separarlos. Cuando la máxima autoridad añade una nueva localidad, *X*, ésta añade *X* a la lista de localidades válidas y delega a la localidad *X* la autoridad sobre todos los nombres que terminen con ". *X*".

22.6 Autoridad para los subconjuntos de nombres

En un espacio jerárquico de nombres, la autoridad puede ser subdividida en cada nivel. En nuestro ejemplo de partición por localidades, la localidad en sí puede consistir en varios grupos administrativos y la autoridad de la localidad puede elegir subdividir sus espacios de nombres entre los grupos. La idea es conservar subdividido el espacio de nombres hasta que cada subdivisión sea lo suficientemente pequeña como para que se pueda manejar.

Sintácticamente, subdividir el espacio de nombres introduce otra partición del nombre. Por ejemplo, añadir una subdivisión *grupo* al nombre ya dividido por localidad, genera la siguiente sintaxis de nombre:

local . *grupo* . *localidad*

Dado que el nivel superior delega autoridad, el nombre de grupo no tiene que concordar en todas las localidades. La localidad de una universidad podría elegir nombres de grupo como *ingeniería, ciencia* y *arte,* mientras que la localidad de una compañía podría seleccionar nombres de grupo como *producción, contabilidad* y *personal.*

El sistema telefónico de Estados Unidos ofrece otro ejemplo de sintaxis jerárquica de nombres. Los diez dígitos de un número telefónico se han particionado en tres dígitos para el *código de área,*

[2] En los nombres de dominio, el carácter delimitador se lee como "punto".

tres dígitos para *intercambio* y cuatro dígitos para el *número de suscriptor* dentro de la central telefónica. Cada central telefónica tiene autoridad para asignar números de suscriptores dentro de su sección del espacio de nombre. Aun cuando es posible un grupo de suscriptores arbitrario en la central telefónica y un grupo arbitrario de centrales telefónicas dentro de un código de área, la asignación de números telefónicos no es caprichosa; se debe realizar una selección cuidadosa para facilitar el ruteo de las llamadas telefónicas a través de la red telefónica.

El ejemplo del teléfono es importante pues ilustra una diferencia clave entre el esquema de nombres jerárquico utilizado en una red de redes TCP/IP y otras jerarquías. Particionar el conjunto de máquinas de una organización a través de las líneas de autoridad no necesariamente implica realizar una partición por localidades físicas. Por ejemplo, puede ser que en algunas universidades exista un sólo edificio para el departamento de matemáticas y para el departamento de ciencias computacionales. Podría suceder que, incluso, aunque las máquinas de estos dos grupos pertenezcan a dominios administrativos separados por completo, se encuentren conectadas a la misma red física. También podría suceder que un solo grupo administrativo fuera propietario de máquinas en varias redes físicas. Por esta razón, el esquema de nombres del TCP/IP permite una delegación arbitraria de autoridad para el espacio de nombres jerárquico sin considerar las conexiones físicas. El concepto puede resumirse de la siguiente manera:

> *En una red de redes TCP/IP, la jerarquía de nombres de máquinas se asigna de acuerdo con la estructura de la organización que obtiene la autoridad para dividir el espacio de nombres y no necesariamente de acuerdo con la estructura de las interconexiones de red física.*

Por supuesto, en muchas localidades la jerarquía organizacional corresponde a la estructura de las interconexiones físicas de la red. En una universidad grande, por ejemplo, la mayor parte de los departamentos tiene su propia red de área local. Si el departamento es una parte asignada a la jerarquía de nombres, todas las máquinas que tengan un nombre en esta parte de la jerarquía, también se conectarán a una sola red física.

22.7 Nombres de dominio TCP/IP de Internet

El mecanismo que implanta una jerarquía de nombres de máquina para las redes de redes TCP/IP se conoce como *Domain Name System* (*Sistema de Nombres o Nomenclatura de Dominio* o *DNS*). El DNS tiene dos aspectos conceptualmente independientes. El primero es abstracto. Especifica la sintaxis del nombre y las reglas para delegar la autoridad respecto a los nombres. El segundo es concreto: especifica la implantación de un sistema de computación distribuido que transforma eficientemente los nombres en direcciones. En esta sección, se considera la sintaxis del nombre y, en secciones posteriores, se examina la implantación.

El sistema de nombres de dominio se vale de un esquema de nombres jerárquico, conocido como *nombre de dominio*. Como en nuestros primeros ejemplos, un nombre de dominio consiste en una secuencia de nombres separados por un carácter delimitador, el punto. En nuestros ejemplos, dijimos que secciones particulares del nombre debían representar localidades o grupos, pero el sistema de dominio sencillamente llama a cada sección *etiqueta*. Así, el nombre de dominio

cs . purdue . edu

contiene tres *etiquetas*: *cs, purdue* y *edu*. Cualquier sufijo de una etiqueta en un nombre de dominio es llamado también *dominio*. En el ejemplo de arriba, el dominio de nivel inferior es *cs . purdue . edu*, (el nombre de dominio para el Departamento de Ciencias Computacionales de la Universidad de Purdue), el segundo nivel de dominio es *purdue . edu* (el nombre de dominio para la Universidad de Purdue) y el nivel superior dominio es *edu* (el nombre de dominio para las instituciones educativas). Como se muestra en el ejemplo, los nombres de dominio están escritos con la etiqueta local primero y el dominio superior al último. Como veremos, describirlos en este orden hace posible comprimir los mensajes que contienen varios nombres de dominio.

22.8 Nombres de dominio oficiales y no oficiales de Internet

En teoría, el estándar de nombres de dominio especifica un espacio de nombre jerárquico abstracto con valores arbitrarios para las etiquetas. Como el sistema de dominio dicta sólo la forma de los nombres y no sus valores actuales, es posible, para cualquier grupo que constituya una instancia de sistema de dominio, seleccionar etiquetas para todas las partes de su jerarquía. Por ejemplo, una compañía privada puede establecer una jerarquía de dominios en la que las etiquetas de nivel superior especifiquen corporaciones y subsidiarias, el siguiente nivel divisiones corporativas y el nivel inferior los departamentos.

Sin embargo, la mayoría de los usuarios de tecnología de dominio sigue la jerarquía de etiquetas utilizada por el sistema de dominio oficial de Internet. Hay dos razones para ello. En primer lugar, como veremos, el esquema de Internet es completo y flexible. Se puede adaptar a una amplia variedad de organizaciones y permite a cada grupo seleccionar entre una jerarquía de nombres asignada geográficamente o en función de la estructura organizativa. En segundo lugar, la mayor parte de las localidades sigue el esquema de Internet porque de esta manera puede conectar sus instalaciones TCP/IP a la red global de Internet sin cambiar nombres. Dado que el esquema de nombres de Internet predomina en casi todos los usos del sistema de nombres de dominios, los ejemplos a lo largo de lo que resta de este capítulo tienen etiquetas tomadas de la jerarquía de dominio de Internet. Los lectores pueden recordar que, aunque es más común encontrar estas etiquetas, la tecnología de sistema de nombres de dominio puede utilizarse con otras etiquetas si se desea.

La autoridad de Internet ha seleccionado particionar su nivel superior en los dominios que se listan en la figura 22.1.

Conceptualmente, el nombre de nivel superior permite dos jerarquías de nombres completamente diferentes: el esquema geográfico y el organizacional. El geográfico divide el universo de máquinas por país. Las máquinas de Estados Unidos quedan bajo el dominio de nivel superior *US*; cuando otro país desea registrar máquinas en el sistema de nombres de dominio, la autoridad central asigna al país un nuevo dominio de nivel superior con el estándar internacional del país identificado por dos letras como su etiqueta. La autoridad para el dominio de US ha seleccionado dividirlo dentro de un dominio de segundo nivel por estado. Por ejemplo, el dominio para el estado de Virginia es

va . us

Nombre de dominio	Significado
COM	Organizaciones comerciales
EDU	Instituciones educativas
GOV	Instituciones gubernamentales
MIL	Grupos militares
NET	Centros mayores de soporte de red
ORG	Organizaciones diferentes a las anteriores
ARPA	Dominio temporal de ARPANET (obsoleto)
INT	Organizaciones internacionales
código de país	País en particular (según esquema geográfico)

Figura 22.1 Dominios de Internet de nivel superior y su significado. Aun cuando los nombres se muestran en mayúsculas, el sistema de nombres de dominio es insensible a la distinción de mayúsculas y minúsculas, así pues, *EDU* es equivalente a *edu*.

Como alternativa para la jerarquía geográfica, el dominio de nivel superior también permite que las organizaciones se agrupen en función de su organización. Cuando una organización desea participar en el sistema de nombres de dominio, decide la forma en que desea que se registre y solicita su aprobación. La autoridad central revisa la solicitud y asigna un subdominio a la organización[3] bajo uno de los dominios de nivel superior existentes. Por ejemplo, es posible que una universidad se registre con un dominio de segundo nivel *EDU* (práctica común) o que se registre según el estado y el país en el que se localiza. De esta manera, algunas organizaciones han seleccionado la jerarquía geográfica; la mayoría prefiere registrarse con *COM, EDU, MIL* o *GOV*. Hay dos razones para ello. En primer lugar, los nombres geográficos son extensos y, además, difíciles de escribir. En segundo lugar, los nombres geográficos son mucho más difíciles de encontrar o adivinar. Por ejemplo, la Universidad de Purdue se localiza en West Lafayette, Indiana. Mientras que un usuario puede adivinar fácilmente un nombre organizacional como *purdue . edu*, un nombre geográfico resulta, con frecuencia, difícil de adivinar pues por lo general es una abreviatura como *laf . in . us*.

Otro ejemplo puede ayudar a aclarar la relación entre la jerarquía de nombres y la autoridad para los nombres. Una máquina llamada *xinu* en el Departamento de Ciencias Computacionales de la Universidad de Purdue tiene el nombre de dominio oficial

xinu . cs . purdue . edu

El nombre de la máquina fue aprobado y registrado por el administrador de red local en el Departamento de Ciencias Computacionales. El administrador del departamento había obtenido previamente autorización para el subdominio *cs . purdue . edu* de una autoridad de la red universitaria, quien a su vez había obtenido permiso para administrar el subdominio *purdue . edu* de la autoridad de Internet. La autoridad de Internet conserva el control del dominio *edu*, de manera que nuevas

[3] El estándar no define el término "subdominio". Elegimos utilizarlo pues su analogía con el término "subconjunto" ayuda a aclarar la relación entre dominios.

universidades pueden añadirse sólo con su permiso. En forma similar, el administrador de red de la Universidad de Purdue conserva la autoridad para el subdominio *purdue . edu*, de manera que los nuevos dominios de tercer nivel sólo pueden ser añadidos con la autorización del administrador.

La figura 22.2 ilustra una pequeña parte de la jerarquía de nombres de dominio de Internet. Como se muestra en la figura, Digital Equipment Corporation, una organización comercial, está registrada como *dec . com*, la Universidad de Purdue está registrada como *purdue . edu*, y la National Science Foundation, una institución gubernamental, está registrada como *nsf . gov*. En contraste, la Corporation for National Research Initiatives eligió registrarse bajo la jerarquía geográfica como *cnri . reston . va . us*.[4]

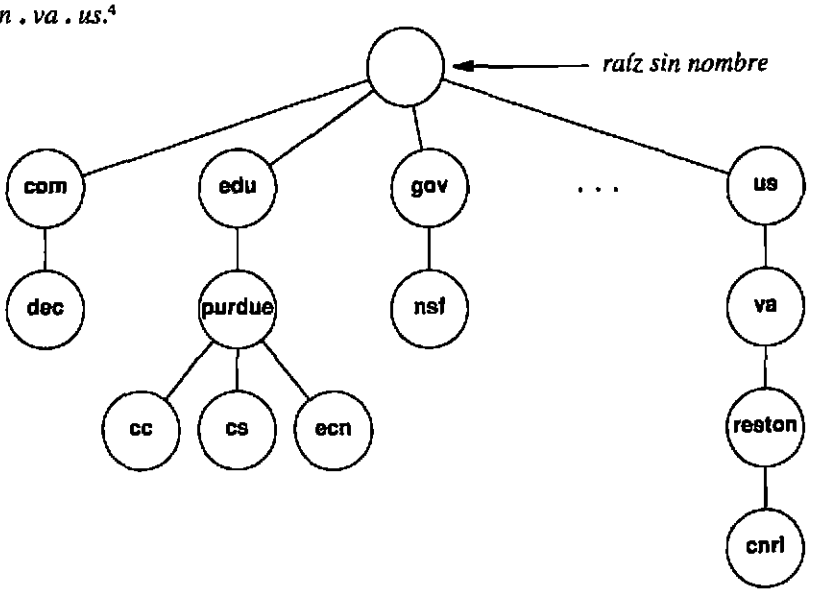


Figura 22.2 Una pequeña parte de la jerarquía de nombres de dominio (árbol) de Internet. En la práctica, el árbol es extenso y plano; la mayor parte de los anfitriones aparecen en el quinto nivel.

22.9 Cosas por nombrar y sintaxis de los nombres

El sistema de nombres de dominio es completamente general ya que permite que múltiples jerarquías de nombres se incorporen en un sistema. Para permitir a los clientes distinguir entre varios tipos de entrada, cada aspecto nombrado, almacenado en el sistema, es asignado a un *tipo* que especifica si se trata de la dirección de una máquina, un buzón, un usuario, etcétera. Cuando un cliente interroga al sistema de dominio para resolver el problema de un nombre, debe especificar el tipo de respuesta deseado. Por ejemplo, cuando una aplicación de correo electrónico se vale del sistema de dominio para resolver el problema de un nombre, especifica que la respuesta debe ser la dirección de una máquina que *intercambia correo*. Una aplicación de acceso remoto especifica que busca la aplicación IP de una máquina. Es importante entender lo siguiente:

[4] Es interesante el hecho de que CNRI también registró el uso de *nri . reston . va . us*.

Un nombre dado puede transformarse en más de un aspecto en el sistema de dominio. El cliente especifica el tipo de aspecto deseado cuando resuelve el problema de un nombre, y el servidor devuelve objetos de ese tipo.

Además de especificar el tipo de respuesta buscado, el sistema de dominio permite al cliente especificar la familia de protocolo que se utilizará. El sistema de dominio divide el conjunto completo de nombres en *clases*, lo que permite a una sola base de datos almacenar transformaciones para varios conjuntos de protocolos.[5]

La sintaxis de un nombre no determina qué tipo de objeto nombra o la clase del conjunto de protocolos. Sobre todo, el número de etiquetas en un nombre no determina si el nombre se refiere a un objeto en particular (una máquina) o a un dominio. Así, en nuestro ejemplo, es posible tener una máquina llamada

gwen . purdue . edu

aun cuando

cs . purdue . edu

nombre a un subdominio. Podemos resumir este importante punto de la siguiente manera:

No se puede distinguir el nombre de subdominio del nombre de objetos particulares o del tipo de objetos utilizando sólo la sintaxis del nombre de dominio.

22.10 Asociación de nombres de dominio en direcciones

Además de las reglas para la sintaxis del nombre y la delegación de autoridad, el esquema de nombres de dominio incluye un sistema distribuido, confiable y de propósito general para asociar nombres en direcciones. El sistema está distribuido en el sentido técnico, esto significa que un conjunto de servidores, que opera en varias localidades de manera conjunta, resuelve el problema de la asociación de nombres en direcciones. Es eficiente en el sentido de que la mayor parte de los nombres se puede asociar localmente; sólo unos pocos requieren tráfico de red de redes. Es de propósito general puesto que no se encuentra restringido a nombres de máquina (aun cuando nosotros utilizaremos este ejemplo por ahora). Por último, es confiable ya que una sola falla de una máquina prevendrá al sistema para que opere correctamente.

El mecanismo de dominio para la asociación de nombres en direcciones consiste en sistemas independientes y cooperativos llamados *servidores de nombres*. Un servidor de nombres es un programa servidor que ofrece la asociación nombre-a-dirección, asociando los nombres de dominio en direcciones IP. A menudo, el software servidor se ejecuta en un procesador dedicado y a la máquina se le conoce como servidor de nombre. El software cliente llamado, un *solucionador de nombres* (*name resolver*), utiliza uno o más servidores de nombre cuando traduce un nombre.

[5] En la práctica, pocos servidores de dominio utilizan varios conjuntos de protocolos.

La forma más fácil de entender cómo trabaja un servidor de dominio es imaginándolo como una estructura de árbol que corresponde a la jerarquía nombrada, como se muestra en la figura 22.3. La raíz del árbol es un servidor que reconoce el dominio de nivel superior y sabe qué servidor resuelve cada dominio. Teniendo un nombre por resolver, la raíz puede resolver el servidor correcto para este nombre. En el siguiente nivel, un conjunto de servidores de nombre proporciona respuestas para un dominio de nivel superior (por ejemplo, *edu*). Un servidor en este nivel sabe qué servidor puede resolver cada uno de los subdominios bajo su dominio. En el tercer nivel del árbol, el servidor de nombres proporciona respuestas para el subdominio (por ejemplo, *purdue* bajo *edu*). El árbol conceptual continua con un servidor en cada nivel para el que se ha definido un subdominio.

Los enlaces en el árbol conceptual no indican conexiones de red física. De hecho, muestran qué otros servidores de nombres conoce y contacta un servidor dado. El servidor por sí mismo puede localizarse en una localidad cualquiera dentro de una red de redes. De esta manera, el árbol de servidores es una abstracción que emplea una red de redes para comunicarse.

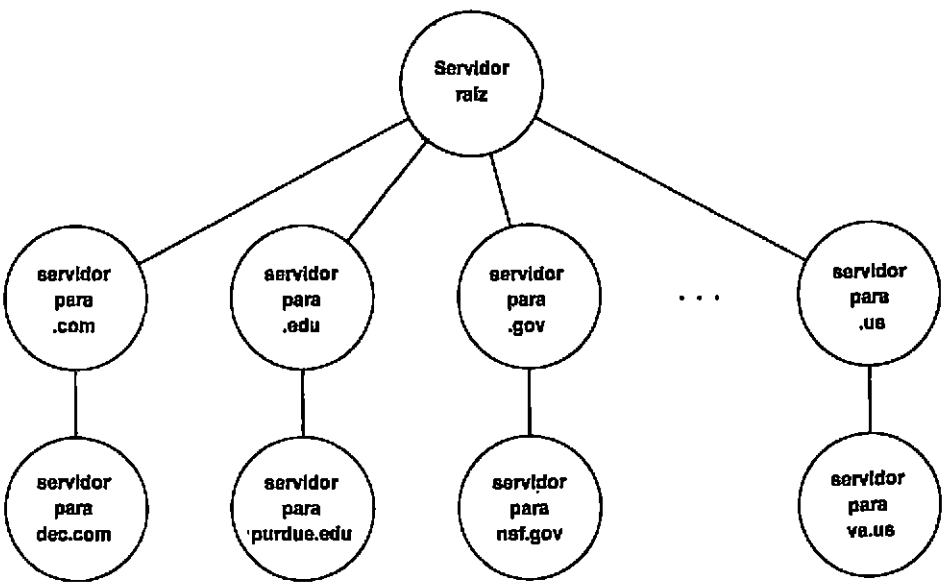


Figura 22.3 Arreglo conceptual del servidor de nombres de dominio en un árbol que corresponde a la jerarquía de nombre. En teoría, cada servidor conoce la dirección de todos los servidores de bajo nivel para todos los subdominios dentro del dominio que maneja.

Si los servidores en el sistema de dominio trabajaran exactamente como lo sugiere nuestro sencillo modelo, la relación entre la conectividad y la autorización sería demasiado simple. Cuando la autoridad está garantizada para un subdominio, la organización que lo solicita necesita establecer un servidor de nombres de dominio para este subdominio y enlazarlo dentro del árbol.

En la práctica, la relación entre una jerarquía de nombres y el árbol de nombres no resulta tan sencilla como nuestro modelo lo plantea. El árbol de servidores tiene pocos niveles pues un sólo servidor físico puede contener toda la información para partes extensas de una jerarquía de nombres. En particular, las organizaciones a menudo reúnen información de todos los subdominios desde un solo servidor. La figura 22.4 muestra una organización más realista de servidores para la jerarquía de nombres de la figura 22.2.

Un servidor raíz contiene información acerca de la raíz y de dominios de nivel superior y cada organización utiliza un sólo servidor para sus nombres. Dado que el árbol de servidores es poco profundo, en la mayor parte de los casos dos servidores necesitan contactarse para resolver un nombre como *xinu . cs . purdue . edu*: el servidor raíz y el servidor para el dominio *purdue . edu* (esto quiere decir que el servidor raíz sabe qué servidor maneja *purdue.edu* y toda la información de dominio reside en un servidor).

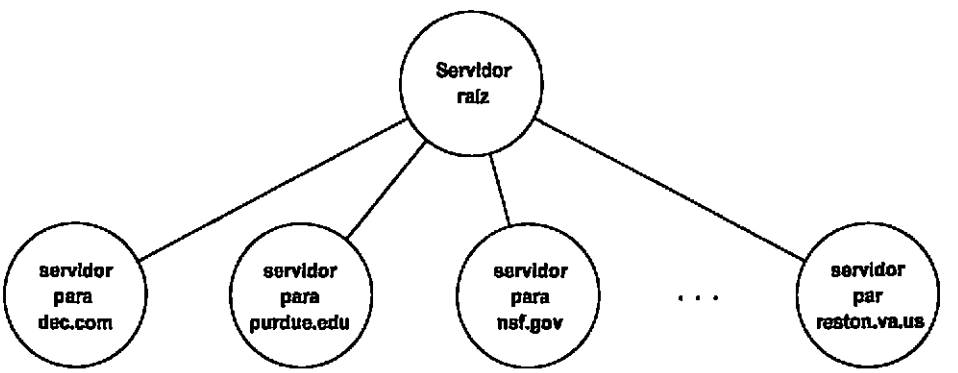


Figura 22.4 Una organización realista de los servidores para la jerarquía de nombres de la figura 22.2. Dado que el árbol es extenso y plano, pocos servidores necesitan contactarse cuando se resuelve un nombre.

22.11 Resolución de nombres de dominio

Aun cuando el árbol conceptual permite entender la relación entre servidores fácilmente, mantiene ocultos varios detalles sutiles. Considerar el algoritmo de resolución de nombres nos ayudará a explicar el proceso. Conceptualmente, la resolución de nombres de dominio procede de arriba hacia abajo, comenzando con el servidor de nombre raíz y siguiendo luego hacia los servidores localizados en las ramas del árbol. Hay dos formas de utilizar el sistema de nombres de dominio: contactar un servidor de nombres cada vez o solicitar al sistema de servidores de nombres que realice la traducción completa. En cada caso, el software cliente forma una solicitud de nombres de dominio que contiene el nombre a resolver, una declaración sobre la clase del nombre, el tipo de respuesta deseada y un código que especifica si el servidor de nombres debe traducir el nombre completamente. Se envía la solicitud a un servidor de nombre para su resolución.

Cuando un servidor de nombres de dominio recibe una solicitud, verifica si el nombre señala un subdominio sobre el cual tenga autoridad. Si así es, traduce el nombre a una dirección de acuerdo con su base de datos y anexa una respuesta a la solicitud, antes de enviarla de regreso al cliente. Si el servidor de nombres no puede resolver el problema del nombre completamente, verifica qué tipo de interacción especificó el cliente. Si el cliente solicita una traducción completa (una *resolución recursiva* en la terminología de nombre de dominio), el servidor se pone en contacto con un servidor de nombres de dominio que pueda resolver el problema del nombre y devuelve la respuesta al cliente. Si el cliente solicita una resolución no recursiva (*resolución iterativa*), el servidor de nombres no puede dar una respuesta. Se genera una réplica que especifica el nombre del servidor que el cliente deberá de contactar la próxima vez para resolver el problema del nombre.

¿Cómo encuentra un cliente un servidor de nombres para comenzar la búsqueda? ¿Cómo encuentra un servidor de nombres a otros servidores de nombres que puedan responder a las solicitudes que él no puede responder? La respuesta es sencilla. Un cliente debe saber cómo contactar al último servidor de nombre. Para asegurarse de que el servidor de nombres de dominio puede alcanzar a otros, el sistema de dominio requiere que cada servidor conozca la dirección del último servidor en la raíz.[6] Además, un servidor podría conocer la dirección de un servidor para el dominio de un nivel inmediatamente superior (llamado *padre*).

Los servidores de nombres de dominio utilizan un puerto de protocolo bien conocido para toda comunicación, así, los clientes saben cómo comunicarse con un servidor una vez que conocen la dirección IP de la máquina que se conecta al servidor. No hay una forma estándar de que los anfitriones localicen una máquina en el entorno local, el cual corre un servidor de nombre; esto se encuentra abierto para quien diseñe el software cliente.[7]

En algunos sistemas, la dirección de la máquina que proporciona el servicio de nombres de dominio está dentro de la frontera de los programas de aplicación en el tiempo de compilación, mientras que en otros la dirección se encuentra configurada dentro del sistema operativo en el arranque. En otros más, el administrador coloca la dirección de un servidor en un archivo en almacenamiento secundario.

22.12 Traducción eficiente

Aun cuando podría parecer natural resolver las solicitudes trabajando hacia abajo del árbol de servidores de nombres, esto podría resultar ineficiente por tres razones. En primer lugar, muchas resoluciones de problemas de nombres se refieren a nombres locales, éstos se encuentran dentro de la misma subdivisión de espacio de nombres que la máquina desde la que se origina la solicitud. Seguir una trayectoria a través de la jerarquía para contactar a la autoridad local sería ineficiente. En segundo lugar, si cada resolución de nombres comienza siempre por contactar al nivel más alto de la jerarquía, la máquina en este punto podría sobrecargarse. En tercer lugar, las fallas de la máquina en el nivel superior de la jerarquía deben prevenir la resolución de problemas de nombres, aun cuando la autoridad local pueda resolver el nombre. La jerarquía de números telefónicos mencionada al comienzo, puede ayudarnos a explicar esto. Aun cuando los números telefónicos son asig-

[6] Por razones de confiabilidad, hay varios servidores para cada nodo en el árbol de servidores de dominio; el servidor raíz es respaldado para que proporcione un balance de carga.

[7] Para un posible método en relación a esto, ver BOOTP/DHCP en el capítulo 21.

nados jerárquicamente, éstos son resueltos dentro de una tendencia de abajo hacia arriba. Como casi todas las llamadas telefónicas son locales, éstas las puede resolver la central telefónica local sin realizar búsquedas en la jerarquía. Además, las llamadas dentro de un código de área dado pueden resolverse sin contactar localidades fuera del código de área. Cuando se aplica a los nombres de dominio, estas ideas conducen a un mecanismo de resolución de problemas de nombres de dos etapas que preserva la jerarquía administrativa pero permite la traducción eficiente.

Hemos dicho que la mayor parte de las solicitudes a los servidores de nombres se refiere a nombres locales. En el proceso de resolución de nombres de dos etapas, la resolución comienza con el servidor de nombres local. Si el servidor local no puede resolver el nombre, la solicitud deberá enviarse hacia otro servidor en el sistema de dominio.

22.13 Desempeño del cache: la clave de la eficiencia

El costo de una búsqueda para nombres no locales puede ser muy alto si se resuelve enviar cada solicitud hacia el servidor raíz. Incluso si las solicitudes pueden ir directamente hacia el servidor que tiene autoridad para el nombre, la búsqueda de nombres puede representar una pesada carga para una red de redes. Así, para mejorar el desempeño global de un sistema servidor de nombres, es necesario reducir los costos de búsqueda para nombres no locales.

Los servidores de nombres de Internet utilizan una memoria inmediata de nombres (*name caching*) para optimizar los costos de búsqueda. Cada servidor mantiene una memoria inmediata de los nombres utilizados más recientemente, así como un registro de dónde fue obtenida la información para la asociación de nombres. Cuando un cliente interroga a un servidor a fin de resolver el problema de un nombre, el servidor verifica primero si tiene autoridad para el nombre de acuerdo con el procedimiento estándar. Si no es así, el servidor verifica su memoria inmediata para ver si el problema del nombre se resolvió recientemente. Los servidores reportan la información almacenada en memoria inmediata a los clientes, pero la marcan como una asignación *no autorizada* y entregan el nombre de dominio del servidor, *S*, desde el cual obtiene la asignación. El servidor local también envía información adicional que le indica al cliente la asignación entre *S* y una dirección IP. De esta manera, los clientes reciben respuestas rápidamente, pero la información podría no estar actualizada. Si la eficiencia es importante, el cliente elegirá aceptar la respuesta no autorizada y continuar. Si la seguridad es importante, el cliente seleccionará contactar a la autoridad y verificar que la asignación entre el nombre y la dirección siga siendo válida.

El procedimiento mediante el uso de memoria inmediata trabaja bien en el sistema de nombres de dominio pues las asignaciones de nombres a direcciones cambian con poca frecuencia. Sin embargo, éstas se modifican. Si el servidor capturó la información la primera vez que le fue solicitada y nunca la cambió. las entradas de información de la memoria inmediata podrían estar incorrectas. Para mantener la memoria inmediata con información correcta, los servidores cronometran cada entrada y suprimen las entradas que excedan un tiempo razonable. Cuando el servidor es interrogado respecto a cierta información luego de que ha removido las entradas de información de la memoria inmediata, debe volver a la fuente autorizada y obtener la asignación de nuevo. Algo muy importante, los servidores no aplican un sólo límite de tiempo fijo para todas las entradas, pero permiten a la autoridad de una entrada configurar su límite de tiempo. Cada vez que una autoridad responde a una solicitud, incluye un valor de *Tiempo de Vida* (*Time To Live o* TTL) en la respuesta, el cual especifica qué tanto se garantiza la conservación de la asignación. Así, las autoridades pueden reducir la sobrecarga en la red especificando límites de tiempo largos para entradas

en las que esperan cambios poco frecuentes, mientras que especifican límites de tiempo cortos para entradas en las que se esperan cambios con frecuencia.

El procedimiento de memoria inmediata es importante en los anfitriones así como en los servidores de nombres de dominio local. Muchos sistemas de tiempo compartido ejecutan una forma compleja de resolución de códigos, la cual trata de proporcionar una mayor eficiencia que el sistema de servidor. El anfitrión baja la base de datos completa de nombres y direcciones desde un servidor de nombres de dominio local en el arranque, mantiene su propia memoria inmediata de nombres utilizados recientemente y utiliza el servidor sólo cuando los nombres no se encuentran. Desde luego, un anfitrión que mantiene una copia de la base de datos del anfitrión local debe verificar su información con el servidor de manera periódica para obtener nuevas transformaciones y el anfitrión debe retirar las entradas de su memoria inmediata luego de que éstas quedan sin validez. Sin embargo, muchas localidades tienen algunos problemas para mantener la consistencia pues los nombres de dominio cambian con poca frecuencia.

Conservar una copia de la base de datos del servidor local en cada anfitrión tiene varias ventajas. Obviamente, hace que la resolución de un nombre en el anfitrión local sea muy rápida pues esto significa que el anfitrión puede resolver los nombres sin ninguna actividad de red. También significa que la localidad cuenta con protección en caso de que el servidor de nombres local falle. Por último, reduce la carga computacional en el servidor de nombres y hace posible que un servidor dado proporcione nombres para más máquinas.

22.14 Formato de los mensajes del servidor de dominios

Observar en detalle el intercambio de mensajes entre clientes y servidores de nombres de dominio nos ayudará a esclarecer cómo opera el sistema desde el punto de vista de un programa de aplicación común. Supongamos que un usuario invoca un programa de aplicación y proporciona el nombre de la máquina con la que la aplicación debe comunicarse. Antes de poder utilizar protocolos como el TCP o el UDP para comunicarse con la máquina especificada, el programa de aplicación debe encontrar la dirección IP de la máquina. Debe pasar el nombre de dominio a la máquina local capaz de resolver el nombre y solicitar una dirección IP. El solucionador local verifica su memoria inmediata y devuelve la respuesta si hay alguna presente. Si el solucionador local no tiene una respuesta, formatea un mensaje y lo envía al servidor (esto es, se convierte en un cliente). Aun cuando nuestro ejemplo sólo comprende un nombre, el formato de mensaje permite a un cliente hacer varias solicitudes en un solo mensaje. Cada uno consiste en un nombre de dominio para el que el cliente busca un dirección IP, una especificación de la clase de solicitud (es decir, una *red de redes*) y el tipo de objeto deseado (esto es, una *dirección*). El servidor responde con la devolución de un mensaje similar que contiene respuestas a las solicitudes para las que el servidor tiene asignaciones. Si el servidor no puede responder a todas las preguntas, la respuesta contendrá información acerca de otro servidor de nombres que el cliente puede contactar para encontrar la respuesta.

Las respuestas también contienen información acerca de servidores que están autorizados para responder y las direcciones IP de tales servidores. La figura 22.5 muestra el formato del mensaje. Como se muestra en la figura, cada mensaje comienza con un encabezado fijo. El encabezado contiene el campo único *IDENTIFICATION* (*IDENTIFICACIÓN*) que el cliente utiliza para confrontar las respuestas solicitadas y el campo *PARAMETER* (PARÁMETRO) que especifica la operación solicitada y el código de respuesta. La figura 22.6 proporciona la interpretación de los bits en el campo *PARAMETER*.

0	16 31
IDENTIFICACIÓN	PARÁMETRO
NÚMERO DE SOLICITUDES	NÚMERO DE RESPUESTAS
NÚMERO DE AUTORIDAD	NÚMERO DE ADICIONAL
SECCIÓN DE SOLICITUDES ...	
SECCIÓN DE RESPUESTAS ...	
SECCIÓN DE AUTORIDAD ...	
SECCIÓN DE INFORMACIÓN ADICIONAL ...	

Figura 22.5 Formato de un mensaje de servidor de nombre de dominio. Las secciones de interrogación, respuesta, autoridad e información adicional son de longitud variable.

Bit del campo PARÁMETRO	Significado
0	Operación: 0 Solicitud 1 Respuesta
1-4	Tipo de solicitud: 0 Estándar 1 Inversa 2 Terminado 1 (obsoleto) 3 Terminado 2 (obsoleto)
5	activado si se tiene una respuesta autorizada
6	activado si el mensaje está truncado
7	activado si se desea recursión
8	activado si la recursión está disponible
9-11	Reservado
12-15	Tipo de respuesta 0 Sin error 1 Error de formato en la solicitud 2 Falla en el servidor 3 El nombre no existe

Figura 22.6 Significado de los bits del campo *PARAMETER* (*PARÁMETRO*) en el mensaje del servidor de nombre de dominio. Los bits están numerados de izquierda a derecha, comenzando con el 0.

El campo *NUMBER OF* (*NÚMERO DE*) proporciona un conteo de las entradas en la sesión correspondiente que se presentan en el último mensaje. Por ejemplo, el campo *NUMBER OF QUESTIONS* (*NÚMERO DE SOLICITUDES*) proporciona el conteo de entradas que aparecen en la *QUESTION SECTION* (*SECCIÓN DE SOLICITUDES*) del mensaje.

QUESTION SECTION contiene las solicitudes para las que se desea una respuesta. El cliente llena sólo la sección de solicitud; el servidor devuelve la solicitud y la respuesta en su réplica. Cada solicitud consiste en un *QUERY DOMAIN NAME* (*SOLICITUD DE NOMBRE DE DOMINIO*) seguido por los campos *QUERY TYPE* (*TIPO DE SOLICITUD*) y *QUERY CLASS*, (*CLASE DE SOLICITUD*) como se muestra en la figura 22.7.

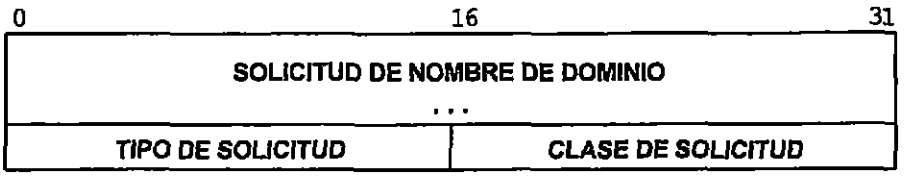


Figura 22.7 Formato de la entrada de información en *QUESTION SECTION*, (*SECCIÓN DE SOLICITUD*) del mensaje de servidor de dominio. El nombre de dominio tiene una longitud variable. El cliente llena la solicitud; el servidor la devuelve junto con la respuesta.

Aun cuando el campo *QUERY DOMAIN NAME* tiene una longitud variable, veremos en la siguiente sección que la representación interna de los nombres de dominio hace posible, para el receptor, conocer la longitud exacta. El *QUERY TYPE* codifica el tipo de solicitud (por ejemplo, si la solicitud se refiere a un nombre de máquina o a una dirección de correo). El campo *QUERY CLASS* permite que los nombres de dominio se utilicen para objetos arbitrarios debido a que los nombres oficiales de Internet son sólo de una clase. Debe notarse que, aun cuando el diagrama en la figura 22.5 sigue nuestra convención de mostrar los formatos en múltiplos de 32 bits, el campo *QUERY DOMAIN NAME* puede contener un número arbitrario de objetos. No se utilizan rellenos. Además, los mensajes hacia o desde un servidor de nombres de dominio pueden contener un número impar de octetos.

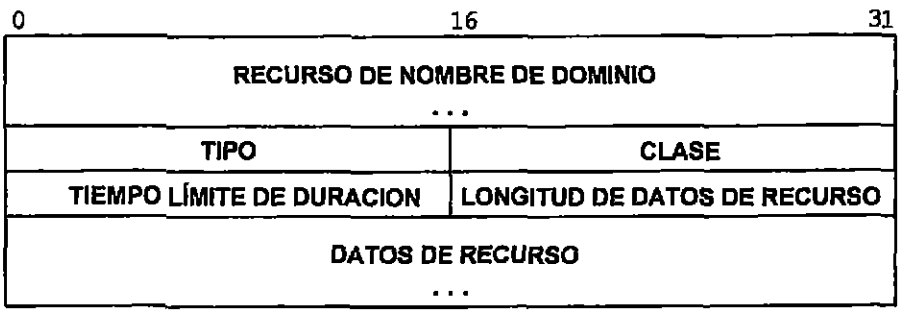


Figura 22.8 Formato de un registro de recurso utilizado en la última sección de los mensajes devueltos por el servidor de nombre de dominio.

En un mensaje de servidor de nombres de dominio, cada uno de los campos *ANSWER SECTION*, (*SECCIÓN DE RESPUESTAS*) *AUTHORITY SECTION* (*SECCIÓN DE AUTORIDAD*) y de la *ADDITIONAL INFORMATION SECTION* (*SECCIÓN DE INFORMACIÓN ADICIONAL*) consiste en un conjunto de *registros de recursos* que describen los nombres de dominio y las transformaciones. Cada registro de recurso describe un nombre. La figura 22.8 muestra el formato.

El campo *RESOURCE DOMAIN NAME* (*RECURSO DE NOMBRE DE DOMINIO*) contiene el nombre de dominio al que este registro de recursos se refiere. El campo *TYPE* (*TIPO*) especifica el tipo de datos incluidos en el registro de recurso; el campo *CLASS* (*CLASE*) especifica la clase de datos. El campo *TIME TO LIVE* (*TIEMPO LÍMITE DE DURACIÓN*) contiene un entero que especifica en número de segundos que la información en este registro de recursos se mantendrá en memoria inmediata. Ésta es utilizada por clientes que han solicitado la asignación de un nombre y desean capturar el resultado. Los dos últimos campos contienen el resultado de la asignación, con el campo *RESOURCE DATA LENGTH* (*LONGITUD DE DATOS DE RECURSOS*) especificando el conteo de octetos en el campo *RESOURCE DATA (DATOS DE RECURSO)*.

22.15 Formato de nombre comprimido

Cuando se representan en un mensaje, los nombres de dominio son almacenados como una secuencia de etiquetas. Cada etiqueta comienza con un octeto que especifica su longitud. Así, el receptor reconstruye un nombre de dominio leyendo repetidamente la longitud de un octeto, *n*, y entonces lee una etiqueta con *n* octetos de longitud. Una longitud de octeto que contenga cero marca el fin del nombre.

Los servidores de nombres de dominio frecuentemente devuelven varias respuestas a una solicitud y en muchos casos los sufijos de dominio se sobreponen. Para conservar espacio en el paquete de réplica, el servidor de nombres comprime los nombres, almacenando sólo una copia de cada nombre de dominio. Cuando se extrae un nombre de dominio para un mensaje, el software cliente debe verificar cada segmento del nombre para ver si está formado por una cadena literal (en el formato del contador de un octeto seguido por los caracteres que forman el nombre) o un apuntador hacia una cadena literal. Cuando encuentra un apuntador, el cliente debe seguir al apuntador hacia el nuevo lugar en el mensaje para encontrar el resto del nombre.

Los apuntadores siempre se presentan al comienzo de los segmentos y están codificados en el octeto de conteo. Si los dos bits superiores del segmento de ocho bits del campo de conteo están puestos en uno, el cliente debe tomar los siguientes 14 bits como un apuntador entero. Si los dos bits superiores están puestos en cero, los siguientes 6 bits especifican el número de caracteres en la etiqueta que siguen del octeto de conteo.

22.16 Abreviatura de nombres de dominio

La jerarquía de número telefónico ilustra otra característica útil de resolución local, la *abreviatura de nombre* (*name abbreviation*). La abreviatura de nombre proporciona un método para hacer más cortos los nombres cuando el proceso para resolverlos puede proporcionar automáticamente parte del nombre. Por lo general, un suscriptor omite el código de área cuando hace una llamada a un nú-

mero telefónico local. Los dígitos resultantes forman un nombre abreviado asumiendo que se permanece dentro del mismo código de área que el teléfono del suscriptor. La abreviatura también funciona para los nombres de máquina. Tomemos un nombre como *xyz*, el proceso de resolución puede asumir que se ubica en la misma autoridad local que la máquina en la que se está resolviendo el problema del nombre. Así, la máquina que resuelve el problema del nombre puede proporcionar de manera automática partes faltantes del nombre. Por ejemplo, dentro del Departamento de Ciencias Computacionales en Purdue, el nombre abreviado

xinu

es equivalente al nombre de dominio completo

xinu . cs . purdue . edu

La mayor parte del software cliente implanta abreviaturas con una *lista de sufijos de dominio*. El administrador de red local configura una lista de posibles sufijos para añadirse a los nombres durante la búsqueda. Cuando una máquina que resuelve problemas de nombres encuentra un nombre, recorre la lista, añadiendo cada sufijo y tratando de observar el nombre resultante. Por ejemplo, la lista de sufijos para el Departamento de Ciencias Computacionales incluye:

. cs . purdue . edu
. cc . purdue . edu
. purdue . edu
null

Así, el proceso de resolución de nombres local primero añade *cs . purdue . edu* al nombre *xinu*. Si esta búsqueda falla, añade *cc . purdue . edu* al nombre y observa el resultado. El último sufijo en la lista de ejemplo es una cadena nula, significa que si todos los otros intentos fallan, en el proceso de resolución del nombre se considerará el nombre sin sufijo. Los administradores pueden utilizar la lista de sufijos para hacer la abreviatura conveniente o para restringir los programas de aplicación a los nombres locales.

Dijimos que el cliente tiene la responsabilidad de la expansión de las abreviaturas, pero se debe enfatizar que estas abreviaturas no son en sí parte del sistema de nombres de dominio. El sistema de dominio sólo permite búsquedas de nombres de dominio completamente especificados. Como consecuencia, los programas que dependen de las abreviaturas pueden no trabajar correctamente fuera del ambiente en el que se construyeron. Podemos resumir lo siguiente:

El sistema de nombres de dominio sólo transforma nombres de dominio completos en direcciones; las abreviaturas no son en sí parte del sistema de dominios, pero son introducidas por el software cliente para hacer los nombres locales convenientes para los usuarios.

22.17 Asociaciones inversas

Dijimos que el sistema de nombres de dominio puede proporcionar transformaciones diferentes a las que convierten nombres de máquina en direcciones IP. Las *solicitudes inversas* permiten que el cliente interrogue a un servidor para hacer transformaciones en "sentido inverso", estableciendo una respuesta y generando la interrogación que produciría esta respuesta. Por supuesto, no todas las respuestas tienen una pregunta única. Cuando esto sucede, un servidor podría no ser capaz de proporcionarla. Aun cuando las solicitudes inversas han sido parte del sistema de dominios desde que fueron especificadas por primera vez, éstas por lo general no son utilizadas pues con frecuencia no hay manera de encontrar un servidor que pueda resolver una solicitud sin buscar en todo el conjunto de servidores.

22.18 Búsquedas de apuntador

Una forma de transformación inversa es tan necesaria que el sistema de dominios soporta un dominio especial y una forma especial de interrogación llamada *búsquedas de apuntador* (*pointer queries*) para responderla. En una búsqueda de apuntador, la interrogación presentada al servidor de nombres de dominio especifica una dirección IP codificada como cadena imprimible en forma de nombre de dominio (esto es, una representación textual de dígitos separados por puntos). Una búsqueda de apuntador solicita al servidor de nombres que devuelva el nombre de dominio correcto para la máquina con la dirección especificada. Las búsquedas de apuntador son especialmente útiles para máquinas sin disco pues permiten que el sistema obtenga un nombre de alto nivel presentando sólo una dirección IP. (Ya hemos visto, en el capítulo 6, cómo puede obtener una máquina sin disco su dirección IP.)

Las búsquedas de apuntador no son difíciles de generar. Si pensamos en una dirección IP escrita en forma decimal con puntos, tiene el siguiente formato

aaa . bbb . ccc . ddd

Para formar una búsqueda de apuntador, el cliente reordena la representación decimal con puntos de la dirección como una cadena con la forma:

ddd . ccc . bbb . aaa . in-addr . arpa

La nueva forma es un nombre en el dominio especial *in-addr.arpa*.[8] Como el servidor de nombres local no puede ser la autoridad para el dominio *arpa* o el dominio *in . addr . arpa*, podría necesitar establecer contacto con otro servidor de nombres para completar la resolución. Para hacer eficiente la resolución de la búsqueda de apuntador, el servidor de dominio de raíz de Internet mantiene una base de datos de direcciones IP válidas, junto con información acerca de los servidores de nombres de dominio que pueden resolver cada dirección.

[8] Los octetos de las direcciones IP deben invertirse cuando forman un nombre de dominio dado que las direcciones IP tienen el octeto más significativo al principio, mientras que los nombres de dominio tienen el octeto menos significativo al principio.

22.19 Tipos de objetos y contenido del registro de recursos

Hemos mencionado que el sistema de nombres de dominio puede usarse para traducir un nombre de dominio a una dirección de intercambio de correo, así como para traducir un nombre de anfitrión a una dirección IP. El sistema de dominio es general en el sentido de que puede utilizarse para jerarquías arbitrarias de nombres. Por ejemplo, se podría decidir almacenar los nombres de servicios computacionales disponibles junto con una transformación de cada nombre en un número telefónico para llamar y encontrar el servicio correspondiente. O se podrían almacenar nombres de productos de protocolo junto con una transformación en nombres y direcciones de vendedores que vendan tales productos.

Recordemos que el sistema se adapta a una gran variedad de transformaciones, incluyendo un elemento *type* (*tipo*)en cada registro de recurso. Cuando envía una solicitud, un cliente debe especificar el tipo en su solicitud;[9] los servidores especifican el tipo de datos en todos los registros de recurso que devuelven. El tipo determina el contenido del registro de recursos de acuerdo con la tabla que se muestra en la figura 22.9.

Tipo	Significado	Contenido
A	Dirección de anfitrión	Dirección IP de 32 bits
CNAME	Nombre de oficina	Nombre de dominio de oficina para un alias
HINFO	CPU&OS	Nombre de CPU y sistema operativo
MINFO	Información de buzón	Información sobre buzón o lista de correo
MX	Transportador de correo	Preferencia de 16 bits y nombre del anfitrión que actúa como transportador de correo para el dominio
NS	Servidor de nombre	Nombre de un servidor autorizado para el dominio
PTR	Puntero	Nombre de dominio (como un enlace simbólico)
SOA	Comienzo de autoridad	Varios archivos que especifican qué partes de la jerarquía de nombres implanta un servidor
TXT	Texto arbitrario	Cadena de texto en ASCII sin interpretación

Figura 22.9 Tipos de registro de recurso en el sistema de nombre de dominio.

La mayor parte de los datos es de tipo *A*, esto significa que consisten en el nombre del anfitrión conectado a Internet así como con la dirección IP del anfitrión. El segundo tipo de dominio más utilizado, *MX*, está asignado a nombres utilizados para intercambiar correo electrónico (mail exchange). Permite que una localidad especifique varios anfitriones que sean capaces de aceptar correo. Cuando se envía correo electrónico, el usuario especifica una dirección de correo electrónico en la forma *usuario@parte-dominio*. El sistema de correo utiliza el sistema de nombres de dominio para resolver *parte-dominio* con interrogadores tipo *MX*. El sistema de dominio devuelve un conjunto de registros de recurso de los que cada uno contiene un campo de preferencia y un nombre de dominio del anfitrión. El sistema de correo pasa a través del conjunto de la mayor a la menor preferencia (los números más bajos significan una preferencia mayor). Para cada registro de recurso *MX*, la máquina

[9] Los interrogadores pueden especificar unos pocos tipos adicionales (por ejemplo, hay un tipo de interrogador que solicita todos registros de recursos.

que maneja la correspondencia extrae el nombre de dominio y utiliza un interrogador (query) tipo *A* para resolver el problema del nombre en una dirección IP. Luego trata de contactar al anfitrión y entregar el correo. Si el anfitrión no está disponible, el manejador de correo intentará hacerlo con otros anfitriones de la lista.

Para hacer la búsqueda eficiente, un servidor siempre devuelve asignaciones adicionales que conoce en la *ADDITIONAL INFORMATION SECTION* (*SECCIÓN DE INFORMACIÓN ADICIONAL*) de una respuesta. En el caso del registro *MX*, un servidor de dominio puede utilizar la *ADDITIONAL INFORMATION SECTION* a fin de devolver un registro de recurso tipo *A* para el nombre de dominio reportado en *ANSWER SECTION* (*SECCIÓN DE RESPUESTAS*). Hacer esto reduce sustancialmente el número de solicitudes que un manejador de correo envía a su servidor de dominio.

22.20 Obtención de autoridad para un subdominio

Antes de que una institución obtenga autorización oficial para un dominio de segundo nivel, debe estar de acuerdo en operar un servidor de nombres de dominio que cumpla con los estándares de Internet. Por supuesto un servidor de nombres de dominio debe obedecer el estándar de protocolo que especifica los formatos de mensaje y las reglas para responder a las solicitudes. El servidor también debe conocer las direcciones de los servidores que manejan cada subdominio (si existe alguno) así como la dirección del ultimo servidor de raíz.

En la práctica, el sistema de dominios es mucho más complejo de lo que hemos bosquejado. En la mayor parte de los casos, un sólo servidor físico puede manejar más de una parte de la jerarquía de nombres. Por ejemplo, un servidor de nombres en la Universidad de Purdue maneja tanto el segundo nivel de dominio *purdue . edu* como el dominio geográfico *laf . in . us*. Un sub-árbol de nombres administrado por un servidor de nombres dado forma una *zona de autoridad*. Otra complicación práctica se debe a que los servidores tienden a manejar muchas solicitudes, aun cuando algunas solicitudes se lleva mucho tiempo resolverlas. Por lo general, los servidores soportan actividad concurrente, permitiendo trabajar y proceder con las últimas solicitudes mientras las primeras se están procesando. Manejar las solicitudes de manera concurrente es especialmente importante cuando el servidor recibe una solicitud recursiva que lo obliga a enviar la solicitud a otro servidor para su resolución.

La implantación de servidores también es complicada pues la autoridad de Internet requiere que la información en todos los servidores de dominio sea respaldada. La información debe aparecer en por lo menos dos servidores que no operen en la misma computadora. En la práctica, los requerimientos son muy restrictivos: los servidores no deben tener punto común alguno de falla. Evitar los puntos comunes de falla significa que dos servidores de nombre no pueden estar conectados a la misma red; éstos no pueden obtener suministro eléctrico de la misma fuente. Así, para cumplir con los requerimientos, una localidad debe encontrar al menos otra localidad con la que acuerde operar un respaldo del servidor de nombres. Por supuesto, en cualquier punto del árbol de servidores, un servidor debe saber cómo localizar los servidores de nombre primarios y de respaldo para los subdominios y debe dirigir sus solicitudes hacia un servidor de nombres de respaldo si el servidor primario no está disponible.

22.21 Resumen

El sistema jerárquico de nombres permite delegar la autoridad para los nombres, es decir adaptarse a un conjunto arbitrariamente extenso de nombres sin saturar una localidad central con tareas administrativas. Aun cuando la resolución de nombres está separada respecto a la delegación de autoridad, es posible crear sistemas de nombres jerárquicos en los que la resolución es un proceso eficiente que comienza en el servidor local aun cuando la delegación de autoridad siempre fluya desde el nivel superior de la jerarquía hacia abajo.

Examinamos el Sistema de Nombres o Nomenclatura de Dominio (DNS) de Internet y dijimos que ofrece un esquema de nombres jerárquico. El DNS se vale de una búsqueda distribuida, mediante la cual los servidores de nombre de dominio transforman cada nombre de dominio en una dirección IP o en una dirección de intercambio de correo. Los clientes comienzan a tratar de resolver los nombres de manera local. Cuando el servidor local no puede resolver el nombre, el cliente debe trabajar a través del árbol de servidores de nombres iterativamente o solicitar al servidor de nombres local que lo haga recursivamente. Por último, dijimos que el sistema de nombres de dominio soporta una variedad de asignaciones, incluyendo asignaciones desde direcciones IP hacia nombres de alto nivel.

PARA CONOCER MÁS

Mockapetris (RFC 1034) analiza los nombres de dominio de Internet en general, exponiendo la filosofía general, en tanto que Mockapetris (RFC 1035) proporciona un estándar de protocolo para el sistema de nombres de dominio. Mockapetris (RFC 1101) trata el uso del sistema de nombres de dominio para codificar nombres de red y propone extensiones útiles para otras transformaciones. Versiones anteriores aparecen en Mockapetris (RFC 882, 883 y 973). Postel y Reynolds (RFC 920) establecen los requerimientos que debe cumplir un servidor de nombres de dominio de Internet. Stahl (RFC 1032) proporciona los lineamientos administrativos para establecer un dominio y Lottor (RFC 1033) los lineamientos para operar un servidor de nombres de dominio. Partridge (RFC 974) relaciona los nombres de dominio con el direccionamiento de correo electrónico. Por último, Lottor (RFC 1296) proporciona un interesante resumen del crecimiento de Internet, obtenido mediante un recorrido por el árbol de nombres de dominio.

EJERCICIOS

22.1 Los nombres de máquina no deben determinarse en el sistema operativo dentro del tiempo de compilación. Explique por qué.

22.2 ¿Preferiría utilizar una máquina que obtenga su nombre de un archivo remoto o de un servidor de nombre? ¿Por qué?

22.3 ¿Por qué cada servidor de nombres debe conocer la dirección IP de su padre, en lugar del nombre de dominio del mismo?

22.4 Construya un esquema de nombres que tolere cambios en la jerarquía de nombres. Como ejemplo, considere dos grandes compañías, cada una tiene una jerarquía de nombres independiente, y suponga que las compañías se fusionan. ¿Puede usted hacer los arreglos necesarios para que los nombres anteriores se mantengan trabajando correctamente?

22.5 Lea el estándar y encuentre cómo utiliza el sistema de nombres de dominio registros *SOA*.

22.6 El sistema de nombres de dominio de Internet puede adaptarse también a los nombres de buzones. Averigüe cómo.

22.7 El estándar sugiere que, cuando un programa necesita encontrar el nombre de dominio asociado con una dirección IP, primero debe enviar una solicitud inversa al servidor local y, después, utilizar el dominio *in-addr. arpa* sólo si éste falla. ¿Por qué?

22.8 ¿Cómo podría adaptar las abreviaturas a un esquema de nombres de dominio? Como ejemplo, muestre dos localidades que estén registradas bajo *. edu* y bajo un servidor de nivel superior. Explique cómo trata cada localidad cada tipo de abreviatura.

22.9 Obtenga la descripción oficial del sistema de nombres de dominio y construya un programa cliente. Considere el nombre *merlin. cs. purdue. edu*.

22.10 Amplíe el ejercicio anterior e incluya un interrogador (*query*) apuntador. Trate de encontrar el nombre de dominio para la dirección *128.10.2.3*.

22.11 Obtenga una copia del programa *nslookup* y encuentre los nombres de los dos ejercicios anteriores.

22.12 Si ampliamos la sintaxis de nombres de dominio a fin de incluir un punto luego de cada dominio de nivel superior, los nombres y las abreviaturas no serían ambiguos. ¿Cuáles son las ventajas y desventajas de la extensión.

22.13 Lea los RFC sobre el sistema de nombres de dominio. ¿Cuáles son los valores máximo y mínimo que un servidor DNS puede almacenar en el campo *TIME-TO-LIVE* de un registro de recurso?

22.14 ¿Debería el sistema de nombres de dominio permitir solicitudes que cumplan con sus requisitos básicos parcialmente (es decir, que utilicen comodines como parte del nombre)? ¿Por qué sí o por qué no?

22.15 El Departamento de Ciencias Computacionales de la Universidad de Purdue decidió colocar la siguiente entrada de registro de recurso de tipo *A* en su servidor de nombre de dominio:

```
localhost.cs.purdue.edu  127.0.0.1
```

Explique qué podría suceder si una localidad remota tratara de ejecutar una función *ping* hacia una máquina con un nombre de dominio *localhost. cs. purdue. edu*.

23

Aplicaciones: acceso remoto (TELNET, Rlogin)

23.1 Introducción

En este capítulo y los cuatro siguientes, continuaremos explorando el enlace de redes mediante el examen de los servicios de alto nivel de la red de redes y los protocolos que la soportan. Tales servicios forman una parte integral del TCP/IP. Determinan cómo perciben los usuarios una red de redes y demuestran el poder de la tecnología.

Aprenderemos que los servicios de alto nivel proporcionan una mayor funcionalidad de comunicación, permiten a los usuarios y a los programas interactuar con servicios automatizados de máquinas remotas y con usuarios remotos. Veremos que los protocolos de alto nivel se implantan con programas de aplicación y aprenderemos cómo es que dependen del nivel de servicios de la red descrito en capítulos anteriores. Este capítulo comienza con un examen del acceso remoto.

23.2 Computación remota interactiva

Ya hemos visto de qué manera el modelo cliente-servidor proporciona servicios computacionales específicos como el servicio de hora del día para varias máquinas. Los protocolos de flujo confiable como el TCP hacen posible el uso interactivo también para las máquinas remotas. Por ejemplo, imaginemos que se construye un servidor que ofrece un servicio de edición remota de texto. Para implantar un servicio de edición, necesitaríamos un servidor que aceptara peticiones para editar un archivo y un cliente que hiciera tales peticiones. Para invocar el servicio de editor remoto, un usua-

rio ejecutaría el programa de cliente. El cliente establecería una conexión TCP de la máquina local al servidor, y entonces comenzaría a enviar las pulsaciones de tecla al servidor y a leer la salida que el servidor mande de respuesta.

¿Cómo se puede generalizar el servicio de edición remota interactiva que imaginamos? El problema con el uso de un servidor para cada servicio computacional es que las máquinas se empantanan rápidamente con los procesos del servidor. Podemos eliminar la mayor parte de los servidores especializados y proporcionar una mayor generalidad permitiendo que el usuario establezca una sesión de acceso en la máquina remota y que entonces ejecute los comandos. Con la infraestructura de *acceso remoto* (*remote login*), los usuarios tienen acceso a todos los comandos disponibles en el sistema remoto, y los diseñadores del sistema no necesitan proporcionar servidores especializados.

Por supuesto, proporcionar acceso remoto puede no ser muy sencillo. Los sistemas computacionales diseñados sin considerar el trabajo en redes esperan obtener sesiones de acceso sólo de un teclado y monitor conectados de manera directa. En una computadora así, añadir un servidor de acceso remoto requiere que se modifique el sistema operativo. Construir un software de cliente interactivo puede también ser difícil. Consideremos, por ejemplo, un sistema que asigna un significado especial a algunas teclas. Si el sistema local interpreta que Control-C significa "abortar el proceso de comandos que se esté ejecutando en ese momento", podría resultar imposible la transferencia de Control-C a la máquina remota. Si el cliente no transfiere Control-C a la localidad remota, podría imposibilitarse el aborto del proceso local del cliente.

A pesar de las dificultades técnicas, los programadores de sistemas han administrado la construcción de software servidor de acceso remoto para la mayor parte de los sistemas operativos y han construido programas de aplicación que actúan como clientes. A menudo, el software de cliente anula la interpretación local de todas las teclas a excepción de una, lo cual permite al usuario interactuar con la máquina remota de la misma manera que lo haría si lo hiciera desde una terminal conectada localmente. Esa sola excepción de tecla proporciona al usuario una forma de escapar del ambiente local y controlar al cliente (por ejemplo, abortarlo). Además, algunos protocolos de acceso remoto reconocen un conjunto de *anfitriones confiables*, permitiendo el acceso remoto de tales anfitriones sin verificar las claves de acceso.

23.3 Protocolo TELNET

El conjunto de protocolos TCP/IP incluye un protocolo de terminal remota sencillo, llamado *TELNET*. TELNET permite al usuario de una localidad establecer una conexión TCP con un servidor de acceso a otro. TELNET transfiere después las pulsaciones de teclado directamente desde el teclado del usuario a la computadora remota como si hubiesen sido hechos en un teclado unido a la máquina remota. TELNET también transporta la salida de la máquina remota de regreso a la pantalla del usuario. El servicio se llama *transparent* (*transparente*) porque da la impresión de que el teclado y el monitor del usuario están conectados de manera directa a la máquina remota.

Aunque TELNET no es sofisticado en comparación con algunos protocolos de terminal remota, se dispone de él ampliamente. El software de cliente TELNET suele permitir que el usuario especifique una máquina remota ya sea dando su nombre de dominio o su dirección IP. Como acepta direcciones IP, TELNET se puede usar con anfitriones aunque no se pueda establecer el en-

lace de un nombre con una dirección (por ejemplo, cuando el software de nombres de dominio se está depurando).

TELNET ofrece tres servicios básicos. El primero, define una *terminal virtual de red* (*network virtual terminal*) que proporciona una interfaz estándar para los sistemas remotos. Los programas clientes no tienen que comprender los detalles de todos los sistemas remotos, se construyen para utilizarse con la interfaz estándar. En el segundo, TELNET incluye un mecanismo que permite al cliente y al servidor negociar opciones, asimismo proporciona un conjunto de opciones estándar (por ejemplo, una de las opciones controla si los datos que se transfieren a través de la conexión se valen del conjunto de caracteres ASCII estándar de siete bits o de un conjunto de caracteres de ocho bits). Por último, TELNET trata con ambos extremos de la conexión de manera simétrica. En particular, TELNET no fuerza la entrada de cliente para que ésta provenga de un teclado, ni al cliente para que muestre su salida en una pantalla. De ésta manera, TELNET permite que cualquier programa se convierta en cliente, Además, cualquier extremo puede negociar las opciones.

En la figura 23.1, se ilustra la forma en que los programas de aplicación implantan un cliente y servidor de TELNET.

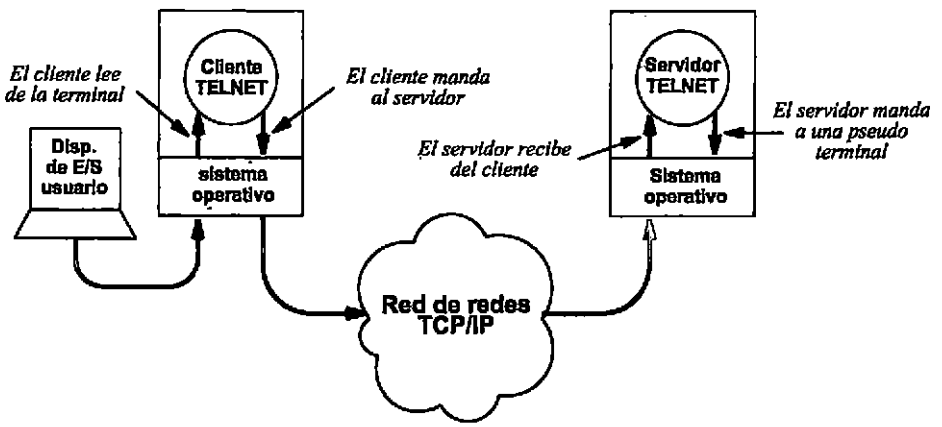


Figura 23.1 Trayectoria de los datos en una sesión de terminal remota con TELNET conforme viaja del teclado del usuario al sistema operativo. La adición de un servidor TELNET a un sistema de tiempo compartido suele requerir la modificación del sistema operativo.

Como se muestra en la figura, cuando un usuario invoca a TELNET, un programa de aplicación en la máquina del usuario se convierte en el cliente. El cliente establece una conexión TCP con el servidor por medio de la cual se comunicarán. Una vez establecida la conexión, el cliente acepta los pulsos de teclado del usuario y los manda al servidor, al tiempo que acepta caracteres de manera concurrente que el servidor regresa y despliega en la pantalla del usuario. El servidor debe aceptar una conexión TCP del cliente y después transmitir los datos entre la conexión TCP y el sistema operativo local.

En la práctica, el servidor es más complejo de lo que muestra la figura pues debe manejar diversas conexiones concurrentes. Normalmente, un proceso de servidor maestro espera nuevas conexiones y crea un nuevo esclavo para manejar cada conexión. De esta forma, el 'servidor de TELNET', que se muestra en la figura 23.1, representa al esclavo que maneja una conexión en particular. La figura no muestra al servidor maestro que está atento a nuevas peticiones, ni se muestra a los esclavos que se encuentran manejando otras conexiones.

Utilizamos el término *pseudo terminal*[1] para describir el punto de entrada del sistema operativo que permite que un programa, que se está corriendo como el servidor TELNET, transfiera caracteres al sistema operativo como si vinieran de un teclado. Es imposible construir un servidor TELNET a menos que el sistema operativo proporcione dicha característica. Si el sistema soporta la abstracción de una pseudo terminal, el servidor TELNET podrá implantarse con programas de aplicación. Cada servidor esclavo conecta una corriente TCP de un cliente a una pseudo terminal en particular.

El arreglo del servidor TELNET para que sea un programa de nivel de aplicación tiene sus ventajas y sus desventajas. La ventaja más obvia es que hace la modificación y el control del servidor más fácil que si el código estuviera enclavado en el sistema operativo. La desventaja evidente es su ineficiencia. Cada pulso de teclado viaja del teclado del usuario a través del sistema operativo hacia el programa cliente, del programa cliente regresa a través del sistema operativo y a través de la red de redes hacia la máquina servidor. Después de llegar a su máquina destino, los datos deben viajar a través del sistema operativo del servidor al programa de aplicación del servidor, y del programa de aplicación del servidor de regreso al sistema operativo del servidor, en un punto de entrada de pseudo terminal. Finalmente, el sistema operativo remoto entrega el carácter al programa de aplicación que el usuario está corriendo. Mientras tanto, la salida (incluyendo el eco de carácter remoto si es que la opción se ha seleccionado) viaja de regreso del servidor al cliente transfiriéndose por la misma trayectoria.

Los lectores que comprendan los sistemas operativos apreciarán que, para la implantación mostrada en la figura 23.1, cada pulso de tecla requiere que las computadoras se conmuten al contexto del proceso en varias ocasiones. En la mayor parte de los sistemas, se requiere adicionalmente la conmutación de contexto ya que el sistema operativo de la máquina del servidor debe transferir caracteres de regreso de la pseudo terminal a otro programa de aplicación (por ejemplo, un intérprete de comandos). Aunque el conmutador de contexto es caro, el esquema es práctico pues los usuarios no teclean a gran velocidad.

23.4 Adaptarse a la heterogeneidad

Para hacer que TELNET interopere entre tantos sistemas como sea posible, debe adaptar los detalles de las computadoras heterogéneas y los sistemas operativos. Por ejemplo, algunos sistemas requieren de líneas de texto que se terminen mediante el carácter de *control de retorno de carro* (*CR*) de ASCII. Para otros es necesario el carácter de *alimentación de línea* (*LF*) de ASCII. Incluso, algunos necesitan la secuencia de los dos caracteres CR-LF. Aunado a lo anterior, los sistemas más interactivos permiten que el usuario pulse una tecla para que interrumpe un programa que se está

[1] UNIX llama al punto de entrada del sistema un *pseudo tty* pues a los dispositivos orientados a caracteres se les llama *ttys*.

corriendo. Sin embargo, el pulso de teclado empleado para interrumpir un programa varía de sistema a sistema (por ejemplo, algunos sistemas utilizan Control-C, mientras otros se valen de ESCAPE).

Para adaptar la heterogeneidad, TELNET define cómo deben mandarse las secuencias de datos y comandos a través de Internet. La definición se conoce como *network virtual terminal* (*terminal virtual de red* o *NVT*). Como se ilustra en la figura 23.2, el software cliente traduce las pulsa-

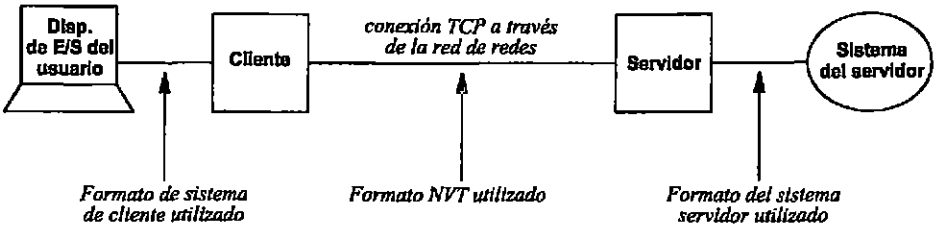


Figura 23.2 TELNET utiliza el formato Network Virtual Terminal (NVT, terminal virtual de red).

ciones de teclado y las secuencias de comandos que vienen de la terminal del usuario a formato NVT y las envía al servidor. El software del servidor traduce los datos y comandos que acaban de llegar de formato NVT al formato que el sistema remoto requiera. Para devolver los datos, el servidor remoto traduce del formato de una máquina remota a NVT y el cliente local traduce del formato NVT al formato de la máquina local.

La definición del formato NVT es bastante clara. Toda comunicación comprende un conjunto de octetos de 8 bits. Al arrancar, NVT utiliza la representación estándar de 7 bits de USASCII para los datos y reserva los octetos con el conjunto de bits de alto orden para las secuencias de comandos. El conjunto de caracteres USASCII incluye 95 caracteres que tienen gráficas "imprimibles" (por ejemplo, letras, dígitos y signos de puntuación) así como 33 códigos de "control". A to-

Código de control ASCII	Valor decimal	Significado asignado
NUL	0	No hay operación (sin efecto en la salida)
BEL	7	Sonido audible/señal visible (sin movimiento)
BS	8	Movimiento a la izquierda de un caracter
HT	9	Movimiento a la derecha al siguiente tab
LF	10	Movimiento hacia abajo (vertical) a la sig. línea
VT	11	Movimiento hacia abajo al sig. tab vertical
FF	12	Movimiento hacia arriba a la siguiente página
CR	13	Movimiento hacia la izquierda en la línea presente
otro control	-	Sin operación (sin efecto en la salida)

Figura 23.3 Interpretación NVT para TELNET de los caracteres de control USASCII. TELNET no especifica los sitios de las paradas de tab.

dos los caracteres imprimibles se les asigna el mismo significado que el conjunto de caracteres estándar de USASCII. El estándar NVT define las interpretaciones para los caracteres de control como se muestra en la figura 23.3.[2]

Además de la interpretación de caracteres de control de la figura 23.3, NVT define la terminación de línea estándar como una secuencia de dos caracteres: *CR-LF*. Cuando un usuario pulsa la tecla que corresponde a fin de línea en la terminal local (por ejemplo, *ENTER o RETORNO*), el cliente TELNET debe transformarla en *CR-LF* para su transmisión. El servidor TELNET traduce a *CR-LF* en la secuencia de caracteres apropiada de fin de línea para la máquina remota.

23.5 Transferencia de comandos que controlan el extremo remoto

Dijimos que la mayor parte de los sistemas proporciona un mecanismo que permite a los usuarios terminar con un programa que se está corriendo. Por lo general, el sistema operativo local enlaza dichos mecanismos a una tecla o secuencia de pulsaciones de teclas en particular. Por ejemplo, a menos que el usuario especifique otra cosa, muchos de los sistemas UNIX reservarán el carácter generado por *CONTROL-C* como la tecla de interrupción. Pulsar *CONTROL-C* hace que UNIX termine con la ejecución de un programa; el programa no recibe a *CONTROL-C* como entrada. El sistema puede reservar otros caracteres o secuencias de caracteres para otras funciones de control.

NVT de TELNET adapta las funciones de control mediante la definición de cómo se transmiten de cliente a servidor. Conceptualmente, pensamos en NVT como entrada de aceptación de un teclado que puede generar más de 128 caracteres. Suponemos que el teclado del usuario tiene teclas virtuales (imaginarias) que corresponden a las funciones que normalmente se utilizan para el procesamiento de control. Por ejemplo, NVT define una tecla de "interrupción" conceptual que pide la terminación de un programa. En la figura 23.4, se listan las funciones de control que NVT permite.

Señal		Significado
IP	(Interrupt Process)	Interrupción del proceso (termina de correrse el programa)
AO	(Abort Output)	Salida abortada (se descarta cualquier salida de búfer)
AYT	(Are You There)	Estás ahí (prueba si el servidor responde)
EC	(Erase Character)	Borra carácter (borra el carácter previo)
EL	(Erase Line)	Borra línea (borra toda la línea actual)
SYNCH	(Synchronize)	Sincroniza (despeja la trayectoria de datos hasta que el punto de datos TCP es urgente, pero interpreta comandos)
BRK	(Break)	Pausa (tecla de pausa o señal de atención)

Figura 23.4 Funciones de control que NVT de TELNET reconoce. Conceptualmente, el cliente recibe estas funciones de un usuario además de los datos normales, y los transmite al sistema del servidor donde se deben interpretar.

[2] La interpretación NVT de control de caracteres sigue a la interpretación usual ASCII.

En la práctica, la mayor parte de los teclados no posee teclas extra para los comandos. De hecho, los sistemas operativos individuales o los interpretadores de comandos tienen una gran variedad de maneras para generarlos. Ya mencionamos la técnica más común: construir un carácter ASCII individual para una función de control, de modo que, cuando el usuario pulsa esa tecla, el sistema operativo lleve a cabo las acciones apropiadas en lugar de aceptar al carácter como entrada. Los diseñadores de NVT eligieron mantener a los comandos separados del conjunto de caracteres ASCII normales por dos razones. En primer lugar, definir las funciones de control de manera separada significa que TELNET tiene una mayor flexibilidad. Puede transferir todas las secuencias de caracteres posibles en ASCII entre el cliente y el servidor así como también todas las funciones de control posibles. En segundo lugar, mediante la separación de señales de los datos normales, NVT permite que el cliente especifique las señales de manera no ambigua, nunca hay confusión acerca de si un carácter de entrada se deberá tratar como dato o como función de control.

Para transferir las funciones de control a través de la conexión TCP, TELNET las codifica mediante la *secuencia de escape*. Una secuencia de escape se vale de un octeto reservado para indicar que sigue a continuación un octeto de código de control. En TELNET, el octeto reservado que inicia una secuencia de escape se conoce como el octeto *interpret as command* (*interpretar como comando* o *IAC*). En la figura 23.5, se listan los comandos posibles y las codificaciones decimales utilizados para cada uno.

Comando	Codificación Decimal	Significado
IAC	255	Se interpreta al siguiente octeto como comando (cuando el octeto IAC aparece como dato, quien envía lo duplica y manda una secuencia de dos octetos IAC-IAC)
DON'T	254	Negación de petición para ejecutar una opción especificada
DO	253	Aprobación para permitir una opción especificada
WON'T	252	Rechazo de ejecución de una opción especificada
WILL	251	Autorización de realizar una opción especificada
SB	250	Inicio de subnegociación de opción
GA	249	Señal para "continuar" (go ahead)
EL	248	Señal de "borrado de línea" (erase line)
EC	247	Señal de "borrado de carácter" (erase character)
AYT	246	Señal de "estás ahí" (are you there)
AO	245	Señal de "aborto de salida" (abort output)
IP	244	Señal de "interrupción de proceso" (interrup process)
BRK	243	Señal de "pausa" (break)
DMARK	242	La porción de corriente de datos de un SYNCH (siempre acompañada de una notificación Urgente del TCP)
NOP	241	Sin operación
SE	240	Fin de la opción de subnegociación
EOR	239	Fin del registro

Figura 23.5 Comandos de TELNET y codificación para cada uno. Los códigos sólo tienen significado si están precedidos por un caracter *IAC*. Cuando se da un caracter *IAC* en los datos, éste se manda dos veces.

Como se muestra en la figura, las señales generadas por las teclas conceptuales en cada teclado NVT tienen un comando correspondiente. Por ejemplo, para pedir que el servidor interrumpa el programa que se está ejecutando, el cliente debe mandar la secuencia de dos octetos *IAC IP* (255 seguido de 244). Los comandos adicionales permiten que el cliente y el servidor negocien qué opciones utilizarán y la comunicación sincronizada.

23.6 Forzar al servidor a leer una función de control

El envío de funciones de control junto con datos normales no siempre es suficiente para garantizar los resultados deseados. A fin de ver por qué, consideremos la situación en la que un usuario podría enviar la función de control de *interrupción de proceso* al servidor. Normalmente, dicho control sólo se necesita cuando el programa que se ejecuta en una máquina remota se está conduciendo mal y el usuario quiere que el servidor deje de correr el programa. Por ejemplo, el programa podría estar ejecutando un ciclo sin fin sin leer la entrada o generando una salida. Por desgracia, si la aplicación en la localidad del servidor se detiene a leer la entrada, los buffers del sistema operativo en ocasiones se llenarán y el servidor será incapaz de escribir más datos en la pseudo terminal. Cuando esto sucede, el servidor debe dejar de leer datos de la conexión TCP que hacen que los búfers se llenen. En ocasiones, el TPC de la máquina servidor comenzará a anunciar un tamaño de ventana cero, previniendo que los datos fluyan a través de la conexión.

Si el usuario genera una función de interrupción de control, cuando los búfers están llenos, la función de control no llegará al servidor. Es decir que el cliente puede formar la secuencia de comandos *IAC IP* y escribirla en la conexión TCP, pero como el TCP ha dejado de enviar a la máquina del servidor, el servidor no leerá la secuencia de control. El punto es que:

> *TELNET no puede confiarse al flujo de datos convencional por sí sola para transportar secuencias de control entre cliente y servidor, pues una aplicación que se conduce mal necesita estar controlada ya que se podría bloquear de manera inadvertida el flujo de datos.*

Para resolver el problema, TELNET utiliza una señal *fuera de banda*. El TCP implementó la señalización fuera de banda con el mecanismo de *dato urgente*. Dondequiera que se coloque una función de control en la corriente de datos, TELNET también mandará un comando *SYNCH*. TELNET después anexará un octeto reservado, llamado *marca de datos* y hará que el TCP emita una señal hacia el servidor enviando un segmento con el conjunto de bits de *URGEN DATA* (*DATOS URGENTES*). Los segmentos que llevan los datos urgentes evitan el control de flujo y llegan de inmediato al servidor. En respuesta a una señal urgente, el servidor lee y descarta todos los datos hasta que encuentra la marca de datos. El servidor regresa a su procesamiento normal cuando encuentra la marca de datos.

23.7 Opciones de TELNET

En nuestra sencilla descripción de TELNET, se omite uno de los aspectos más complejo: las opciones. En TELNET, las opciones son negociables, lo que hace posible reconfigurar su conexión para el cliente y el servidor. Por ejemplo, dijimos que la corriente de datos solía transmitirse en datos de 7 bits y utilizaba octetos con el conjunto del octavo bit para transmitir la información de control como el comando de *interrupción de proceso*. Sin embargo, TELNET también ofrece una opción que permite que el cliente y el servidor transmitan datos de 8 bits (cuando se transmiten datos de 8 bits, el octeto reservado *IAC* debe aún duplicarse si aparece en los datos). El cliente y el servidor deben negociar, y ambos tienden a llegar al acuerdo de transmitir datos de 8 bits antes de que tales transmisiones sean posibles.

El rango de opciones de TELNET es amplio: algunos extienden las capacidades de manera significativa mientras que otros tratan con los detalles menores. Por ejemplo, el protocolo original fue diseñado para un ambiente half-duplex en el que era necesario indicar al otro extremo que "continuara" antes de que se pudieran mandar más datos. Una de las opciones controla si TELNET opera en modo half-duplex o full-duplex. Otra de las opciones permite que el servidor, en una máquina remota, determine el tipo de terminal del usuario. El tipo de terminal es importante para el software que genera las secuencias de posicionamiento del cursor (es decir, un editor de pantalla completa que se ejecuta en una máquina remota).

En la figura 23.6, se listan algunas de las opciones de TELNET que se implantan con mayor frecuencia.

Nombre	Código	RFC	Significado
Transmisión binaria	0	856	Se cambia la transmisión a modo binario de 8 bits
Eco	1	857	Se permite que uno de los lados haga eco para los datos que recibe
Supresión de GA	3	858	Se suprime (ya no se manda) la señal de continuar después de los datos
Estado	5	859	Petición del estado de la opción TELNET de una localidad remota
Marca de tiempo	6	860	Petición de que se inserte una marca de tiempo en la corriente de retorno para sincronizar dos extremos de una conexión
Tipo de terminal	24	884	Intercambio de información sobre la elaboración y modelo de una terminal que se está usando (permite que los programas se ajusten a la salida como las secuencias de posicionamiento del cursor para la terminal del usuario)
Fin de registro	25	885	Termina los datos mandados con código EOR
Modo de línea	34	1116	Utiliza la edición local y envía líneas completas en lugar de caracteres individuales

Figura 23.6 Opciones de TELNET que se usan con mayor frecuencia.

23.8 Negociación de opciones de TELNET

La manera en que TELNET negocia las opciones es muy interesante. Como en algunas ocasiones tiene sentido para el servidor iniciar una opción en particular, el protocolo está diseñado para permitir o dejar de hacer una petición. De este modo, se dice que el protocolo es *simétrico* con respecto al procesamiento de opciones. El extremo de recepción puede responder a una petición con una aceptación positiva o un rechazo. En la terminología de TELNET, la petición es *WILL X*, que significa *estarías de acuerdo en dejarme usar la opción X*; y la respuesta podría ser *DO X* o *DON'T X*, que significa *estoy de acuerdo en dejarte utilizar la opción X* o *no estoy de acuerdo en dejarte utilizar la opción X*. La simetría surge porque *DO X* pide que la parte receptora comience a usar la opción *X*, y *WILL X* o *WON'T X* significa *comenzaré a usar la opción X* o *no comenzaré a usar la opción X*.[3]

Otro concepto de negociación interesante surge por el hecho de que se requiere que ambos extremos corran una implantación de NVT no agrandada (es decir, una sin ninguna de las opciones activadas). Si una de las localidades trata de negociar una opción que la otra no comprende, la localidad que recibe la petición puede sencillamente declinarla. De este modo, es posible interoperar versiones más nuevas y sofisticadas de clientes y servidores TELNET (es decir, software que comprenda más opciones) con versiones más viejas y menos sofisticadas. Si el cliente y el servidor comprenden las nuevas opciones, pueden ser capaces de mejorar la interacción. Si no es así, se revertirán a un estilo menos eficiente pero trabajable.

Podemos resumir que:

> *TELNET utiliza un mecanismo de negociación de opción simétrica para permitir a los clientes y a los servidores reconfigurar los parámetros que controlan su interacción. Como el software TELNET comprende un protocolo NVT básico, los clientes y los servidores pueden interoperar aun cuando uno comprenda las opciones y el otro no.*

23.9 Rlogin (BSD de UNIX)

Los sistemas operativos que se derivan de BSD de UNIX incluyen un servicio de acceso remoto, llamado *rlogin*, que soporta a anfitriones confiables. Permite que los administradores del sistema elijan un conjunto de máquinas en las que se compartirán nombres de acceso y protecciones de acceso a archivos, y establezcan equivalencias entre los usuarios de los accesos. Los usuarios pueden controlar el acceso a sus cuentas autorizando un acceso remoto basado en un anfitrión remoto y un nombre de usuario remoto. De este modo, es posible para el usuario tener un nombre de acceso *X* en una máquina y *Y* en otra, e incluso ser capaz de hacer el acceso remoto de una de las máquinas a la otra sin teclear una clave de acceso en cada ocasión.

Tener una autorización automática hace que las características de acceso remoto sean útiles para los programas de propósitos generales así como también para la interacción humana. Una va-

[3] Para eliminar los ciclos potenciales que puedan surgir cuando cada uno de los dos lados piensa que el reconocimiento del otro es una petición, el protocolo especifica que no se habrá de dar ningún reconocimiento a una petición para una opción que ya está en uso.

riante del comando *rlogin* es *rsh*, el cual invoca un interpretador de comandos en la máquina UNIX remota y transmite los argumentos de la línea de comandos al interpretador de comandos, saltándose el paso de acceso por completo. El formato para una invocación de comando utilizando *rsh* es:

`rsh` *machine command*

De este modo, se teclea

```
rsh merlin ps
```

en cualquiera de las máquinas del Departamento de Ciencias Computacionales de la Universidad de Purdue, lo cual ejecuta el comando *ps* en la máquina *merlin*, con la entrada y la salida estándar de UNIX conectadas a través de la red al teclado y monitor del usuario. El usuario ve la salida como si estuviera trabajando en la máquina *merlin*. Debido a que el usuario puede hacer un arreglo para tener los comandos de invocación remota *rsh* sin necesidad de escribir una clave de acceso, se puede usar en programas lo mismo que también en teclado.

Ya que los protocolos como *rlogin* comprenden los ambientes computacionales locales y remotos, se comunican mejor que protocolos de acceso remoto con propósitos generales como TELNET. Por ejemplo, *rlogin* comprende las nociones de UNIX de *entrada estándar, salida estándar* y *error estándar*, asimismo se vale del TCP para conectarlos a la máquina remota. De este modo, es posible teclear:

```
rsh merlin ps > filename
```

y tener una salida del comando remoto redireccionada[4] al archivo *filename*. *Rlogin* también comprende las funciones de control de terminal como caracteres de control de flujo (que suelen ser Control-S y Control-Q). Se arregla para detener la salida inmediatamente sin esperar la demora requerida para mandarlos a través de la red hacia el anfitrión remoto. Por último, rlogin exporta parte del ambiente del usuario hacia la máquina remota, incluyendo información como el tipo de terminal del usuario (es decir, la variable *TERM*). Como resultado, una sesión de acceso parece comportarse casi de la misma manera que una sesión de acceso local.

23.10 Resumen

Buena parte de la rica funcionalidad asociada con el TCP/IP es resultado de una gran variedad de servicios de alto nivel proporcionados por programas de aplicación. Los protocolos de acceso remoto de alto nivel de estos programas emplean servicios básicos integrados: la entrega de datagramas no confiable y el transporte de flujo confiable. Los servicios por lo general siguen el modelo cliente servidor, en el cual los servidores operan en puertos de protocolo conocido de modo que los clientes saben cómo ponerse en contacto con ellos.

---

[4] El símbolo "mayor que" es de la sintaxis usual de UNIX y sirve para direccionar la salida de un comando dentro de un archivo.

Revisamos dos sistemas de acceso remoto: TELNET, la red de redes TCP/IP estándar, y rlogin, un protocolo muy popular utilizado con sistemas derivados de BSD de UNIX. TELNET proporciona un servicio básico. Este permite que el cliente transmita comandos tales como la *interrupción de proceso* así como también datos al servidor. También permite que el cliente y el servidor negocien muchas opciones. En contraste con TELNET, *rlogin* ofrece la posibilidad de que los administradores y usuarios del sistema tengan mayor flexibilidad al establecer la equivalencia de cuentas sobre varias máquinas, pero no se dispone de él con la amplitud con la que se dispone de TELNET.

PARA CONOCER MÁS

Varios protocolos de alto nivel han sido propuestos, pero sólo se usa unos cuantos de manera regular. Edge (1979) compara los protocolos de extremo a extremo con los de salto a salto. Saltzer, Reed y Clark (1984) arguyen que tienen la ejecución de protocolos de más alto nivel con el conocimiento de los de extremo a extremo y la detección de errores.

Postel (RFC 854) contiene la especificación del protocolo de acceso remoto de TELNET. Éste fue precedido por más de tres docenas de RFC en los que se analiza las opciones TELNET, sus debilidades, los experimentos y cambios propuestos, incluyendo a Postel (RFC 764) que contiene un estándar anterior. Postel y Reynolds (RFC 855) dan una especificación para las opciones y consideran la subnegociación. Se puede encontrar una larga lista de opciones en los RFC 856, 857, 858, 859, 860, 861, 884, 885, 1041, 1091, 1096, 1097, 1184, 1372, 1416 y 1572. El programa *tn3270* emplea un mecanismo parecido al de TELNET para proporcionar el acceso a computadoras IBM que corran el sistema operativo VM/CMS (RFC 1576, 1646 y 1647); Rekhter (RFC 1041) cubre la opción de TELNET que permite la comunicación con pantallas IBM 3270.

EJERCICIOS

23.1 Experimente con TELNET y con *rlogin*. *¿Cuáles son las diferencias más notables?*

23.2 A pesar del gran volumen de notas escritas acerca de TELNET, se puede argüir que el protocolo aún no está bien definido. Experimente con TELNET: utilícelo para llegar a la máquina *A* e invoque TELNET en *A* para llegar a una segunda máquina denominada como *B*. ¿La combinación de dos conexiones TELNET maneja apropiadamente los caracteres de *alimentación de línea* y de *control de retorno*?

23.3 ¿Qué es una llamada de procedimiento remoto?

23.4 Se dice que los sistemas operativos vienen y van mientras que los protocolos son para siempre. Compruebe este axioma investigando en la localidad en la que se encuentre su computadora a fin de ver si los sistemas operativos o los protocolos de comunicación han cambiado con más frecuencia.

23.5 Construya un software de cliente TELNET.

23.6 Utilice un cliente TELNET para conectar un teclado y monitor al puerto de protocolo TCP para *eco* o *cambio* en su sistema local y vea qué sucede.

23.7 Lea el estándar de TELNET y averigüe cómo funciona la operación SYNCH.

23.8 TELNET emplea el mecanismo de *dato urgente* del TCP para forzar al sistema operativo remoto a que responda a las funciones de control rápidamente. Lea el estándar para averiguar qué comandos del servidor remoto trabajan mientras se explora el flujo de entrada.

23.9 ¿Cómo puede la opción de negociación simétrica DO/DON'T- WIL/WON'T producir un ciclo sin fin de respuestas si la otra parte *siempre* reconoce una petición?

23.10 El archivo de texto para RFC 854 (la especificación del protocolo TELNET) contiene exactamente 854 líneas. ¿Piensa que hay una coincidencia cósmica en esto?

24

Aplicaciones: transferencia y acceso de archivos (FTP, TFTP, NFS)

24.1 Introducción

En este capítulo, continuaremos explorando los protocolos de aplicación. Examinaremos los protocolos de acceso y transferencia de archivos que son parte del conjunto de protocolos TCP/IP. Describiremos su diseño y mostraremos el ejemplo de una interfaz de usuario típica. Aprenderemos que el protocolo de transferencia de archivos usado con mayor frecuencia está basado en el TCP, tratado en el capítulo 13, y en TELNET, descrito en el capítulo anterior.

24.2 Acceso y transferencia de archivos

Muchos sistemas de red proporcionan computadoras con la capacidad de accesar archivos en máquinas remotas. Los diseñadores han explorado una gran variedad de enfoques al acceso remoto; cada enfoque optimiza un conjunto particular de metas. Por ejemplo, algunos diseños utilizan el acceso remoto de archivos a un costo general menor. En dichas arquitecturas, un *servidor de archivos* único y centralizado proporciona almacenamiento secundario para un conjunto de computadoras de bajo costo que no tienen disco local de almacenamiento. Por ejemplo, las máquinas sin discos pueden ser dispositivos portátiles y manejables utilizados para tareas como los inventarios. Dichas má-

quinas se comunican con un servidor de archivos a través de una red de trabajo inalámbrica de alta velocidad.

Algunos diseños se valen del almacenamiento remoto para archivar datos. En tales diseños, los usuarios tienen computadoras convencionales con capacidades de almacenamiento local y las manejan como es habitual. Las computadoras convencionales envían copias de archivos periódicamente (o copias de discos completos) a través de la red a un dispositivo de almacenamiento, en el que se guardan en caso de pérdida accidental.

Por último, en algunos se enfatiza la capacidad de compartir datos a través de diversos programas, usuarios o lugares. Por ejemplo, una organización podría elegir tener un catálogo en línea sencillo de productos, compartido por todos los grupos de la empresa.

24.3 Acceso compartido en línea

Compartir archivos se presenta en dos formas distintas: *acceso en línea* y *copiado de archivo completo*. El acceso compartido en línea significa que se permite a varios programas acceder de manera concurrente a un solo archivo. Los cambios que se realizan al archivo se efectúan inmediatamente y están disponibles para todos los programas que accedan al archivo. El copiado de archivo completo significa que, cada vez que un programa quiera acceder a un archivo, éste obtendrá una copia local. El copiado a menudo se utiliza para datos de sólo lectura, pero si el archivo debe modificarse, el programa hace los cambios en la copia local y transfiere de regreso el archivo modificado a la localidad original.

Muchos de los usuarios piensan que compatir de archivos en línea sólo se puede lograr mediante un sistema de base de datos que opere un servidor y que permita a los usuarios (clientes) ponerse en contacto con éste desde localidades remotas. Sin embargo, compartir archivos suele ser más sofisticado y fácil de usar. Por ejemplo, un sistema de archivos que proporciona acceso compartido en línea a los usuarios remotos no requiere necesariamente que el usuario invoque un programa especial de cliente, como lo haría un sistema de base de datos. De hecho, el sistema operativo proporciona acceso a archivos remotos compartidos de la misma manera que proporciona acceso a los archivos locales. Un usuario puede ejecutar cualquier programa de aplicación mediante un archivo remoto como entrada o salida. Decimos que el archivo remoto está *integrado* con los archivos locales y que todo el sistema de archivos proporciona *acceso transparente* a los archivos compartidos.

La ventaja del acceso transparente debería ser obvia: el acceso a archivos remotos ocurre sin cambios visibles en los programas de aplicación. Los usuarios pueden accesar tanto archivos locales como remotos, y además se les permite al mismo tiempo ejecutar cualquier tipo de cómputo en los datos compartidos. Las desventajas son menos evidentes. Los usuarios podrían sorprenderse por los resultados. Por ejemplo, consideremos un programa de aplicación que emplea archivos locales y remotos. Si la red de trabajo o la máquina remota está apagada, el programa de aplicación puede no funcionar aun cuando la máquina del usuario esté trabajando. Incluso si la máquina remota está en operación, podría estar sobrecargada o la red de trabajo podría estar congestionada, lo que ocasionaría que el programa de aplicación corriera lentamente o causara que los protocolos de comunicación reportaran condiciones de interrupción temporal que el usuario no espera. El programa no parece confiable así.

A pesar de sus desventajas, implantar un acceso a archivos integrado y transparente puede ser algo difícil. En un ambiente heterogéneo, los nombres disponibles de archivos en una computadora puede que no tengan posibilidad de transformase en el espacio de nombres de archivo de otro. De manera similar, un mecanismo de acceso remoto de archivos debe manejar nociones de propiedad, autorización y protección de acceso, lo cual no trasciende las fronteras del sistema computacional. Por último, como las representaciones de archivos y las operaciones permitidas varían de una máquina a otra, podría ser difícil o imposible implantar todas las operaciones en todos los archivos.

24.4 Compartir mediante la transferencia de archivos

La alternativa para el acceso en línea integrado y transparente es *la transferencia de archivos*. Accesar a datos remotos mediante un mecanismo de transferencia es un proceso de dos pasos: el usuario obtiene, primero, la copia local de un archivo y, después, trabaja en ella. La mayor parte de los mecanismos de transferencia opera fuera del sistema local de archivos (es decir que éstos no están integrados). El usuario debe invocar un programa de cliente de propósito especial para transferir archivos, cuando invoca al cliente el usuario especifica una computadora remota en la que residen los archivos deseados, pero tal vez se necesite una autorización para obtener acceso (es decir una cuenta o clave de acceso). El cliente se pone en contacto con un servidor en la máquina remota y pide una copia del archivo. Una vez que la transferencia se lleva a cabo, el usuario sale del cliente y utiliza los programas de aplicación en el sistema local para leer o modificar la copia local. Una ventaja del copiado de un archivo completo descansa en la eficiencia de operaciones —una vez que un programa ha obtenido una copia del archivo remoto, puede manipular la copia de manera eficiente. De este modo, muchas de las operaciones computacionales se corren más rápido con el copiado de un archivo completo que con el acceso remoto a archivos.

Al igual que con la opción de compartir en línea, la transferencia de archivos completos entre máquinas heterogéneas puede ser algo difícil. El cliente y el servidor deben estar de acuerdo en cuanto a autorizaciones, nociones de propiedad de archivos, protecciones de acceso y formatos de datos. Esto último es especialmente importante ya que las traducciones inversas son imposibles. Para ver por qué, consideremos el copiado entre dos máquinas, A y B, que utilizan representaciones diferentes para los números de punto flotante así como también representaciones diferentes para los archivos de texto. Como la mayoría de los programadores ha comprendido, podría ser imposible convertir el formato de punto flotante de una máquina al de otra sin perder precisión. Lo mismo puede pasar con los archivos de texto. Supongamos que el sistema A almacena archivos de texto con líneas de longitud variable y el sistema B da formato a las líneas de texto a una longitud fija. La transferencia de un archivo de A a B y de regreso puede incluir el formato de cada línea, lo cual hace que la copia final sea diferente de la original. Sin embargo, la remoción automática del formato de los finales de las líneas durante la transferencia de regreso a A también hará que la copia sea diferente del original para cualquier archivo que haya dado formato a algunas líneas.

Los detalles precisos acerca de las diferencias en la representación y las técnicas para manejarlos dependen de los sistemas computacionales comprendidos. Además, hemos visto que no todas las diferencias de representación pueden adaptarse, la información se puede perder cuando los datos se traducen de una representación a otra. Sin embargo, no es esencial aprender acerca de to-

das las posibles diferencias de representación, recordemos que el TCP/IP está diseñado para un ambiente heterogéneo, lo cual ayudará a explicar algunas de las características de los protocolos TCP/IP de transferencia de archivos.

24.5 FTP: el mayor protocolo TCP/IP para transferencia de archivos

La transferencia de archivos se da entre las aplicaciones TCP/IP utilizadas con mayor frecuencia, y que cuenta con mucho tráfico en red. Existían protocolos de transferencia de archivos estándar para ARPANET antes de que comenzara a funcionar el TCP/IP. Estas versiones tempranas de software de transferencia de archivos evolucionaron hasta llegar al estándar actual, conocido como *File Transfer Protocol* (*FTP, protocolo de transferencia de archivos*).

24.6 Características del FTP

Dado un protocolo de transporte confiable de extremo a extremo como el TCP, la transferencia de archivos podría parecer trivial. Sin embargo, como se ha señalado en secciones anteriores, los detalles de autorización, el nombre y la representación entre máquinas heterogéneas hace que el protocolo sea complejo. Además, el FTP ofrece muchas facilidades que van más allá de la función de transferencia misma.

- *Acceso interactivo.* Aunque el FTP está diseñado para usarse mediante programas, la mayor parte de las implantaciones proporciona una interfaz interactiva que permite a las personas interactuar fácilmente con los servidores remotos. Por ejemplo, un usuario puede pedir una lista de todos los archivos de un directorio en una máquina remota. También, el cliente suele responder a la entrada "help" (ayuda) mostrando la información del usuario acerca de los comandos posibles que se puedan invocar.
- *Especificación de formato (representación).* El FTP permite al cliente especificar el tipo y formato de datos almacenados. Por ejemplo, el usuario puede especificar si un archivo contiene enteros de texto o binarios, así como, si los archivos de texto utilizan los conjuntos de caracteres ASCII o EBCDIC.
- *Control de autenticación.* El FTP requiere que los clientes se autoricen a sí mismos con el envío de un nombre de conexión y una clave de acceso al servidor antes de pedir la transferencia de archivos. El servidor rechaza el acceso a clientes que no puedan abastecer una conexión o clave de acceso válida.

24.7 Modelo de proceso FTP

Como en otros servidores, la mayor parte de las implantaciones FTP de servidores permiten el acceso concurrente de varios clientes. Los clientes se valen del TCP para conectarse a un servidor. Como se describió en el capítulo 19, un proceso sencillo de servidor maestro espera las conexiones

y crea un proceso esclavo para manejar cada conexión. Sin embargo, a diferencia de casi todos los servidores, el proceso esclavo no ejecuta todos los cómputos necesarios. Por el contrario, el esclavo acepta y maneja la *conexión de control* de cliente, pero utiliza un proceso (o procesos) adicional para manejar una *conexión de transferencia de datos* separada. La conexión de control transporta comandos que indican al servidor qué archivo transferir. La conexión de transferencia de datos, que también usa el TCP como protocolo de transporte, transporta todas las transferencias de datos.

Por lo general, el cliente y el servidor crean un proceso separado para manejar la transferencia de datos. Si bien los detalles precisos acerca de la arquitectura de proceso dependen de los sistemas operativos utilizados, en la figura 24.1, se ilustra el concepto:

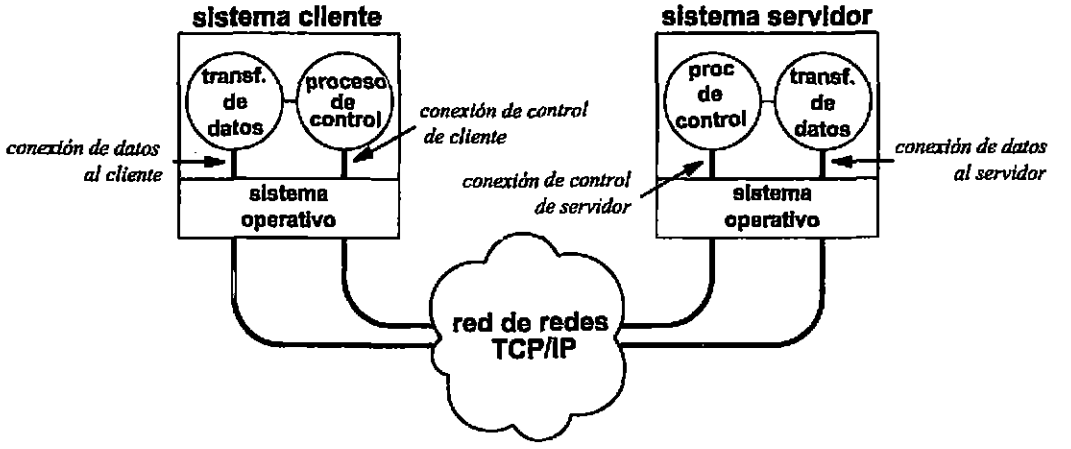


Figura 24.1 Un cliente y servidor FTP con una conexión de control TCP entre ambos y una conexión TCP separada entre sus procesos de transferencia de datos asociados.

Como se muestra en la figura, el proceso de control de cliente se conecta al proceso de control de servidor mediante una conexión TCP, mientras que los procesos de transferencia de datos asociados utilizan su propia conexión TCP. En general, los procesos de conexión y la conexión de control permanecen activos mientras el usuario continua con la sesión FTP. Sin embargo, el FTP establece una nueva conexión de transferencia de datos para cada transferencia de archivos. De hecho, muchas de las implantaciones crean un nuevo par de procesos de transferencia de datos, así como también una nueva conexión TCP desde donde quiera que el servidor necesite enviar información al cliente. La idea puede resumirse como sigue:

> *Las conexiones de transferencia de datos y los procesos de transferencia de datos que los emplean pueden crearse de manera dinámica cuando se necesitan, pero la conexión de control continúa a través de una sesión. Una vez que la conexión de control desaparece, la sesión se termina y el software en ambos extremos termina todos los procesos de transferencia de datos.*

Por supuesto, las implantaciones de cliente, que se ejecuten en una computadora sin el soporte de sistema operativo para diversos procesos, puede tener una estructura menos compleja. Tales implantaciones a menudo sacrifican la generalidad utilizando un solo programa de aplicación para ejecutar la transferencia de datos y las funciones de control. Sin embargo, el protocolo requiere incluso que tales clientes utilicen diversas conexiones TCP, una para el control y otras para la transferencia de datos.

24.8 Asignación de números de puerto TCP

Cuando un cliente establece una conexión inicial con un servidor, el cliente utiliza un número de puerto de protocolo aleatorio asignado localmente, pero se pone en contacto con el servidor en un puerto bien conocido (*21*). Como se señaló en el capítulo 19, un servidor que utiliza sólo un puerto de protocolo puede aceptar las conexiones de muchos clientes, puesto que el TCP se vale de ambos puntos extremos para identificar una conexión. Surge pues la pregunta: ¿cuándo crean los procesos de control una nueva conexión TCP para una transferencia de datos dada, qué números de puerto de protocolo emplean? Obviamente, no pueden usar el mismo par de números de puerto utilizados en la conexión de control. Por el contrario, el cliente obtiene un puerto no utilizado en su máquina y se vale del puerto para ponerse en contacto con el proceso de transferencia de datos en la máquina del servidor. Este proceso de transferencia de datos puede usar el puerto bien conocido (*20*), reservado para la transferencia de datos FTP. Para asegurar que un proceso de transferencia de datos en el servidor se conecte al proceso de transferencia de datos correcto en la máquina del cliente, por el lado del servidor, no debe aceptarse conexiones de un proceso arbitrario. De hecho, cuando se emite la petición abierta pasiva TCP, un servidor especifica el puerto que se usará en la máquina cliente así como también el puerto local.

Hallar un puerto remoto parecía difícil, pero ahora podemos ver por qué el protocolo utiliza dos conexiones: el proceso de control de cliente puede obtener un puerto local aleatorio que se use en la transferencia de archivos, comunicar el número de puerto al servidor a través de la conexión de control, esperar a que el servidor establezca un proceso de transferencia de datos aceptando una conexión de ese punto y, después, comenzar un proceso de transferencia en la máquina del cliente para realizar la conexión. En general:

> *Además de enviar comandos del usuario al servidor, el FTP utiliza la conexión de control para permitir los procesos de control cliente y servidor, y así, coordinar el uso de puertos de protocolo TCP asignados dinámicamente y la creación de procesos de transferencia de datos que utilicen tales puertos.*

¿Qué formato debería usar el FTP para enviar datos a través de la conexión de control? Aunque podían haber inventado una nueva especificación, los diseñadores del FTP no lo hicieron. De hecho, permitieron que el FTP utilizara el protocolo de terminal virtual de red de trabajo TELNET que se describe en el capítulo 23. A diferencia del protocolo completo de TELNET, el FTP no permite la negociación de opciones, emplea sólo la definición básica NVT. De este modo, la administración de una conexión de control FTP es mucho más sencilla que la administración de una cone-

xión estándar de TELNET. Sin importar las limitaciones, usar la definición de TELNET, en lugar de inventar una, ayuda a simplificar considerablemente al FTP.

24.9 El FTP desde el punto de vista del usuario

Los usuarios ven al FTP como un sistema interactivo. Una vez que se invoca, el cliente ejecuta repetidamente las siguientes operaciones: leer una línea de entrada, analizar la línea para extraer un comando y sus argumentos, así como ejecutar el comando con los argumentos especificados. Por ejemplo, para iniciar la versión del FTP disponible bajo BSD de UNIX, el usuario invoca el formato *ftp*:

```
% ftp
```

El programa local de cliente FTP comienza y despliega un indicador para el usuario. Después del indicador, el usuario puede desplegar comandos como *help*.

```
ftp> help
Los comandos pueden abreviarse.  Los comandos son:

!           cr          macdef      proxy       send
$           delete      mdelete     sendport    status
account     debug       mdir        put         struct
append      dir         mget        pwd         sunique
ascii       disconnect  mkdir       quit        tenex
bell        form        mls         quote       trace
binary      get         mode        recv        type
bye         glob        mput        remotehelp  user
case        hash        nmap        rename      verbose
cd          help        ntrans      reset       ?
cdup        lcd         open        rmdir
close       ls          prompt      runique
```

Para obtener más información acerca de un comando, el usuario teclea el *comando de ayuda (help command)* como en los siguientes ejemplos (la salida se muestra en el formato que produce *ftp*):

```
ftp> help ls
ls          lista el contenido del directorio remoto
ftp> help cdup
cdup        cambia el directorio de trabajo remoto por un directorio padre
ftp> help glob
glob        conmutación de metacaracteres de expansión de los nombres
             de archivo local
ftp> help bell
bell        hace un sonido cuando el comando se termina
```

Para ejecutar un comando, el usuario teclea el nombre del comando:

```
ftp> bell
Bell mode on. (modo de sonido activado)
```

24.10 Ejemplo de una sesión con FTP anónimo

Si bien las características de autorización de acceso del FTP lo hacen más seguro, el reforzamiento estricto prohibe a un cliente arbitrario el acceso a cualquier archivo hasta que se obtenga una conexión y una clave de acceso para la computadora en la que opera el servidor. Para proporcionar acceso a los archivos públicos, muchas de las localidades TCP/IP permiten el *FTP anónimo*. El acceso al FTP anónimo significa que el cliente no necesita una cuenta o clave de acceso, sino especificar un nombre de conexión *anónimo* y una clave de acceso de *invitado*. El servidor permite que el usuario anónimo se conecte pero restringe su acceso únicamente a los archivos públicos disponibles.[1]

En general los usuarios sólo ejecutan algunos comandos FTP para establecer una conexión y obtener un archivo; algunos usuarios ni siquiera han probado la mayor parte de los comandos. Por ejemplo, supongamos que alguien ha puesto una copia en línea de un texto en el archivo *tcpbook.tar* en el subdirectorio *pub/comer* en la máquina *arthur.cs.purdue.edu*. Un usuario conectado en otra localidad, por ejemplo *usera*, podría obtener una copia del archivo con sólo ejecutar lo siguiente.

```
% ftp ftp.cs.purdue.edu
Connected to arthur.cs.purdue.edu.
220 arthur.cs.purdue.edu. FTP server (Version 6.8) ready.
Name (ftp.cs.purdue.edu:usera): anonymous
331 Guest login ok, send e-mail address as password.
Password: guest
230 Guest login ok, access restrictions apply.
ftp> get pub/comer/tcpbook.tar bookfile
200 PORT command okay.
150 Opening ASCII mode data connection for tcpbook.tar (9895469 bytes).
226 Transfer complete.
9895469 bytes received in 22.76 seconds (4.3e+02 kbytes/s)
ftp> close
221 Goodbye.
ftp> quit
```

En este ejemplo, el usuario especifica la máquina *arthur.cs.purdue.edu* como un argumento para el comando FTP, de manera que el cliente abre automáticamente una conexión y pone la señal para una autorización. El usuario invoca al FTP anónimo especificando la conexión *anonymous*

[1] En muchos sistemas UNIX, el servidor restringe al FTP anónimo cambiando la raíz de sistema de archivo a un directorio pequeño y restringido (es decir, */usr/ftp*).

y la clave de acceso *guest*[2] (aunque nuestro ejemplo muestra la clave de acceso que el usuario teclea, el programa no la despliega en la pantalla del usuario).

Después de teclear una conexión y una clave de acceso, el usuario requiere la copia de un archivo y para ello utiliza el comando *get*. En el ejemplo, el comando *get* va seguido por dos argumentos que especifican el nombre del archivo remoto y el nombre de la copia local. El nombre del archivo remoto es *pub/comer/tcpbook.tar* y la copia local se colocará en *bookfile*. Una vez que se lleva a cabo la transferencia, el usuario teclea *close* para interrumpir al conexión con el servidor y, luego, *quit* para dejar al cliente.

Los mensajes de información se encuentran entremezclados con los comandos que teclea el usuario. Los mensajes FTP siempre comienzan con un número de 3 dígitos seguido de texto. La mayor parte viene del servidor; otra salida viene del cliente local. Por ejemplo, el mensaje que comienza con *220* viene del servidor y contiene el nombre de dominio de la máquina en la que se ejecuta el servidor. Las estadísticas que reportan el número de octetos recibidos y la proporción de transferencia que viene del cliente. En general:

> *Los mensajes de control y error entre el cliente y el servidor FTP comienzan con un número de tres dígitos seguido de texto. El software interpreta el número; el texto está dirigido a los usuarios.*

La sesión de ejemplo también ilustra una característica del FTP descrita al principio: la creación de nuevas conexiones TCP para transferencia de datos. Observe que el comando *PORT* está en la salida. El comando de cliente *PORT* reporta que un nuevo número de puerto TCP ha sido obtenido para usarse como conexión de datos. El cliente envía la información de puerto al servidor a través de la conexión de control; los procesos de transferencia de datos en ambos extremos se valen del nuevo número de puerto cuando se forma una conexión. Luego de que se completa la transferencia los procesos de transferencia de datos cierran la conexión.

24.11 TFTP

Aunque el FTP es el protocolo de transferencia de archivos más generalizado en el conjunto TC/IP, también es el más complejo y difícil de programar. Muchas aplicaciones no necesitan de la funcionalidad completa que ofrece el FTP, ni pueden afrontar la complejidad. Por ejemplo, el FTP requiere que tanto clientes como servidores manejen diversas conexiones TCP concurrentes, algo que puede ser difícil o imposible en computadoras personales que no tengan sistemas operativos sofisticados.

El conjunto TCP/IP contiene un segundo protocolo de transferencia de archivos que proporciona un servicio económico y poco sofisticado. Se conoce como *Trivial File Transfer Protocol* (*TFTP, protocolo trivial de transferencia de archivos*) *y* se diseñó para aplicaciones que no necesitan interacciones complejas entre cliente y servidor. El TFTP restringe las operaciones a transferencia de archivos sencilla y no proporciona autenticación. Como es más restrictivo el software TFTP resulta mucho más pequeño que el FTP.

---

[2] En la práctica, el servidor emite mensajes adicionales que piden al usuario que utilice una dirección e-mail en lugar de *guest*.

El tamaño reducido es importante para muchas aplicaciones. Por ejemplo, los fabricantes de dispositivos sin disco pueden codificar al TFTP en la memoria de sólo lectura (ROM) y usarlo para obtener una imagen de memoria inicial cuando se encienda la máquina. Al programa, en ROM, se le llama *arranque* del sistema. La ventaja de utilizar el TFTP es que permite al código de arranque emplear los mismos protocolos TCP subyacentes que el sistema operativo utiliza una vez que empieza la ejecución. De este modo es posible para una computadora arrancar desde un servidor en otra red física.

A diferencia del FTP, el TFTP no necesita un servicio de transporte de flujo confiable. Corre bajo UDP o cualquier otro sistema de entrega de paquetes no confiable y utiliza tiempos límites y retransmisión para asegurar que los datos lleguen. El lado que envía transmite un archivo de tamaño fijo (*512* octetos) bloquea y espera un acuse de recibo para cada bloque antes de enviar el siguiente. El receptor envía un acuse de recibo para cada bloque cuando le llega.

Las reglas del TFTP son sencillas. El primer paquete enviado requiere de una transferencia de archivo y establece la interacción entre cliente y servidor, el paquete especifica el nombre de archivo y si el archivo se leerá (transferido al cliente) o escrito (transferido al servidor). Los bloques del archivo están numerados en forma consecutiva comenzando con *1*. Cada paquete de datos contiene un encabezado que especifica el número de bloque que se transporta. Y cada acuse de recibo contiene el número de bloque del que se está recibiendo el acuse de recibo. Un bloque de menos de *512* octetos señala el final del archivo. Es posible enviar un mensaje de error en lugar de los datos o del acuse de recibo. Los errores terminan con la transferencia.

En la figura 24.2, se muestra el formato de los cinco tipos de paquetes TFTP. El paquete inicial debe utilizar códigos de operación *1* o *2*, especificando si se trata de una *petición de lectura* o de una *petición de escritura*. El paquete inicial contiene el nombre del archivo así como el modo de acceso que el cliente requiere (acceso de *lectura* o acceso de *escritura*).

cod.ope. 2 octetos	octetos n	1 octeto	n octetos	1 octeto
LEER PETIC. (1)	FILENAME	0	MODE	0

cod.ope. 2 octetos	octetos n	1 octeto	n octetos	1 octeto
ESCRI. PETIC. (2)	FILENAME	0	MODE	0

cod.ope. 2 octetos	2 octetos	hasta 512 octetos
DATOS (3)	# de BLOQUE	OCTETOS DE DATOS...

cod.ope. 2 octetos	2 octetos
ACUSE (3)	# de BLOQUE

cod.ope. 2 octetos	2 octetos	n octetos	1 octeto
ERROR (5)	CÓD. ERROR	MENSAJE ERROR	0

Figura 24.2 Los cinco tipos de mensaje TFTP. Los campos no se muestran a escala porque algunos son de longitud variable: un código de operación inicial de 2 octetos identifica el formato de mensaje.

Una vez que se ha hecho una petición de *escritura* o de *lectura*, el servidor utiliza la dirección IP y el número de puerto de protocolo UDP del cliente para identificar las operaciones subsiguientes. De este modo, ni los mensajes de *datos* (los mensajes que transportan bloques del archivo) ni los mensajes *ack* (los mensajes que dan el acuse de recibo de los bloques de datos) necesitan especificar el nombre de archivo. El tipo de mensaje final ilustrado en la figura 24.2 se utiliza para reportar errores. Los mensajes perdidos se pueden volver a transmitir después de un tiempo límite, pero la mayor parte de los demás errores simplemente causa que la interacción se acabe.

La retransmisión TFTP resulta inusual porque es simétrica. Cada lado implanta un tiempo límite y una retransmisión. Si el lado que envía llega a su tiempo límite, vuelve a transmitir el último bloque de datos. Si el lado responsable de los acuses de recibo llega a su tiempo límite vuelve a transmitir el último acuse de recibo. Tener a ambas partes participando en la transmisión ayuda a asegurar que no falle la transferencia después de que se haya perdido un paquete.

Aunque que la retransmisión simétrica garantiza potencia, puede conducir a retransmisiones excesivas. El problema conocido como *Falla del aprendiz de brujo* surge cuando un acuse de recibo para el paquete de datos k se demora pero no se pierde. El emisor vuelve a transmitir el paquete de datos para el cual el receptor emite un acuse de recibo. Ambos acuses de recibo llegan en algún momento y cada uno dispara una transmisión del paquete de datos $k + 1$. El receptor emite su acuse de recibo para ambas copias del paquete de datos $k + 1$, y los dos acuses de recibo causan que el emisor transmita el paquete de datos $k + 2$. La falla del aprendiz de brujo también puede iniciarse si la red de redes subyacente duplica los paquetes. Una vez que empieza el ciclo, continúa indefinidamente con cada paquete de datos que se transmite justo dos veces.

Aunque el TFTP contiene lo mínimo necesario para la transferencia, soporta diversos tipos de archivo. Un aspecto interesante del TFTP es el hecho de que puede estar integrado al correo electrónico.[3] Un cliente puede especificar al servidor que enviará un archivo al que se le considerará como correo con el campo *FILENAME* tomado como nombre para un buzón en el que el servidor deberá entregar el mensaje.

24.12 NFS

El *Network File System* (*Sistema de archivos de red o NFS*), que originalmente fue desarrollado por Sun Microsystem Incorporated, proporciona un acceso de archivos compartidos en línea que es transparente e integrado; muchas de las localidades TCP/IP utilizan el NFS para interconectar los archivos de sus computadoras. Desde la perspectiva del usuario, el NFS es casi invisible. Un usuario puede ejecutar un programa de aplicación arbitrario y valerse de archivos arbitrarios de entrada o salida. Los nombres de los archivos *per se* no muestran si son locales o remotos.

---

[3] En la práctica, las localizaciones no deberán usar TFTP para transportar correo. Refiérase al Capítulo 25 si requiere detalles sobre correo electrónico.

24.13 Implantación NFS

En la figura 24.3, se ilustra cómo es que el NFS está embebido en un sistema operativo. Cuando se ejecuta un programa de aplicación, se llama al sistema operativo para que *abra* un archivo o para que *almacene* y *recupere* datos en archivos. El mecanismo de acceso de archivos acepta la petición y la transmite de manera automática al software de sistema de archivo local o al cliente NFS, dependiendo de si el archivo está en el disco local o en una máquina remota. Cuando recibe una petición, el software de cliente utiliza el protocolo NFS para ponerse en contacto con el servidor apropiado en una máquina remota y ejecutar la operación requerida. Cuando contesta el servidor remoto, el software del cliente devuelve los resultados al programa de aplicación.

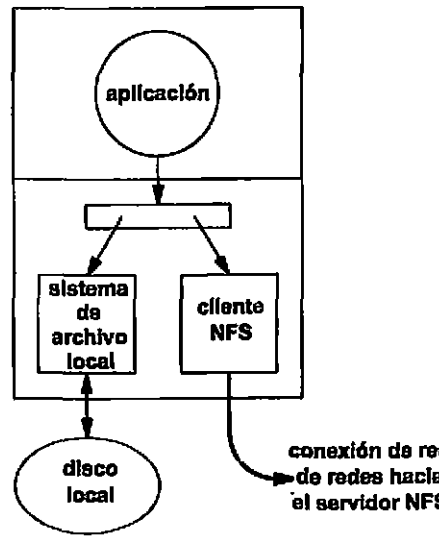


Figura 24.3 El código NFS en un sistema operativo. Cuando un programa de aplicación solicita una operación de archivo, el sistema operativo debe transportar la petición al sistema de archivos local o al software de cliente NFS.

24.14 Llamada de procedimiento remoto (RPC)

En lugar de definir el protocolo NFS de cero, los diseñadores prefirieron construir tres piezas independientes. El protocolo NFS es en sí, un mecanismo general de *llamada de procedimiento remoto (remote procedure call* o *RPC* por sus siglas en inglés) y una *representación de datos externa* (*eXternal data representation* o *XDR*) de propósito general. Su intención era separar los tres de manera que pudieran usar la RPC y la XDR en otro software, incluyendo programas de aplicación y otros protocolos.

Desde el punto de vista del programador, el NFS en sí mismo no proporciona nuevos procedimientos que un programa pueda llamar. En cambio, una vez que un administrador ha configurado al NFS, los programas acceden a los archivos remotos valiéndose exactamente de las mismas operaciones que se emplean para los archivos locales. Sin embargo, la RPC y la XDR proporcionan mecanismos que los programadores pueden utilizar para construir programas distribuidos. Por ejemplo, un programador puede dividir un programa de un lado como cliente y de otro como servidor y que utilicen la llamada RPC como principal mecanismo de comunicación. En el lado cliente, el programador designa como *remotos* algunos procedimientos, forzando así al compilador a incorporar el código RPC en dichos procedimientos. En el lado del servidor, el programador implanta los procedimientos deseados y utiliza otras características de RPC para declararlos como parte de un servidor. Cuando el programa de cliente que se está ejecutando llama a uno de los procedimientos remotos, la RPC recolecta automáticamente los valores para los argumentos, forma un mensaje, envía el mensaje al servidor remoto, espera una respuesta y almacena los valores devueltos en los argumentos designados. En esencia, la comunicación con el servidor remoto ocurre de manera automática como efecto colateral de una llamada de procedimiento remoto. El mecanismo RPC oculta todos los detalles de los protocolos, haciendo posible que los programadores que saben un poco acerca de los protocolos de comunicación subyacentes escriban programas distribuidos.

XDR, que es una herramienta relacionada, brinda a los programadores una manera de transmitir datos entre máquinas heterogéneas sin procedimientos de escritura que convertir entre las representaciones de datos de hardware. Por ejemplo, no todas las computadoras representan enteros binarios de 32 bits en el mismo formato. Algunas almacenan al octeto más significativo en la dirección de memoria más alta, mientras que otros almacenan al octeto menos significativo en la memoria más alta. De esta manera, si los programadores utilizan una red sólo para transportar los octetos de un entero de una máquina a otra sin arreglarlos, el valor del entero puede cambiar. XDR resuelve el problema definiendo una representación independiente de la máquina. En un extremo del canal de comunicación, un programa invoca procedimientos XDR para hacer conversiones de la representación de hardware local a la representación independiente de máquina. Una vez que los datos se han transmitido a la otra máquina, el programa receptor invoca las rutinas XDR para convertir de la representación independiente de máquina a la representación local de la máquina.

La ventaja principal de XDR es que automatiza buena parte de la tarea de conversión de datos. Los programadores no necesitan teclear manualmente las llamadas de procedimiento XDR. De hecho, se proporciona el compilador XDR con los enunciados de declaración del programa para el que deben transformarse los datos, y el compilador genera de manera automática un programa con las llamadas de biblioteca XDR necesarias.

24.15 Resumen

El acceso a datos de archivos remotos tiene dos formas: el copiado de archivos completos y el acceso compartido en línea. El protocolo de transferencia de archivos FTP es el principal protocolo de transferencia de archivos del conjunto FTP. El FTP utiliza el copiado de archivos completos y permite que los usuarios listen directorios en la máquina remota así como también transferir archivos en cualquier dirección. El protocolo trivial de transferencia de archivos TFTP ofrece una alternativa pequeña y sencilla al FTP para las aplicaciones que sólo necesitan de la transferencia de ar-

chivos. Como es lo suficientemente pequeña y cabe en ROM, el TFTP se puede usar en las máquinas de arranque sin disco.

El sistema de archivos de red NFS diseñado por Sun Microsystems Incorporated, proporciona acceso en línea a archivos compartidos. Utiliza el UDP para transporte de mensajes y la llamada de procedimiento remoto (RPC) de Sun, así como mecanismos de representación de datos externos (XDR). Debido a que RPC y XDR se definen de manera separada de NFS, los programadores pueden usarlas para construir aplicaciones distribuidas.

PARA CONOCER MÁS

Postel (RFC 595) contiene el estándar del protocolo FTP. Existen más de tres docenas de comentarios en RFC sobre el FTP, se proponen modificaciones o se definen nuevas versiones del protocolo. Entre ellas, Lottor (RFC 913) describe el protocolo sencillo de transferencia de archivos. DeSchon y Braden (RFC 1068) muestran cómo usar la transferencia tripartita del *FTP* para respaldar la transferencia de archivos. El Protocolo Trivial de Transferencia de Archivos descrito aquí viene de Sollins (RFC 783); Finlayson [RFC 906] describe su uso en los sistemas computacionales de arranque.

Sun Microsystems ha publicado tres RFC que definen el Sistema de Archivos de Red y los protocolos relacionados. El RFC 1094 contiene el estándar para NFS, el RFC 1057 define a RPC, y el RFC 1014 especifica a XDR. Se pueden encontrar más detalles sobre RPC y NFS en el volumen 3 de esta obra.

EJERCICIOS

24.1 ¿Por qué los protocolos de transporte de archivos deberían realizar una suma de verificación en los archivos de datos que reciben, a pesar de que utilizan un protocolo de transferencia de flujo de extremo a extremo confiable como el TCP?

24.2 Averigüe si el FTP realiza una revisión de suma de los archivos que transfiere.

24.3 ¿Qué pasa en el FTP si la conexión TCP que se emplea para la transferencia de datos se interrumpe pero la conexión de control no?

24.4 ¿Cuál es la ventaja principal de utilizar conexiones separadas del TCP para el control y la transferencia de datos? (*Pista: piense en condiciones anormales.*)

24.5 Señale un método que utilice el TFTP para arrancar una máquina sin disco. Sea cuidadoso. ¿Exactamente qué direcciones IP lo utilizan a cada paso?

24.6 Implante un cliente TFTP.

24.7 Experimente con el FTP o con un protocolo equivalente para ver qué tan rápido puede transferir un archivo entre dos sistemas razonablemente grandes a través de una red de área local. Trate de experimentar cuando la red esté ocupada y cuando se encuentre "ociosa". Explique el resultado.

24.8 ¿Pruebe el FTP desde una máquina a usted mismo y después de una máquina a otra máquina en la misma red de área local. ¿Le sorprendió la proporción de datos transferidos?

24.9 Compare las proporciones de transferencia para FTP y NFS en una red de área local. ¿Puede explicar la diferencia?

24.10 Examine la definición RPC. ¿Maneja pérdida de datagramas, duplicación, demora o corrupción?

24.11 ¿Bajo qué circunstancias es ineficiente el esquema XDR

24.12 Considere la traducción de los números de punto flotante de una forma interna a una forma externa y de vuelta a una forma interna. ¿Qué son los trueques en la elección de tamaños de exponente y mantisa en la forma externa?

25

Aplicaciones: correo electrónico (822, SMTP, MIME)

25.1 Introducción

En este capítulo, continuamos con nuestra exploración del enlace de redes considerando el servicio de correo electrónico y los protocolos que lo soportan. También, se describe cómo está organizado un sistema de correo, se explica la expansión de alías y se muestra cómo utiliza el software de sistema de correo el paradigma cliente-servidor para transferir cada mensaje.

25.2 Correo electrónico

Varios de los primeros encuentros de los usuarios con las redes de computadora se dan cuando envían o reciben correo electrónico (e-mail) desde o hacia una localidad remota. E-mail es el servicio de aplicación utilizado más ampliamente. En realidad, muchos de los usuarios de computadoras acceden a las redes sólo a través del correo electrónico.

E-mail es popular porque ofrece un método rápido y conveniente de transferencia de información. E-mail puede adaptarse al envío de pequeñas notas o grandes y voluminosos documentos mediante un mecanismo sencillo. No debería sorprenderle enterarse de que hay más usuarios que envían archivos por el correo electrónico que por los protocolos de transferencia de archivos.

La entrega de correo es un nuevo concepto porque difiere fundamentalmente de otros usos de las redes que hayamos discutido. En todos nuestros ejemplos, los protocolos de red envían paquetes directamente a sus destinos, utilizando límite de tiempo y retransmisión para los segmentos individuales si no se devuelve un acuse de recibo. Sin embargo, en el caso del correo electrónico, el sistema

debe proporcionar los medios cuando la máquina remota o las conexiones de la red han fallado. El emisor no desea esperar a que la máquina remota esté disponible para continuar trabajado, ni el usuario quiere que se aborte la transmisión sólo porque las comunicaciones con la máquina remota no están disponibles temporalmente.

Para manejar las entregas con retraso, el sistema de correo utiliza una técnica conocida como *spooling*.[1] Cuando el usuario envía un mensaje de correo, el sistema coloca una copia en su área de almacenamiento privado (*spool*)[2] junto con la identificación del emisor, recipiente, máquina de destino y hora de depósito. El sistema, entonces, inicia la transferencia hacia la máquina remota como una actividad subordinada o secundaria, permitiendo al emisor que continúe con otras actividades computacionales. La figura 25.1 ilustra la idea.

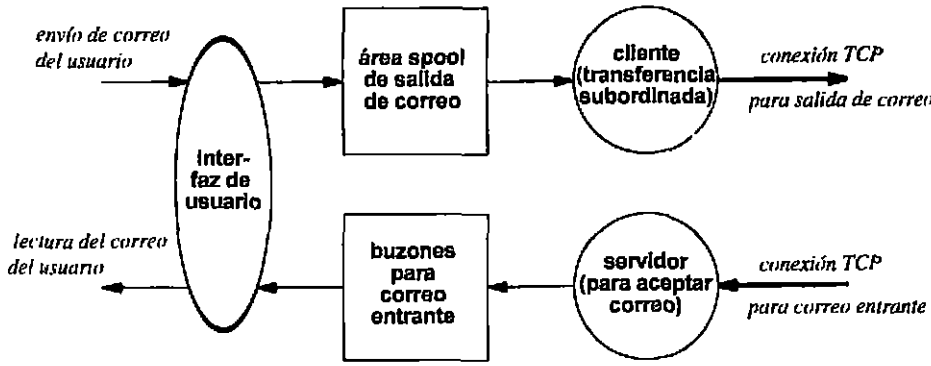


Figura 25.1 Componentes conceptuales de un sistema de correo electrónico. El usuario invoca una interfaz de usuario para depositar o recuperar correo; Todas las transferencias se dan en un proceso subordinado.

El proceso subordinado de transferencia de correo se establece como un cliente. El proceso primero utiliza el sistema de nombres de dominio para transformar el nombre de la máquina de destino en una dirección IP y luego trata de establecer una conexión TCP hacia el servidor de correo en la máquina destino. Si tiene éxito, el proceso de transferencia envía una copia del mensaje al servidor remoto, el cual almacena la copia en el área de proceso spool del sistema remoto. Una vez que el cliente y el servidor acuerdan que la copia ha sido aceptada y almacenada, el cliente desecha la copia local. Si no se puede establecer una conexión TCP o si la conexión falla, el proceso de transferencia registra la hora en que se intentó la entrega y termina el proceso. El proceso de transferencia subordinado realiza de manera periódica un barrido a través del área spool, por lo general, una vez cada 30 minutos, en busca de correo no enviado. Cada vez que se encuentra un mensaje o que un usuario deposita correo pendiente, el proceso subordinado intenta entregarlo de nuevo. Si encuentra que el mensaje de correo no se puede entregar después de un tiempo prolongado (por ejemplo, tres días), el software de correo devuelve el mensaje al emisor.

[1] Se trata de una técnica de procesamiento simultáneo de tareas periféricas. (N. del T.)

[2] El área spool de correo se conoce a veces como *cola de correo*, aun cuando el término es técnicamente impreciso.

25.3 Nombres y alias de los buzones de correo

Hay tres ideas importantes que subyacen en nuestra descripción general de la entrega de correo. En primer lugar, los usuarios especifican recipientes, proporcionando pares de cadenas que identifican el *nombre de la máquina destino para el correo* y una *dirección de buzón* en tal máquina. En segundo lugar, los nombres utilizados en estas especificaciones son independientes de otros nombres asignados a las máquinas. Usualmente, la dirección de un buzón es la misma que la identificación del usuario ante el sistema, y el nombre de la máquina destino es el mismo que el nombre de dominio de la máquina, aunque esto no necesariamente es así. Es posible asignar una dirección de buzón a la posición de un empleo (por ejemplo, el identificador de buzón *jefe de departamento* puede referirse a cualquier presidente actual del departamento). También, debido a que el sistema de nombre de dominio incluye un tipo de solicitud separada para destinos de correo, es posible separar nombres de destino de correo de los nombres de destino normalmente asignados a las máquinas. Así, el correo enviado al usuario *machine.com* puede ir hacia una máquina diferente a una conexión telnet con el mismo nombre de máquina. En tercer lugar, nuestro sencillo diagrama falla en nuestra consideración del *correo en proceso* y del *correo en envío*, los cuales incluyen el correo enviado de un usuario a otro en la misma máquina y el correo que llega a una máquina pero que deberá ser enviado hacia otra máquina.

25.4 Expansión de alias y direccionamiento de correspondencia

La mayor parte de los sistemas proporciona un software de *envío de correo* que incluye un mecanismo de *expansión de alias de correo*. Un emisor de correo permite a una localidad de zona transformar identificadores por medio de una dirección de correo para conjuntos de una o más direcciones de correo. Por lo regular, luego de que el usuario compone un mensaje y nombra un recipiente, el programa de interfaz de correo consulta los alias locales para remplazar el recipiente con la versión transformada antes de pasar el mensaje al sistema de entrega. Los recipientes para los que no se ha especificado transformaciones se mantienen sin cambios. De la misma forma, el sistema de correo subyacente utiliza los alias de correo para transformar direcciones entrantes de recipientes.

Los alias incrementan sustancialmente la funcionalidad del sistema de correo. En términos matemáticos, las transformaciones de alias pueden ser de muchos a uno o de uno a muchos. Por ejemplo, el sistema de alias permite a un solo usuario tener varios identificadores de correo, incluyendo sobrenombres y posiciones, transformando un conjunto de identificadores hacia una sola persona. El sistema también permite que una localidad asocie grupos de recipientes con un solo identificador. El uso de alias que transforman un identificador en una lista de identificadores hace posible establecer un *distribuidor* que acepte un mensaje entrante y lo envíe a un amplio conjunto de recipientes. El conjunto de recipientes asociados con un identificador se conoce como *lista de correo electrónico*. No todos los recipientes en una lista necesitan estar en el mismo local. Aun cuando es poco común, es posible tener una lista en una localidad, Q, sin ninguno de los recipientes de la lista localizados en Q. Extender un alias de correo en un conjunto amplio de recipientes es una técnica popular y muy utilizada. La figura 25.2 ilustra los componentes de un sistema de correo que soporta alias de correo y listas de expansión.

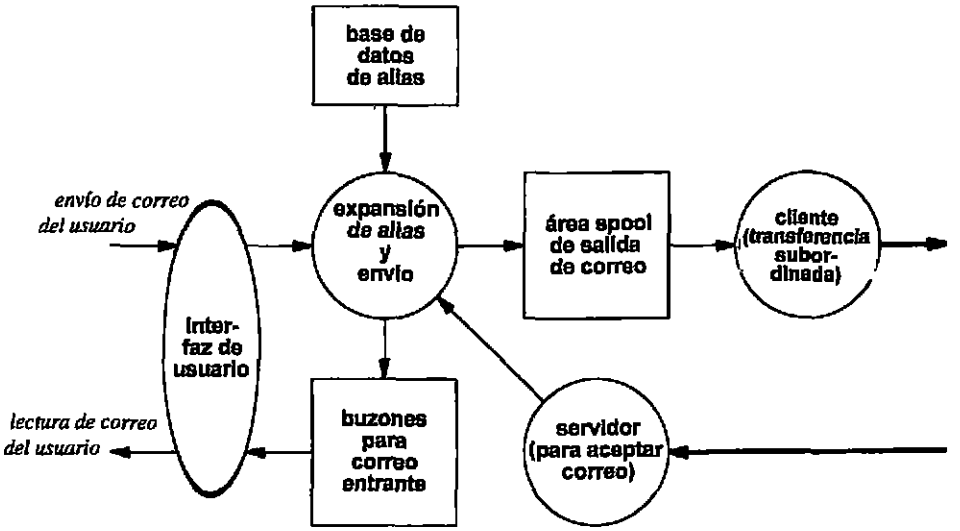


Figura 25.2 Extensión del sistema de correo electrónico de la figura 25.1, que soporta alias de correo y envíos. Tanto la entrada como la salida de correo pasa a través del mecanismo de expansión de alias.

Como se muestra en la figura 25.2, el correo que entra y sale pasa a través del emisor que expande los alias. De esta manera, si la base de datos de los alias especifica que la dirección de correo *x* se transforma a fin de ser remplazada por *y*, la expansión de alias reescribirá la dirección de destino *x*, cambiándola por *y*. El programa de expansión de alias determina entonces si *y* especifica una dirección local o remota para saber si debe colocar el mensaje en la cola de correo entrante o en la cola de correo que sale.

La expansión de alias de correo puede ser peligrosa. Supongamos que dos localidades establecen alias conflictivos. Por ejemplo, supongamos que la localidad *A* transforma la dirección de correo *x* en la dirección de correo *y* en la localidad *B*, mientras que la localidad *B* transforma la dirección de correo *y* en la dirección *x* en la localidad *A*. Un mensaje de correo enviado hacia la dirección *x* en la localidad *A* puede rebotar indefinidamente entre las dos localidades.[3] De la misma forma, si el administrador en la localidad *A* transforma accidentalmente el nombre de identificación en el sistema de un usuario en dicha localidad, en una dirección de otra localidad, el usuario será incapaz de recibir correo. El correo puede ir hacia el otro usuario o, si el alias especifica una dirección ilegal, el emisor recibirá mensajes de error.

25.5 Relación entre el enlace de redes y el correo electrónico

Muchos sistemas de computadoras comerciales pueden enviar correo electrónico desde localidades que no están conectadas a Internet. ¿Qué tanto difieren estos sistemas de los sistemas de correo des-

[3] En la práctica, la mayor parte de los distribuidores de correo terminan los mensajes luego de que el número de intercambios alcanza un umbral predeterminado.

critos aquí? Existen dos diferencias cruciales. En primer lugar, una red de redes TCP/IP hace posible el servicio de entrega universal. En segundo lugar, el sistema de correo electrónico construido en el TCP/IP es inherentemente más confiable que los construidos a partir de redes arbitrarias. La primera idea es fácil de entender. El TCP/IP hace posible la entrega de correo universal pues proporciona interconexión universal entre máquinas. En esencia, todas las máquinas conectadas a una red de redes se comportan como si estuvieran conectadas a una sola red independiente de los equipos específicos de ciertos vendedores. Con los servicios de red básicos, construir un protocolo de intercambio de correo estándar se hace más fácil.

El segundo argumento, es decir, que el uso del TCP/IP hace la entrega de correspondencia más confiable que otros mecanismos, necesita de una mayor explicación. La idea clave aquí es que el TCP proporciona conectividad de extremo a extremo. Lo que significa que el software de correo en la máquina emisora actúa como un cliente, contactando a un servidor en el destino final. Sólo después de que el cliente logra transferir un mensaje de correo al servidor, se elimina el mensaje de la máquina local. Así pues, la entrega directa y de extremo a extremo refuerza el siguiente principio:

> *Los sistemas de correo que utilizan la entrega de extremo a extremo pueden garantizar que cada mensaje de correo se mantenga en la máquina del emisor hasta ser copiado con éxito en la máquina del receptor.*

Con cada sistema, el emisor puede determinar siempre el estado exacto de un mensaje, verificando el área spool de correo local.

La alternativa para una entrega de correo electrónico emplea *mail gateways* (*compuertas de correo*),[4] a veces llamadas *mail bridges, mail relay* o *intermediate mail stops* para transferir mensajes. En cada sistema, la máquina del emisor no establece contacto directamente con la máquina del receptor, sino que envía el correo a través de una o más maquinas intermedias que completan el envío.

La principal desventaja de utilizar compuertas de correo es que reducen la confiabilidad. Una vez que la máquina del emisor transfiere un mensaje a la primera máquina intermedia, se descarta la copia local. Así, mientras el mensaje está en tránsito, ni el receptor ni el emisor tienen una copia. Si se dan fallas en las máquinas intermedias esto puede provocar una pérdida del mensaje sin que se informe ni al emisor ni al receptor. También, se puede dar una pérdida de mensajes si las compuertas de correo rutean el correo de manera incorrecta. Otra desventaja de las compuertas de correo es que introducen un retardo. Una compuerta de correo puede manejar mensajes por minutos, horas o incluso días, si no pueden enviar el correo a la siguiente máquina. Ni el emisor ni el receptor pueden determinar en qué lugar se ha retrasado un mensaje, por qué no ha llegado o con qué retardo llegará. El punto importante es que el emisor y el receptor dependen de máquinas sobre las que pueden no tener ningún control.

Si las compuertas de correo son menos confiables que la entrega de extremo a extremo, ¿porqué se usan? La mayor ventaja de las compuertas de correo es su interoperabilidad. Las compuertas de correo proporcionan conexiones entre el estándar TCP, el sistema de correo TCP/IP estándar y otros sistemas de correo, así como entre redes de redes TCP/IP y redes que no soportan los protocolos de Internet. Supongamos, por ejemplo, que la compañía *X* tiene una red interna extensa y que los empleados utilizan el correo electrónico, pero que el software de la red no soporta el TCP/IP. Aun cuando podría ser imposible hacer que la red de la compañía tuviera que conectarse a Internet, podría

---

[4] El lector no debe confundir el término *compuerta de correo* con el término *compuerta IP*, analizado antes.

ser fácil colocar una compuerta de correo entre la red privada de la compañía e Internet así como diseñar software que aceptara mensajes de correo desde la red local y los enviara hacia Internet.

Aun cuando la idea de las compuertas de correo podría parecer algo difícil, el correo electrónico la ha convertido en una herramienta importante de la que dependen los usuarios que no tienen acceso a Internet. Así, aunque el servicio de compuertas no es confiable o conveniente como la entrega de extremo a extremo, puede ser útil.

25.6 Estándares TCP/IP para el servicio de correo electrónico

Recordemos que el objetivo del protocolo TCP/IP es esforzarse por proporcionar interoperatividad a través de un amplio rango de sistemas de computadoras y redes. Para extender la interoperatividad del correo electrónico, el TCP/IP divide sus estándares de correo en dos grupos. Un estándar especifica el formato para los mensajes de correo.[5] El otro especifica los detalles del intercambio de correo electrónico entre dos computadoras. Mantener los dos estándares separados para el correo electrónico hace posible construir compuertas de correo que conecten redes de redes TCP/IP con algunos sistemas de entrega de correo de otros vendedores, siempre y cuando utilicen el mismo formato de mensajes para ambos.

Cualquiera que haya utilizado el correo electrónico sabe que cada memorándum está dividido en dos partes: un encabezado y un cuerpo de mensaje, separados por una línea en blanco. El estándar TCP/IP para los mensajes de correo específica el formato exacto de los encabezados de correo así como el significado de interpretación de cada campo del encabezado; la definición del formato del cuerpo se deja al emisor. En particular, el estándar específica que los encabezados contienen texto que es posible leer, dividido en líneas que consisten en palabras clave, seguidas por puntos y por un valor. Algunas palabra clave son necesarias, otras son opcionales y el resto no tiene interpretación. Por ejemplo, el encabezado debe contener una línea que especifique el destino. La línea comienza con *To:* y el resto contiene la dirección del correo electrónico del recipiente proyectado. Una línea que comienza con *From:* contiene la dirección de correo electrónico del emisor. Opcionalmente, el emisor puede especificar una dirección a la que se pueden enviar réplicas (esto es, para permitir al emisor especificar que las réplicas deben enviarse a una dirección diferente al buzón del emisor). Si está presente, una línea que comienza con *Reply-to:* especifica la dirección para las réplicas. Si esta línea no existe, el recipiente usará la información en la línea *From:* como la dirección de retorno.

El formato de los mensajes de correo ha sido seleccionado para facilitar el proceso y realizar el transporte a través de máquinas heterogéneas. Mantener el formato del encabezado de correo sin cambios permite utilizarlo dentro de un amplio rango de sistemas. La restricción de los mensajes al formato de sólo texto evita los problemas de seleccionar una representación binaria estándar y traducir entre la representación estándar y la representación de la máquina local.

[5] Los expertos en el sistema de correo frecuentemente se refieren al formato de mensaje de correo como "822" ya que el RFC 822 contiene el estándar (RFC 733 es un estándar anterior que ya no se usa).

25.7 Direcciones de correo electrónico

Un usuario familiarizado con el correo electrónico sabe que los formatos de las direcciones de correo varían entre sistemas de e-mail. Así, puede ser difícil determinar una dirección de correo electrónico correcta o, incluso, entender la intención del emisor. Dentro de la red global de Internet, las direcciones tienen una forma simple y fácil de recordar:

local-part @domain-name

Donde *domain-name* es el nombre de dominio de un destino de correo[6] al que el correo debe ser entregado y *local-part* es la dirección de un buzón en la máquina. Por ejemplo, dentro de Internet, la dirección de correo electrónico del autor de esta obra es:

comer@purdue . edu

Sin embargo, las compuertas de correo vuelven las direcciones complejas. Cualquiera que esté fuera de Internet debe direccionar el correo hacia la compuerta de correo más cercana o tener software que lo haga de manera automática. Por ejemplo, cuando CSNET operaba una compuerta de correo que conectaba redes exteriores con Internet, algunos usuarios con acceso a la compuerta podían usar la siguiente dirección para ponerse en contacto con el autor:

comer % purdue . edu@relay . cs . net

Una vez que el correo alcanzaba la máquina *relay.cs.net*, el software de compuerta de correo extraía la *local-part*, cambiando el signo de (%) por el signo (@) y se utilizaba el resultado como una dirección de destino para enviar la correspondencia.

La razón por la que las direcciones se hacen complejas cuando se incluyen localidades que no son de Internet, es que la función de transformación de las direcciones de correo electrónico se hace localmente para cada máquina. Así, algunas compuertas de correo requieren que la parte local contenga direcciones de la forma:

user % domain-name

mientras que otras requieren:

user:domain-name

e incluso otras necesitan formas completamente diferentes. Algo muy importante, los sistemas de correo electrónico por lo general no acuerdan convenciones sobre la precedencia o las cuotas, lo que hace imposible para un usuario garantizar la forma en que será tratado su direccionamiento. Por ejemplo, consideremos la dirección de correo electrónico:

comer%purdue . edu @relay . cs . net

---

[6] Técnicamente el nombre de dominio especifica un *distribuidor de correo*, no un nombre de máquina.

mencionada inicialmente. Una localidad que utiliza el estándar TCP/IP para enviar correo debe interpretar el significado de la dirección, "envía el mensaje al distribuidor de correo *relay . cs . net* y deja que el distribuidor de correo decida como interpretar *comer%purdue.edu*" (la *local-part*). En esencia, la localidad actúa como si la dirección estuviera entre paréntesis:

(comer%purdue . edu)@(relay . cs . net)

En las localidades que utilizan % para separar nombres de su área de la máquina destino, la misma dirección puede significar: "envía el correo al usuario *comer* en la localidad que se proporciona en el resto de la dirección": Esto es, cada localidad actúa como si la dirección estuviera entre paréntesis:

(comer)%(purdue . edu @ relay . cs . net)

Podemos resumir el problema de la siguiente forma:

> *Como cada compuerta de correo determina los detalles exactos de cómo interpretar y transformar las direcciones de correo electrónico, no hay un estándar para los direccionamientos que cruzan las fronteras de las compuertas de correo hacia redes que están fuera de Internet.*

25.8 Pseudo direcciones de dominio

Para ayudar a resolver el problema de los diversos sistemas de correo, cada uno con su propio formato de dirección, una localidad puede utilizar los nombres de dominio-tipo para todos los direccionamientos de e-mail, aun cuando la localidad no utilice el sistema de nombres de dominio. Por ejemplo, una localidad que emplee el UUCP puede implantar un seudo-dominio *uucp*, que permita a los usuarios especificar direcciones de correo de la forma:

dirección tipo-uucp @ uucp

o en la forma relacionada:

usuario @ localidad-uucp.uucp

El software para envío de correo local reconoce las direcciones especiales y las transforma en la sintaxis de dirección requerida por el software de red UUCP. Desde la perspectiva del usuario, la ventaja es clara: todas las direcciones electrónicas tienen el mismo formato general, independientemente de la red de comunicaciones subyacente utilizada para llegar hasta el recipiente. Por supuesto, cada direccionamiento sólo opera en donde los transportadores de correo locales han sido instruidos para transformar las direcciones en las formas apropiadas y sólo cuando el mecanismo de transporte apropiado está disponible. Además, aunque el pseudodominio de las direcciones de correo tiene la misma forma que los nombres de dominio, sólo pueden utilizarse con el correo electrónico

no se pueden encontrar direcciones IP o direcciones de intercambio de correo para éstas, utilizando el sistema de nombres de dominio.

25.9 Protocolo de transferencia de correo simple (SMTP)

Además del formato de los mensajes, el conjunto de protocolos TCP/IP especifica un estándar para el intercambio de correo entre máquinas. Es decir, el estándar especifica el formato exacto de los mensajes a un cliente en una máquina que lo utiliza para transferir correo hacia el servidor en otra. El protocolo de transferencia estándar se conoce como *SMTP*, *Simple Mail Transfer Protocol* (*Protocolo de transferencia de correo simple*). Como se podrá adivinar, el SMTP es más sencillo que el *Mail Transfer Protocol*, *MTP* original. El protocolo SMTP se enfoca específicamente en cómo transfiere el sistema de entrega de correo subyacente los mensajes a través de un enlace de una máquina a otra. No especifica de qué manera acepta el sistema de correo los mensajes de correo de un usuario o cómo presenta al usuario la interfaz de usuario el correo entrante. El SMTP tampoco específica en qué forma se almacena el correo o con qué frecuencia el sistema de correo trata de enviar mensajes.

El SMTP es sorprendentemente sencillo. La comunicación entre un cliente y un servidor consiste en texto ASCII que es posible leer. Aun cuando el SMTP define rígidamente el formato de los comandos, los usuarios pueden leer fácilmente una transcripción de interacciones entre un cliente y un servidor. Inicialmente, el cliente establece una conexión de flujo confiable con el servidor y espera que el servidor envíe un mensaje *220 READY FOR MAIL*. (Si el servidor está sobrecargado, deberá retardar el envío del mensaje *220* temporalmente.) Al recibir el mensaje *220*, el cliente envía un comando *HELO*.[7] El extremo de una línea marca el fin de un comando. El servidor responde identificándose. Una vez que la comunicación se ha establecido, el emisor puede transmitir uno o más mensajes de correo, terminar la conexión o solicitar al servidor que intercambie las funciones de emisor y receptor para que los mensajes puedan fluir en la dirección opuesta. El receptor debe enviar un acuse de recibo por cada mensaje. También puede abortar la conexión completa o la transferencia del mensaje actual.

Las transacciones de correo comienzan con un comando *MAIL* que proporciona la identificación del emisor así como un campo *FROM*: que contiene la dirección en la que los errores se deberán reportar. Un recipiente prepara su estructura de datos para recibir un nuevo mensaje de correo y responde al comando *MAIL* enviando la respuesta *250*. La respuesta *250* significa que todo está bien. La respuesta completa consiste en el texto *250 OK*. Como con otros protocolos de aplicación, los programas leen los comandos abreviados y los números de tres dígitos al comienzo de las líneas; el texto restante es un intento por ayudar a los usuarios a depurar el software de correo.

Luego de un comando *MAIL* exitoso, el emisor emite una serie de comandos *RCPT* que identifican a los recipientes del mensaje de correo. Los receptores deben enviar un acuse de recibo por cada comando *RCPT* enviando un *250 OK* o el mensaje de error *550 No such user here*.

Después de que todos los comandos *RCPT* han sido reconocidos, el emisor emite un comando *DATA*. En esencia, un comando *DATA* informa al receptor que el emisor está listo para transferir un mensaje de correo completo. El receptor responde con el mensaje *354 Start mail input* y especifica

[7] *HELO* es una abreviatura de "hello".

la secuencia de caracteres utilizada para terminar el mensaje de correo. El fin de la secuencia consiste en cinco caracteres: retorno de carro, alimentación de línea, punto, retorno de carro y alimentación de línea.[8]

Un ejemplo aclarará el intercambio SMTP. Supongamos que el usuario *Smith* en el anfitrión *Alpha.EDU* envía un mensaje al usuario *Jones, Green* y *Brown* en el anfitrión *Beta.GOV*. El software cliente SMTP en el anfitrión *Alpha.EDU* contacta al software servidor SMTP en el anfitrión *Beta.GOV* y comienza el intercambio como se muestra en la figura 25.3.

```
S: 220 Beta.GOV Simple Mail Transfer Service Ready
C: HELO Alpha.EDU
S: 250 Beta.GOV

C: MAIL FROM:<Smith@Alpha.EDU>
S: 250 OK

C: RCPT TO:<Jones@Beta.GOV>
S: 250 OK

C: RCPT TO:<Green@Beta.GOV>
S: 550 No such user here

C: RCPT TO:<Brown@Beta.GOV>
S: 250 OK

C: DATA
S: 354 Start mail input; end with <CR><LF>.<CR><LF>
C: ...sends body of mail message...
C: ...continues for as many lines as message contains
C: <CR><LF>.<CR><LF>
S: 250 OK

C: QUIT
S: 221 Beta.GOV Service closing transmission channel
```

Figura 25.3 Ejemplo de una transferencia SMTP de Alpha.EDU hacia Beta.GOV. Las líneas que comienzan con "C:" son transmitidas por el cliente (Alpha) y las líneas que comienzan con "S:" son transmitidas por el servidor. En el ejemplo, la máquina Beta.GOV no reconoce al recipiente Green.

En el ejemplo, el servidor rechaza el recipiente *Green* porque no reconoce el nombre como un destino de correo válido (es decir que no es un usuario ni una lista de correos). El protocolo SMTP no específica los detalles de cómo maneja un cliente estos errores —el cliente debe decidir. Aun

[8] El SMTP utiliza *CR LF* para terminar una línea y prohíbe que el cuerpo de un mensaje de correo tenga un punto en una línea.

cuando los clientes pueden abortar la entrega completamente si se presenta un error, la mayor parte de los clientes no lo hace. Por el contrario, continúan con la entrega hacia todos los recipientes válidos y, luego, reportan los problemas al emisor original. Por lo general, el cliente reporta los errores mediante correo electrónico. El mensaje de error contiene un resumen de los errores así como el encabezado del mensaje de correo que ha ocasionado el problema.

Una vez que el cliente ha terminado de enviar todos los mensajes de correo a su destino particular, puede emitir el comando *TURN*[9] para cambiar la conexión. Si esto sucede, el receptor responde con un *250 OK* y asume el control de la conexión. Con las funciones invertidas, el lado en el que originalmente estaba un servidor envía de regreso cualquier mensaje de correo en espera. Cualquier lado que controle la interacción puede elegir terminar la sesión. Para hacerlo, emiten el comando *QUIT*. El otro lado responde con el comando *221*, el cual significa que está de acuerdo en terminar. Entonces, ambos lados cierran la conexión TCP cortésmente.

El SMTP es mucho más complejo de como lo hemos bosquejado aquí. Por ejemplo, si un usuario se ha movido, el servidor puede conocer la dirección de correo nueva del usuario. El SMTP permite que el servidor elija informar al cliente acerca de direcciones nuevas que el cliente pueda utilizar en el futuro. Cuando se informa al cliente sobre nuevas direcciones, el servidor debe elegir enviar el correo que ha activado el mensaje o solicitar que el cliente asuma la responsabilidad para el envío.

25.10 La extensión MIME para datos no ASCII

Para permitir la transmisión de datos no ASCII a través de e-mail, la IETF definió la *Multipurpose Internet Mail Extension (MIME)*. La MIME no cambia al SMTP ni lo reemplaza. De hecho, la MIME permite que datos arbitrarios sigan codificándose en ASCII y luego se envíen por medio de mensajes e-mail estándar. Para adaptarse a tipos y representaciones arbitrarias de datos, cada mensaje MIME incluye datos que informan al recipiente del tipo de datos y de la codificación utilizada. La información de MIME reside en el encabezado de correo 822 —la línea del encabezado MIME que especifica la versión de MIME utilizada, el tipo de datos que se envían y la codificación empleada para convertir los datos en ASCII. Por ejemplo, la figura 25.4 ilustra un mensaje MIME que contiene una fotografía en la representación estándar *GIF*.[10] La imagen GIF ha sido convertida en una representación ASCII de siete bits mediante la codificación *base64*.

```
From: bill@acollege.edu
To: john@somewhere.com
MIME-Version: 1.0
Content-Type: image/gif
Content-Transfer-Encoding: base64

    ...data for the image...
```

Figura 25.4 Ejemplo de un mensaje MIME. Las líneas en el encabezado identifican el tipo de datos así como la codificación utilizada.

[9] En la práctica, pocos servidores de correo utilizan el comando *TURN*.
[10] GIF es el Graphics Interchange Format (formato de intercambio de gráficos).

En la figura, la línea del encabezado *MIME-Version:* establece que el mensaje fue compuesto por medio de la versión *1.0* del protocolo MIME. *Content-Type:* establece que la información es una imagen GIF y *Content-Transfer-Encoding*; el encabezado establece que la decodificación *base64* se utilizó para convertir la imagen ASCII. Para ver la imagen, establece que el sistema de correo del receptor debe convertir la codificación *base64* a binario, y luego correr una aplicación que presente una imagen GIF en la pantalla del usuario.

El estándar MIME especifica que una declaración *Content-Type* debe contener dos identificadores: *un tipo de contenido* y un *subtipo*, separados por una diagonal. En el ejemplo, *imagen* es el tipo de contenido y *gif* es el subtipo.

El estándar define siete tipos de contenidos básicos, los subtipos válidos para cada uno y las codificaciones de transferencia. Por ejemplo, aun cuando una *imagen* puede tener los subtipos *jpeg* o *gif*, el *texto* no puede utilizar ningún subtipo. Además de los tipos estándar y los subtipos, MIME permite a un emisor y a un receptor definir tipos de contenido privado.[11] La figura 25.5 lista los siete tipos de contenidos básicos.

Tipo de contenido	Se utiliza cuando los datos en el mensaje son
text	Texto (por ejemplo, un documento)
image	Imágenes estáticas o imágenes generadas en computadora
audio	Grabaciones de sonido
video	Grabaciones de video que incluyen movimiento
application	Datos para un programa
multipart	Mensajes múltiples de los que cada uno tiene una codificación y un tipo de contenido diferentes
message	Mensajes e-mail completos (por ejemplo, un memorándum que se está enviando) o una referencia externa a un mensaje (por ejemplo, un servidor FTP y un servidor de archivo)

Figura 25.5 Los siete tipos básicos que pueden aparecer en una declaración *Content-Type* de MIME y sus significados.

25.11 Mensajes MIME multipart

El tipo de contenido multipart de MIME es útil pues añade una flexibilidad considerable. El estándar define cuatro posibles subtipos para un mensaje multipart, cada uno proporciona una funcionalidad importante. El subtipo *mixed* permite que un solo mensaje contenga submensajes independientes, de los que cada uno tiene un tipo independiente y una codificación diferente. Los mensajes multipart mezclados hacen posible incluir textos, gráficos y audio en un solo mensaje, o permiten el envío de un memorándum con segmentos de dato adicionales asociados, similares a los *enclosures* incluidos en una carta de negocios. El subtipo *alternative* permite que un solo mensaje incluya varias representaciones de los mismos datos. Algunas alternativas de los mensajes multipart son útiles cuando

[11] Para evitar posibles conflictos en los nombres, el estándar requiere que los nombres seleccionados para contenidos privados comiencen la cadena con *X*.

se envía un memorándum a muchos recipientes de los que no todos utilizan el mismo hardware y software de sistema. Por ejemplo, se puede enviar un documento como texto en ASCII y con formato, permitiendo que los recipientes que tienen computadoras con capacidades gráficas seleccionen la opción con formato. El subtipo *parallel* permite que un solo mensaje incluya subpartes que deben ser vistas juntas (por ejemplo, subpartes de audio y video que deben presentarse de manera simultánea). Por último, el subtipo *digest* permite que un solo mensaje contenga un conjunto de otros mensajes (por ejemplo, la colección de mensajes e-mail de una discusión).

La figura 25.6 ilustra uno de los principales usos de un mensaje multipart: un mensaje e-mail puede contener un texto corto que explique el propósito del mensaje y otra parte que contenga información no textual. En la figura, una nota en la primera parte del mensaje explica que la segunda parte contiene una imagen fotográfica.

```
From: bill@acollege.edu
To: john@somewhere.com
MIME-Version: 1.0
Content-Type: Multilpart/Mixed; Boundary=StartOfNextPart

--StartOfNextPart
John,
Esta es la fotografía de nuestro laboratorio de investigación
que prometí enviarte. Aquí puedes ver el equipo que donaste.

Gracias de nuevo,
Bill

--StartOfNextPart
Content-Type: image/gif
Content-Transfer-Encoding: base64
     ...data for the image...
```

Figura 25.6 Ejemplo de un mensaje multipart mezclado de MIME. Cada parte de este mensaje puede tener un tipo de contenido independiente.

La figura también ilustra unos cuantos detalles de MIME. Por ejemplo, cada línea de encabezado puede contener parámetros de la forma $X=Y$ después de las declaraciones básicas. La palabra clave *Boundary*= que sigue a la declaración de tipo de contenido multipart en el encabezado define la cadena utilizada para separar partes del mensaje. En el ejemplo, el emisor ha seleccionado la cadena *StartOfNextPart* para que sirva como límite demarcador. Las declaraciones de tipo de contenido y de codificación de transferencia para un submensaje, si se incluye, se siguen inmediatamente a la línea de demarcación. En el ejemplo, el segundo submensaje se declara como una imagen GIF.

25.12 Resumen

El correo electrónico es uno de los servicios de aplicación disponibles más ampliamente utilizados. *Como en la mayor parte de los servicios TCP/IP, utiliza el paradigma* cliente-servidor. Los búfers del sistema de correo de los mensajes que entran y salen permiten que la transferencia desde el cliente y el servidor se realice como un proceso subordinado.

El conjunto de protocolos TCP/IP proporciona estándares separados para el formato de los mensajes de correo y la transferencia de correo. El formato de mensaje de correo conocido como *822 utiliza una línea en blanco para separar el encabezado de mensaje* del cuerpo del mensaje. El Simple Mail Transfer Protocol (SMTP) define cómo un sistema de correo en una máquina transfiere correo hacia el servidor en otra.

Las Multipurpose Internet Mail Extensions (MIME) proporcionan un mecanismo que permite que datos arbitrarios se transfieran mediante el SMTP. MIME añade líneas al encabezado de un *mensaje e-mail para definir el tipo de datos y la codificación utilizada.* El tipo multipart mezclado de MIME permite que un solo mensaje contenga varios tipos de datos.

PARA CONOCER MÁS

Los protocolos descritos en este capítulo se especifican en los RFC de Internet. Postel (RFC 821) describe el Simple Mail Transfer Protocol y proporciona muchos ejemplos. El formato exacto de los mensajes de correo se encuentra en Crocker (RFC 822). Borenstein y Freed (RFC 1521) especifican el estándar para MIME, incluyendo la sintaxis de las declaraciones del encabezado, la interpretación de los tipos de contenido y la codificación *base64* mencionada en este capítulo. Moore (RFC 1522) define la extensión de los encabezados MIME para texto no ASCII y Postel (RFC 1590) describe el procedimiento para registrar nuevos contenidos y tipos de codificación. Partridge (RFC 974) analiza la relación entre el ruteo de correo y el sistema nombre de dominio. Horton (RFC 976) propone un estándar para el sistema de correo electrónico UUCP de UNIX.

EJERCICIOS

25.1 Algunos sistemas de correo obligan al usuario a especificar una secuencia de máquinas a través de las cuales el mensaje debe viajar para alcanzar su destino. El protocolo de correo en cada máquina sólo pasa el mensaje a la siguiente máquina. Liste tres desventajas de cada esquema.

25.2 Averigüe si su sistema de cómputo permite invocar el SMTP de manera directa.

25.3 Construya un cliente SMTP y utilícelo para entregar un mensaje de correo.

25.4 Pruebe si usted puede enviar un mensaje de correo a través de una compuerta de correo y devolverlo hacia sí mismo.

25.5 Haga una lista de las formas de direcciones de correo que su localidad maneja y escriba un conjunto de reglas para traducirlas.

25.6 Descubra cómo puede utilizarse el programa *sendmail* de UNIX para implantar una compuerta de correo.

25.7 Averigüe con qué frecuencia su sistema de correo local trata de realizar entregas y durante cuánto tiempo realiza los intentos.

25.8 Muchos sistemas de correo permiten a los usuarios dirigir el correo entrante hacia un programa en lugar de almacenarlo en un buzón. Construya un programa que acepte su correo entrante, coloque su correspondencia en un archivo y luego envíe una réplica para indicar al emisor que usted está de vacaciones.

25.9 Lea el estándar SMTP cuidadosamente. Luego, use TELNET para conectarse a un puerto SMTP en una máquina remota y solicite al servidor SMTP remoto que expanda un alias de correo.

25.10 Un usuario recibe correspondencia cuyo campo *To* especifica la cadena *gente-importante*. El correo fue enviado desde una computadora en la que el alias *gente-importante* incluye identificadores de buzón no válidos. Lea las especificaciones SMTP cuidadosamente para ver cómo es posible tal situación.

25.11 Lea el estándar MIME cuidadosamente. ¿Qué servidores pueden especificarse en una referencia externa MIME?

26

Aplicaciones: manejo de Internet (SNMP, SNMPv2)

26.1 Introducción

Además de los protocolos que proporcionan servicios a nivel de red y los programas de aplicación que los utilizan, una red de redes necesita software que permita a los administradores depurar problemas, controlar rutas y localizar computadoras que violen los estándares de los protocolos. Nos referiremos a estas actividades como *administración de red de redes*. En este capítulo, se considera las ideas que subyacen en el software de administración de red de redes TCP/IP y se describe un protocolo de administración de red de redes.

26.2 Nivel de los protocolos de manejo

Originalmente, muchas redes de área amplia incluían protocolos administrativos como parte de sus protocolos de nivel de enlace. Si un conmutador de paquetes tenía un comportamiento errático, el administrador de red podía instruir a los conmutadores de paquetes vecinos para que enviaran un *paquete de control* especial. Los paquetes de control hacían que el receptor suspendiera su operación normal y respondiera a los comandos del administrador. El administrador podía interrogar al conmutador de paquetes para identificar problemas, examinar o cambiar rutas, probar algunas de las interfaces de comunicación o arrancar el conmutador. Una vez que los administradores resolvían el problema, podían instruir al conmutador para que continuara con su operación normal. Como las herramientas administrativas eran parte de los protocolos de nivel más bajo, los adminis-

tradores casi siempre eran capaces de controlar los conmutadores aun cuando los protocolos de alto nivel fallaran.

A diferencia de una red de área amplia homogénea, una red de redes TCP/IP no tiene un solo protocolo de nivel de enlace. Por el contrario, la red de redes consiste en varias redes físicas interconectadas por ruteadores IP. Como resultado, la administración de red de redes difiere de la administración de red. En primer lugar, un solo administrador puede controlar ruteadores heterogéneos.[1] En segundo lugar, el control completo no puede compartirse en un protocolo de nivel de enlace común. En tercer lugar, el conjunto de máquinas que controla un administrador puede localizarse en puntos arbitrarios en una red de redes. En particular, quizás un administrador necesite llevar el control de una o más máquinas que no estén conectadas al arreglo de la red física como lo están las computadoras del administrador. Así, podría no ser posible para un administrador comunicarse con las máquinas que se están controlando a menos que el software de administración utilice protocolos que proporcionen conectividad de extremo a extremo a través de la red de redes. Como consecuencia, el protocolo de administración de red de redes utilizado con el TCP/IP opera sobre el nivel de transporte:

> *En una red de redes TCP/IP, los ruteadores IP forman los conmutadores activos que los administradores necesitan para las funciones de revisión y control. Dado que los ruteadores conectan redes heterogéneas, los protocolos para la administración de red de redes operan en el nivel de aplicación y se comunican mediante los protocolos de nivel de transporte del TCP/IP.*

Diseñar el software de administración de red de redes para que opere en el nivel de aplicación tiene varias ventajas. Dado que los protocolos pueden diseñarse sin observar el hardware de red subyacente, un conjunto de protocolos puede utilizarse para todas las redes. Como los protocolos se pueden diseñar sin considerar el harware en la máquina de administración, los mismos protocolos pueden utilizarse para todos los dispositivos de administración. Desde el punto de vista de un administrador, tener un solo conjunto de protocolos de administración significa contar con cierta uniformidad —todos los ruteadores responden exactamente al mismo conjunto de comandos. Además, dado que el software de administración utiliza el IP para comunicarse, un administrador puede controlar los ruteadores a lo largo de una red de redes TCP/IP completa, sin tener conexiones directas con todas las redes físicas o ruteadores.

Por supuesto, construir el software de administración en el nivel de aplicación también tiene desventajas. A menos que el sistema operativo, el software IP y el software de protocolo de transporte trabajen de manera correcta, el administrador no será capaz de contactar a un ruteador. Por ejemplo, si las tablas de ruteo de un ruteador se dañan, puede ser imposible corregir la tabla o arrancar la máquina desde una localidad remota. Si el sistema operativo en un ruteador queda fuera de funcionamiento, será imposible accesar el programa de aplicación que implante los protocolos de administración de la red de redes, aun cuando el ruteador pueda devolver interrupciones de hardware y paquetes de ruteo.

---

[1] Aun cuando los administradores pueden controlar tanto los ruteadores como los anfitriones, nos enfocaremos en el control de los ruteadores debido a que presentan una mayor complejidad.

26.3 Modelo arquitectónico

Aun con las desventajas potenciales, tener el software de administración del TCP/IP operando en el nivel de aplicación ha funcionado bien en la práctica. La ventaja más significativa de colocar los protocolos de administración de red en un nivel elevado se pone de manifiesto cuando se considera una red de redes extensa, en la cual la computadora del administrador no necesita conectarse directamente hacia todas las redes físicas que contienen entidades administradas. La figura 26.1 muestra un ejemplo de la arquitectura.

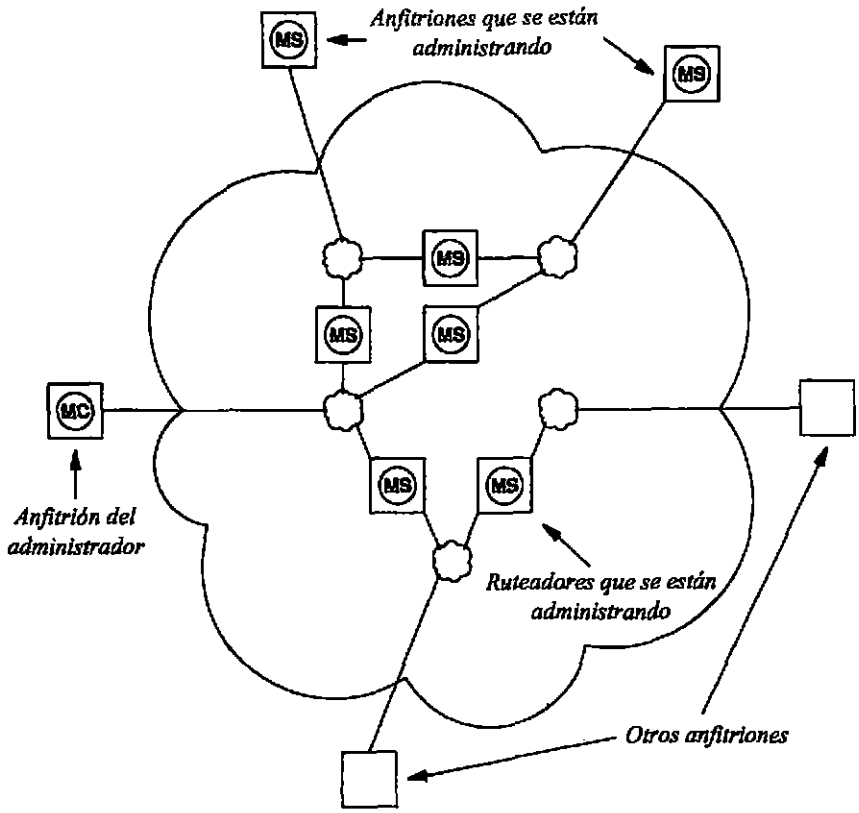


Figura 26.1 Ejemplo de administración de red. Un administrador invoca software cliente de administración (MC) que contacta al software servidor de administración (MS) en los ruteadores, a través de la red de redes.

Como se muestra en la figura, cada anfitrión o ruteador participante corre un programa servidor. Técnicamente, el servidor se conoce como *agente administrador*. Un administrador invoca al software cliente en la computadora anfitrión local y especifica al agente con el que se comunica. Luego de que el cliente contacta al agente, envía solicitudes para obtener información o manda co-

mandos para cambiar las condiciones en el ruteador. Por supuesto, no todos los ruteadores en una red de redes extensa quedan bajo una sola administración. La mayoría de los administradores sólo controla unos cuantos ruteadores en la localidad de su zona.

El software de administración de red de redes utiliza un mecanismo de autenticación para asegurarse de que sólo los administradores autorizados puedan accesar o controlar un ruteador en particular. Algunos protocolos de administración soportan varios niveles de autorización, lo que permite privilegios específicos de administración en cada ruteador. Por ejemplo, un ruteador específico podrá configurarse para permitir que varios administradores obtuvieran información mientras que sólo se permitiría que un subconjunto seleccionado de éstos mismos cambiara información o controlara los ruteadores.

26.4 Arquitectura de protocolo

Los protocolos de administración de red[2] TCP/IP dividen el problema de la administración en dos partes y especifican estándares separados para cada parte. La primera parte se relaciona con la comunicación de información. Un protocolo especifica cómo se comunica el software cliente que corre en el anfitrión del administrador con un agente. El protocolo define el formato y el significado de los mensajes que intercambian los clientes y los servidores así como la forma de nombres y direcciones. La segunda parte se relaciona con los datos que se están administrando. Un protocolo especifica qué aspectos de los datos debe conservar un ruteador así como el nombre de cada aspecto de tales datos y la sintaxis utilizada para expresar el nombre.

26.4.1 Un protocolo estándar de administración de red

El protocolo estándar de administración de red del TCP/IP actualmente en uso es el *Simple Network Management Protocol* (*SNMP*). Se había aprobado una segunda versión, pero aún no se usaba ampliamente cuando se escribió este libro. Conocida como *SNMPv2*, la nueva versión añade más capacidades, incluyendo una seguridad más confiable.

26.4.2 Un estándar para la información administrada

Un ruteador administrado debe conservar el control y los estados de información que el administrador puede accesar. Por ejemplo, un ruteador mantiene estadísticas del estado de sus interfaces de red, del tráfico que entra y sale, de los datagramas eliminados y de los mensajes de error generados. Aun cuando se permite al administrador accesar estas estadísticas, el SNMP no especifica

[2] Técnicamente, hay una distinción entre los protocolos de administración de red de redes y los protocolos de administración de red. Históricamente, sin embargo, los protocolos de administración de red de redes TCP/IP se conocen como protocolos de *administración de red*; aquí seguiremos el manejo de esta terminología.

exactamente qué datos se pueden accesar. De hecho, un estándar separado especifica los detalles. Conocido como *Management Informacion Base* (*MIB*), el estándar especifica los elementos de los datos que un anfitrión o un ruteador deben conservar y las operaciones permitidas en cada uno. Por ejemplo, el MIB especifica qué software IP debe llevar una cuenta de todos los objetos que llegan en cada interfaz de red y especifica cuál es el único software de administración de red que puede leer estos valores.

El MIB para TCP/IP divide la información de la administración en 8 categorías, como se muestra en la figura 26.2. La selección de categorías es importante pues los identificadores utilizados para especificar características incluyen un código para la categoría.

Categoría MIB	Incluye información sobre
system	Sistema operativo del anfitrión o del ruteador
interfaces	Interfaz de red individual
addr. trans.	Dirección de traducción (por ejemplo, transformación ARP)
ip	Software de Protocolo de Internet
icmp	Software de Protocolo de Mensajes de Control de Internet
tcp	Software de Protocolo de Transmisión de Internet
udp	Software de Protocolo de Datagrama de Usuario
egp	Software de Protocolo de Compuerta Exterior

Figura 26.2 Categorías de información en el MIB. La categoría es codificada en el identificador utilizado para especificar un objeto.

Mantener la definición MIB independiente del protocolo de administración de red tiene ventajas tanto para vendedores como para usuarios. Un vendedor puede incluir software agente SNMP en un producto como un ruteador con la garantía de que el software continuará cumpliendo con el estándar luego de que se definan nuevas características MIB. Un cliente puede utilizar el mismo software cliente de administración de red para administrar varios ruteadores que tengan diferentes versiones de MIB. Por supuesto, un ruteador que no tenga nuevas características MIB no puede proporcionar la información de éstas características. Sin embargo, como todos los ruteadores utilizan el mismo lenguaje para comunicarse, todos pueden analizar una solicitud y proporcionar la información solicitada o enviar un mensaje de error explicando que no cuentan con la característica solicitada.

26.5 Ejemplos de variables MIB

Además del estándar MIB del TCP/IP, que se conoce como *MIB-II*, muchos RFC documentan variables MIB para dispositivos específicos. Si se examina algunas de las características de datos que incluye el estándar MIB se podrá aclarar su contenido. La figura 26.3 lista ejemplos de variables MIB junto con sus categorías.

Variable MIB	Categoría	Significado
sysUpTime	sistema	Tiempo desde el último arranque
ifNumber	interfaz	Número de interfaces de red
ifMtu	interfaz	MTU para una interfaz en particular
ipDefaultTTL	ip	Valor IP utilizado en el campo de tiempo límite
ipInReceives	ip	Número de datagramas recibidos
ipForwDatagrams	ip	Número de datagramas enviados
ipOutNoRoutes	ip	Número de fallas de ruteo
ipReasmOKs	ip	Número de datagramas reensamblados
ipFragOKs	ip	Número de datagramas fragmentados
ipRoutingTable	ip	Tabla de ruteo IP
icmpInEchos	icmp	Número de Solicitudes de Eco ICMP recibidas
tcpRtoMin	tcp	Tiempo de retransmisión mínimo TCP permitido
tcpMaxConn	tcp	Conexión TCP máxima permitida
tcpInSegs	tcp	Número de segmentos que TCP ha recibido
udpInDatagrams	udp	Número de datagramas UDP recibidos
egpInMsgs	egp	Número de mensajes EGP recibidos

Figura 26.3 Ejemplos de variables MIB junto con sus categorías.

Los valores para la mayor parte de las características listadas en la figura 26.3 pueden almacenarse en un solo entero. Sin embargo, el MIB también define estructuras más complejas. Por ejemplo, la variable *ipRoutingTable* se refiere a la tabla de ruteo de un ruteador. Otras variables MIB definen el contenido de una entrada de información en una tabla de ruteo y permiten a los protocolos de administración de red referirse a los datos de entradas individuales. Por supuesto, las variables MIB presentan sólo una definición lógica de cada característica de datos —la estructura interna de datos que un ruteador utiliza puede diferir de la definición MIB. Cuando llega una solicitud, el software en el agente dentro del ruteador es responsable de la transformación entre la variable MIB y la estructura de datos que el ruteador utiliza para almacenar información.

26.6 Estructura de la información de administración

Además del estándar MIB, el cual especifica variables de administración de red y sus significados, un estándar separado especifica un conjunto de reglas utilizadas para definir e identificar variables MIB. Las reglas se conocen como especificaciones *Structure of Management Information (SMI)*. Para mantener los protocolos de administración de red simples, SMI establece restricciones a los tipos de variables permitidas en MIB, especifica las reglas para nombrar tales variables y crea reglas para definir tipos de variables. Por ejemplo, el estándar SMI incluye definiciones de términos como *IpAddress* (definiéndolo como una cadena de cuatro octetos) y *Counter* (definida como un entero en el rango de 0 a 2^{32}-1) y especifica que son los términos utilizados para definir variables MIB. Algo muy importante, las reglas en SMI describen cómo se refiere MIB a las tablas de valores (por ejemplo, la tabla de ruteo IP).

26.7 Definiciones formales mediante la ASN.1

El estándar SMI especifica que todas las variables MIB deben definirse y ser referidas por medio de la *Abstract Syntax Notation 1 (ASN.1)*[3] de ISO. La ASN.1 es un lenguaje formal que tiene dos características principales: una notación utilizada en documentos que los usuarios pueden leer y una representación codificada compacta de la misma información empleada en los protocolos de comunicación. En ambos casos, la notación formal precisa suprime cualquier posible ambigüedad tanto de la representación como del significado. Por ejemplo, en lugar de decir que una variable contiene un valor entero, un diseñador de protocolos que utilice ASN.1 debe establecer la forma exacta y el rango de los valores numéricos. Esta apreciación es en especial importante cuando las implantaciones incluyen computadoras heterogéneas de las que no todas utilizan la misma representación para los datos.

Además de hacer que los documentos estándar estén libres de ambigüedades, la ASN.1 también ayuda a simplificar la implantación de protocolos de administración de red y garantiza su interoperabilidad. Define con precisión cómo codificar los nombres y los datos en un mensaje. Así, una vez que la documentación de un MIB ha sido expresada por medio de la ASN.1, la forma que puede ser leída por los usuarios puede traducirse de manera directa y en forma mecánica hacia una forma codificada utilizada en los mensajes. En resumen:

> *Los protocolos de administración de red TCP/IP utilizan una notación formal llamada ASN.1 para definir nombres y tipos de variables en el manejo básico de la información. La notación precisa hace que la forma y el contenido de las variables se mantenga libre de ambigüedades.*

26.8 Estructura y representación de nombres de objetos MIB

Hemos dicho que la ASN.1 especifica cómo representar datos y nombres. Sin embargo, entender los nombres utilizados por las variables MIB requiere que entendamos algo acerca del espacio de nombres subyacente. Los nombres utilizados por las variables MIB son tomados del espacio de nombres *identificador de objetos* administrado por ISO y por ITU. La idea clave que subyace sobre el espacio de nombres identificador de objetos es que proporciona un espacio de nombre en el que se pueden nombrar todos los objetos posibles. El espacio de nombres no está restringido a variables empleadas en la administración de red —se incluyen nombres para objetos arbitrarios (por ejemplo, cada documento estándar de protocolos internacionales tiene un nombre).

El espacio de nombres identificador de objetos es *absoluto* (*global*), esto significa que los nombres están estructurados para hacerlos únicos globalmente. Como la mayor parte de los espacios de nombres son extensos y absolutos, el espacio de nombres identificador de objetos es jerárquico. La autoridad para las partes del espacio de nombres se subdivide en cada nivel, lo que permite a grupos particulares obtener autoridad para asignar algunos de los nombres sin consultar a una autoridad central para cada asignación.[4]

---

[3] ASN.1 usualmente se pronuncia con el punto: "A-S-N punto 1"

[4] Los lectores recordarán del estudio del Sistema de Nombres de Dominio, en el capítulo 22, cómo se subdivide la autoridad para un espacio de nombre jerárquico.

La raíz de la jerarquía de identificación de objetos no tiene nombre, pero posee tres descendientes directos administrados por: ISO e ITU, así como una fusión de ISO e ITU. Los descendientes se asignan a cadenas de texto cortas y a enteros para identificarlos (las cadenas de texto se utilizan cuando los usuarios necesitan entender los nombres de los objetos, el software de la computadora se vale de los enteros para formar representaciones codificadas y compactas de los nombres). ISO ha asignado un sub-árbol para uso de otras organizaciones de estándares nacionales o internacionales (incluyendo organizaciones de estándares de Estados Unidos) y el National Institute for Standards and Technology de Estados Unidos[5] ha asignado un sub-árbol para el Departamento de Defensa de Estados Unidos. Por último, el IAB ha solicitado al Departamento de la Defensa la asignación de un sub-árbol en el espacio de nombre.

La figura 26.4 ilustra las partes pertinentes de la jerarquía de identificación de objetos y muestra la posición del nodo utilizado por los protocolos de administración de red TCP/IP.

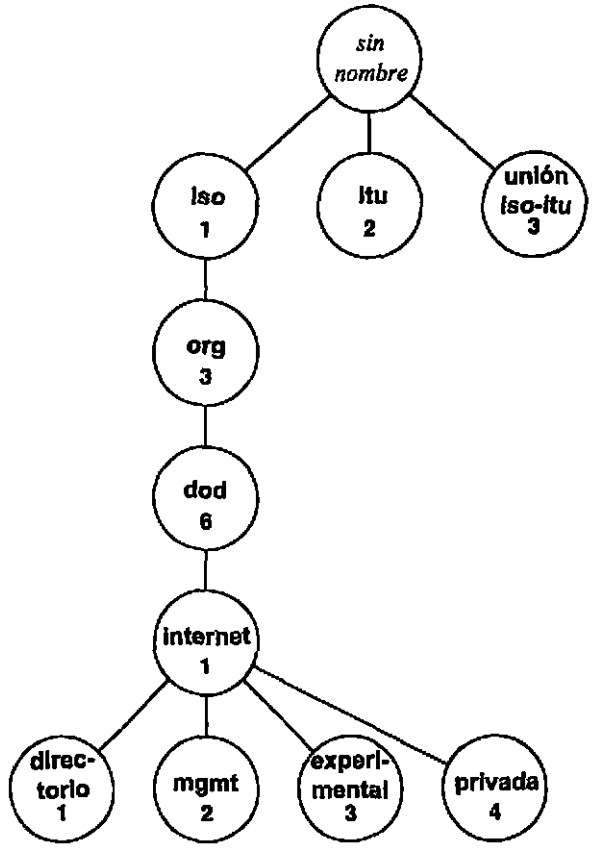


Figura 26.4 Parte de la jerarquía del espacio de nombres identificador de objetos. Un nombre de objeto consiste en una etiqueta numérica junto con una trayectoria, desde la raíz hacia el objeto.

[5] NIST formalmente es el National Bureau of Standards.

El nombre de un objeto en la jerarquía es la secuencia de etiquetas numéricas de los nodos a lo largo de la trayectoria desde la raíz hacia el objeto. La secuencia está escrita con puntos que separan a los componentes individuales. Por ejemplo, el nombre *1. 3. 6. 1. 1* denota al nodo con el nombre *directorio*. El MIB ha sido asignado a un nodo bajo el subgrupo *internet mgmt* con el nombre *mib* y el valor numérico *1*. Debido a que todas las variables MIB quedan bajo el nodo, todas tienen nombres que comienzan con el prefijo *1. 3. 6. 1. 2. 1.*

Inicialmente dijimos que todos los grupos MIB variaban dentro de ocho categorías. El significado exacto de las categorías puede explicarse ahora: estos son los ocho sub-árboles del nodo *mib* del espacio de nombres identificador de objetos. La figura 26.5 ilustra la idea y muestra parte de los sub-árboles bajo el nodo *mib*.

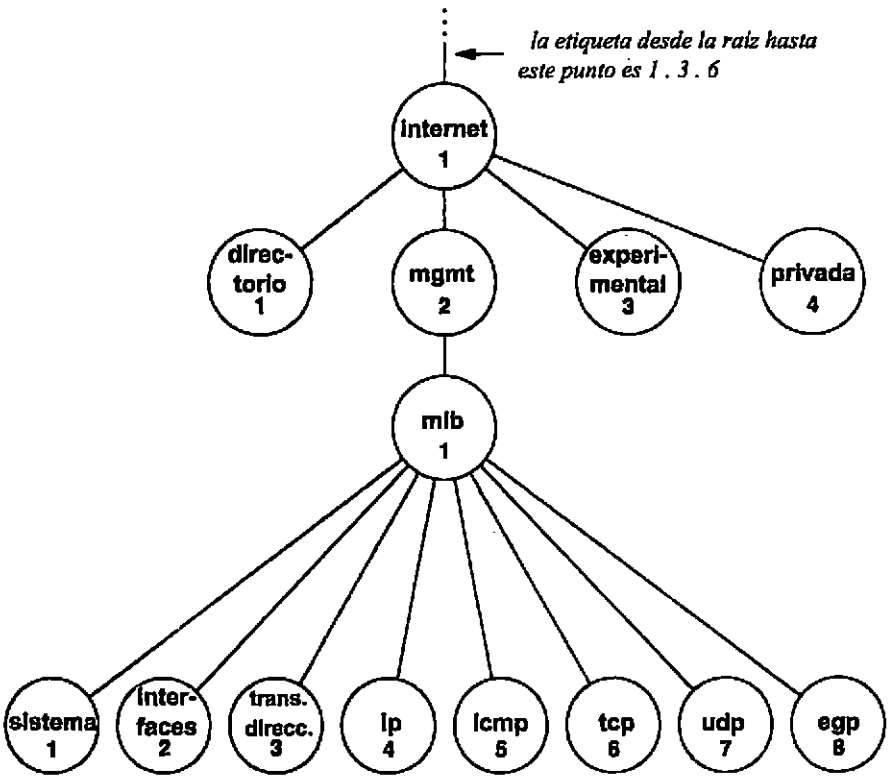


Figura 26.5 Espacio de nombre identificador de objeto bajo el nodo *mib* de IAB. Cada sub-árbol corresponde a una de las ocho categorías de las variables MIB.

Dos ejemplos aclararán la sintaxis de los nombres. La figura 26.5 muestra que la categoría con la etiqueta *ip* ha sido asignada al valor numérico *4*. Así, el nombre de todas las variables MIB

correspondientes al IP tienen un identificador que comienza con el prefijo *1. 3. 6. 1. 2. 1. 4.* Si se quisiera escribir etiquetas textuales en lugar de la representación numérica, el nombre sería:

iso . org . dod . internet . mgmt . mib . ip

Una variable MIB llamada *ipInReceives* ha sido asignada al identificador numérico *3* bajo el nodo *ip* en el espacio de nombres, así su nombre será:

iso . org . dod . internet . mgmt . mib . ip . ipInReceives

y la correspondiente representación numérica es:

1 . 3 . 6 . 1 . 2 . 1 . 4 . 3

Cuando los protocolos de administración de red utilizan nombres de variables MIB en los mensajes, cada nombre tiene un sufijo añadido. Para variables simples, el sufijo *0* hace referencia a la instancia de las variables con este nombre. Así, cuando aparece en un mensaje enviado a un ruteador, la representación numérica de *ipInReceives* es:

1 . 3 . 6 . 1 . 2 . 1 . 4 . 3 . 0

la cual hace referencia a la instancia de *ipInReceives* en este ruteador. Obsérvese que no hay manera de adivinar el valor numérico o el sufijo asignado a una variable. Se deben consultar los estándares publicados para encontrar qué valor numérico ha sido asignado a cada tipo de objeto. Así, los programas que proporcionan transformaciones entre las formas textuales y los valores numéricos subyacentes hacen esto consultando tablas de equivalencias —no hay una forma computacional estricta que realice la transformación.

Como un segundo ejemplo más complejo, consideremos la variable MIB *ipAddrTable*, la cual contiene una lista de direcciones IP para cada interfaz de red. La variable existe en el espacio de nombres como un sub-árbol bajo *ip*, y ha sido asignada al valor numérico *20*. De esta manera, su referencia tiene el prefijo:

iso . org . dod . internet . mgmt . mib . ip . ipAddrTable

con el equivalente numérico:

1 . 3 . 6 . 1 . 2 . 1 . 4 . 20

En términos de los lenguajes de programación, pensamos en una tabla de direcciones IP como en un arreglo unidimensional, en el que cada elemento del arreglo consiste en una estructura (registro) que contiene cinco elementos: una dirección IP, el índice entero de una interfaz que corresponde a una entrada de información, una máscara de subred IP, una dirección de difusión IP y un entero que especifica el tamaño máximo de datagrama que el ruteador reensamblará. Por supuesto, no todos los ruteadores tienen un arreglo en memoria. El ruteador puede guardar esta información en muchas variables o tener que seguir apuntadores para encontrarla. Sin embargo, el MIB

proporciona un nombre para el arreglo, si existe, y permite al software de administración de red en ruteadores individuales transformar las referencias de la tabla en variables internas apropiadas.

Utilizando la notación tipo ASN.1, podemos definir *ipAddrTable*:

```
ipAddrTable ::= SEQUENCE OF IpAddrEntry
```

donde *SEQUENCE* y *OF* son palabras clave que definen una *ipAddrTable* como un arreglo unidimensional de *IpAddrEntrys*. Cada elemento en el arreglo se define como si estuviera formado por cinco campos (la definición asume que *IpAddress* ya ha sido definido).

```
IpAddrEntry ::= SEQUENCE {
   ipAdEntAddr
         IpAddress,
      ipAdEntIfIndex
         INTEGER,
   ipAdEntNetMask
         IpAddress,
   ipAdEntBcastAddr
         IpAddress,
   ipAdEntReasmMaxSize
         INTEGER (0..65535)
}
```

Se debe proporcionar otras definiciones para asignar valores numéricos a *ipAddrEntry* y para cada elemento en la secuencia *IpAddrEntry*. Por ejemplo la definición:

```
ipAddrEntry { ipAddrTable 1 }
```

especifica que *ipAddrEntry* cae bajo *ipAddrTable* y tiene un valor *1*. De la misma forma, la definición:

```
ipAdEntNetMask { ipAddrEntry 3 }
```

asigna a *ipAdEntNetMask* el valor numérico *3* bajo *ipAddrEntry*.

Dijimos que *ipAddrTable* es como un arreglo unidimensional. Sin embargo, hay una diferencia importante entre la forma en que los programadores utilizan los arreglos y la forma en que el software de administración de red se vale de tablas en el MIB. Los programadores consideran que un arreglo es un conjunto de elementos que tiene un índice utilizado para seleccionar un elemento específico, por ejemplo, el programador podría escribir *xyz[3]* para seleccionar el tercer elemento del arreglo *xyz*. La sintaxis ASN.1 no utiliza índices enteros. En realidad, las tablas MIB añaden un sufijo al nombre para seleccionar un elemento específico en la tabla. En el caso de nuestro ejemplo de una tabla de dirección IP, el estándar especifica que el sufijo se utilice para seleccionar un elemento que consiste en una dirección IP. Sintácticamente, la dirección IP (en notación decimal con puntos) está unida al extremo del nombre de objeto para formar la referencia. Así, para especificar el campo de máscara de red en el elemento de la tabla de direcciones IP correspondiente a la dirección 128 . 10 . 2 . 3, se utiliza el nombre

iso.org.dod.internet.mgmt.mib.ip.ipAddrTable.ipAddrEntry.ipAdEntNetMask.128.10.2.3

el cual en forma numérica, es:

1.3.6.1.2.1.4.20.1.3.128.10.2.3

Aunque vincular un índice al extremo de un nombre puede parecer difícil, proporciona una herramienta poderosa que permite a los clientes buscar en las tablas, sin conocer el número de elementos o el tipo de datos utilizados como índice. La siguiente sección muestra cómo utilizan los protocolos de administración de red esta característica para pasar a través de una tabla, un elemento a la vez.

26.9 Protocolo de manejo de red simple

Los *protocolos de administración de red especifican la comunicación* entre un programa cliente de administración de red, que un administrador invoca, y un programa servidor de administración de red en un anfitrión o ruteador. Además de definir la forma y el significado de los mensajes intercambiados así como la representación de nombres y valores en estos mensajes, los protocolos de administración de red también definen las relaciones administrativas entre los ruteadores que son administrados. Éstos proveen la autenticación de administradores.

Se podría esperar que los protocolos de administración de red tuvieran un gran número de comandos. Algunos protocolos originales, soportaban comandos que permitían al administrador: *arrancar* el sistema, *añadir* o *borrar* rutas, *habilitar* o *inhabilitar* una interfaz de red en particular o *suprimir asignaciones de direcciones en memoria inmediata.* La principal desventaja de construir los protocolos de administración a partir de comandos se debe a la complejidad resultante. El protocolo requiere un comando separado para cada operación en un elemento de datos. Por ejemplo, el comando para borrar un elemento de la tabla de ruteo difiere del comando para inhabilitar una interfaz . Como resultado, el protocolo debe cambiar para adaptarse a nuevos elementos de datos.

El SNMP cuenta con un enfoque alternativo interesante para la administración de red. En lugar de definir un extenso conjunto de comandos, el SNMP reúne todas las operaciones en el *paradigma obtener—almacenar* (*fetch-store paradigm*).[6] Conceptualmente, el SNMP contiene sólo dos comandos que permiten a un administrador buscar y obtener un valor desde un elemento de datos o almacenar un valor en un elemento de datos. Todas las otras operaciones se definen como consecuencia de estas dos operaciones. Por ejemplo, aun cuando el SNMP no tiene una operación de *arranque* explícita, una operación equivalente puede definirse declarando un elemento de datos que proporciona el tiempo hasta el próximo arranque y permite al administrador asignar el elemento a un valor (incluyendo 0).

La mayor ventaja de usar el paradigma obtener-almacenar es la estabilidad, simplicidad y flexibilidad. El SNMP es especialmente estable ya que sus definiciones se mantienen fijas aun cuando nuevos elementos de datos se añadan al MIB y se definan nuevas operaciones como efectos

[6] El paradigma de obtener-almacenar proviene de un protocolo de administración de sistema conocido como HEMS. Para obtener más detalles ver Partridge y Trewitt (RFC 1021, 1022, 1023 y 1024).

del almacenamiento de estos elementos. El SNMP es simple en su implantación, fácil de entender y depurar porque evita la complejidad de manejar casos especiales para cada comando. Por último, el SNMP es especialmente flexible pues se puede adaptar a comandos arbitrarios dentro de una estructura elegante.

Desde el punto de vista de los administradores, por supuesto, el SNMP se mantiene oculto. El usuario de una interfaz para software de administración de red puede expresar operaciones como comandos imperativos (por ejemplo, *arrancar*). Así pues, hay una pequeña diferencia visible entre la forma en que un administrador utiliza SNMP y otros protocolos de administración de red. De hecho los vendedores han comenzado a vender software de administración de red que ofrece una interfaz gráfica de usuario. Este software presenta diagramas de la conectividad de la red y utiliza un tipo de interacción "apuntar y elegir".

Como se muestra en la figura 26.6, el SNMP ofrece más que las dos operaciones que hemos descrito.

Comando	Significado
get-request	Obtener un valor desde una variable específica
get-next-request	Obtener un valor sin conocer su nombre exacto
get-response	Replicar a una operación fetch
set-request	Almacenar un valor en una variable específica
trap	Réplica activada por un evento

Figura 26.6 Conjunto de posibles operaciones SMNP.[7] *Get-next-request* permite al administrador realizar un procedimiento iterativo a través de la tabla.

Las operaciones *get-request, get-response* y *set-request* proporcionan la búsqueda básica y las operaciones de almacenamiento (así como las réplicas a estas operaciones). El SNMP especifica qué operaciones deben ser atómicas (*atomic*), lo que significa que, si un sólo mensaje SNMP especifica operaciones en múltiples variables, el servidor realizará todas las operaciones o ninguna de ellas. En particular no se harán asignaciones si existe un error en alguna de ellas. La operación *trap* permite a los administradores programar servidores para enviar información cuando ocurra un evento. Por ejemplo, un servidor SNMP puede programarse para enviar a un administrador *trap* si una de las redes conectadas comienza a ser inaccesible (es decir, si una interfaz queda fuera de servicio).

26.9.1 Búsqueda de nombres por medio de tablas

Hemos dicho que la ASN.1 no proporciona mecanismos para declarar arreglos o para indexarlos en la forma acostumbrada. Sin embargo, es posible denotar elementos individuales de una tabla aña-

[7] SMNPv2 añade una operación *get-bulk* que permite a un administrador obtener varios valores con una sola petición.

diendo un sufijo al identificador de objetos para la tabla. Por desgracia, un programa cliente podría desear examinar elementos en la tabla para los cuales no conoce todos los sufijos válidos. La operación *get-next-request* permite que el cliente realice un procedimiento imperativo a través de una tabla sin necesidad de saber cuántos elementos contiene una tabla. Las reglas son muy sencillas. Cuando envía una *get-next-request*, el cliente proporciona el prefijo de un identificador de objeto válido, *P*. El servidor examina el conjunto de identificadores de objetos para todas las variables que controla y responde enviando un comando *get-response* para el que tiene un identificador de objeto lexicográficamente mayor que *P*. Debido a que MIB utiliza sufijos para indexar tablas, un cliente puede enviar el prefijo de un identificador de objeto correspondiente a una tabla y recibir el primer elemento en la tabla. El cliente puede mandar el nombre del primer elemento en una tabla y recibir el segundo, y así sucesivamente.

Consideremos un ejemplo de búsqueda. Recordemos que *ipAddrTable* utiliza direcciones IP para identificar elementos en una tabla. Un cliente que no sepa qué direcciones IP están en la tabla para un ruteador dado no puede formar un identificador de objeto completo. Sin embargo, el cliente tiene la posibilidad aún de utilizar la operación *get-next-request* para buscar la tabla enviando el prefijo:

iso . org . dod . internet . mgmt.mib . ip . ipAddrTable . ipAddrEntry . ipAdEntNetMask

el cual tiene la forma numérica:

1 . 3 . 6 . 1 . 2 . 1 . 4 . 20 . 1 . 3

El servidor devuelve el campo de máscara de red del primer elemento en *ipAddrTable*. El cliente utiliza el identificador de objeto completo devuelto por el servidor para solicitar el próximo elemento en la tabla.

26.10 Formato de los mensajes SNMP

A diferencia de la mayor parte de los protocolos TCP/IP, los mensajes SNMP no tienen campos fijos. Por el contrario, utilizan la codificación ASN.1 estándar. Aunque esto puede ser difícil de codificar y entender para los usuarios. Luego de examinar la definición del mensaje SNMP en la notación ASN.1, revisaremos brevemente el esquema de codificación ASN.1 y veremos un ejemplo de un mensaje SNMP codificado.

Un mensaje SNMP consiste en tres partes principales: una *versión* de protocolo, un identificador *community* de SNMP (utilizada para reunir los ruteadores administrados por un solo administrador dado), y un área *data*. El área de datos se divide en *protocol data units* (*PDUs*). Cada PDU consiste en una solicitud (enviada por el cliente) o una respuesta (mandada por el servidor). La figura 26.7 muestra cómo puede describirse el mensaje en la notación ASN.1.

```
SNMP-Message ::=
   SEQUENCE {
      version INTEGER {
         version-1 (0)
      },
      community
         OCTET STRING,
      data
         ANY
   }
```

Figura 26.7 Formato del mensaje SNMP en notación ASN.1. El área data contiene uno o más protocolos de unidades de datos.

Los cinco tipos de unidades de datos de protocolo se describen a continuación en notación ASN.1 en la figura 26.8

```
SNMP-PDUs ::=
   CHOICE {
      get-request
         GetRequest-P-DU,
      get-next-request-PDU
         GetNextRequest-PDU,
      get-response
         GetResponse-PDU,
      set-request
         SetRequest-PDU,
      trap
         Trap-PDU,
   }
```

Figura 26.8 Definición ASN.1 de un PDU del SNMP. La sintaxis para cada tipo de solicitud debe especificarse.

La definición especifica que cada unidad de datos de protocolo consiste en uno de cinco tipos de solicitudes o repuestas. Para completar la definición de un mensaje SNMP, debemos especificar la sintaxis de los cinco tipos individuales. Por ejemplo, la figura 26.9 muestra la definición de una *get-request.*

Otras definiciones en el estándar especifican los términos no definidos restantes. *Request-ID* se define como un entero de cuatro octetos (utilizado para cotejar las repuestas y las solicitudes). Tanto *ErrorStatus* como *ErrorIndex* son enteros de un solo octeto que contienen un valor cero en una solicitud. Por último, *VarBindList* contiene una lista de identificadores de objetos para los que el cliente busca valores. En términos de ASN.1 las definiciones especifican que *VarBindList* es una secuencia de pares de nombres y valores de objetos. ASN.1 representa el par como una secuencia

```
GetRequest-PDU ::= [0]
   IMPLICIT SEQUENCE {
      request-id
         RequestID,
      error-status
         ErrorStatus,
      error-index
         ErrorIndex,
      variable-bindings
         VarBindList
   }
```

Figura 26.9 Definición ASN.1 de un mensaje *get-request*. Formalmente el mensaje se define como un *GetRequest-PDU*.

de dos elementos. Así, en la solicitud más sencilla posible, *VarBindList* es una secuencia de dos elementos: un nombre y un *null*.

26.11 Ejemplo de un mensaje codificado SNMP

La forma de codificación de ASN.1 utiliza campos de longitud variable para representar elementos. En general, cada campo comienza con un encabezado que especifica el tipo de objeto y su longitud en octetos. Por ejemplo, la figura 26.10 muestra la cadena de octetos codificados en un mensaje *get-request* para elementos de datos sys*Descr* (identificador de objeto numérico *1.3.6.1.2.1.1.1*).

Como se muestra en la figura 26.10, el mensaje comienza con un código para *SEQUENCE*, el cual tiene una longitud de 41 octetos. El primer elemento en la secuencia es el entero de un octeto que especifica la *versión* del protocolo. El campo *community* se almacena en una cadena de caracteres, la cual en el ejemplo es una cadena de seis octetos que contiene la palabra *public*.

La *GetRequest-PDU* ocupa el resto del mensaje. El código inicial especifica una operación *get-Request*. Debido a que el bit de orden superior está activado, la interpretación *depende del contexto*. Es decir, el valor hexadecimal *A0* sólo especifica una *GetRequest-PDU* cuando se utiliza en un mensaje SNMP; no es un valor reservado universalmente. Luego del octeto de solicitud, el octeto de longitud especifica que la solicitud es de *28* octetos de largo. La solicitud ID es de *4* octetos, pero cada uno de los estados de error y de los índices de error son de un octeto. Por último, la secuencia de pares contiene una asignación, un solo identificador de objeto unido a un valor *null*. El identificador está codificado como se espera salvo que las dos primeras etiquetas numéricas están combinadas dentro de un solo octeto.

```
   30        29        02        01        00
SEQUENCE  len=41   INTEGER   len=1    vers=0

   04        06        70        75        62        6C        69        63
 string    len=6       p         u         b         1         i         c

   A0        1C        02        04        05        AE        56        02
getreq.   len=28   INTEGER   len=4       ______    request   ID      ______

   02        01        00        02        01        00
INTEGER    len=1    status   INTEGER   len=1   error index

   30        0E        30        0C        06        08
SEQUENCE  len=14  SEQUENCE  len=12  objectid    len=8

   2B        06        01        02        01        01        01        01
  1.3   .    6    .    1    .    2    .    1    .    1    .    1    .    1

   05        00
 null      len=0
```

Figura 26.10 Forma codificada de un *get-request* para el elemento de datos *sysDescr* con octetos que se muestran en hexadecimal y con sus significados en la parte inferior de cada uno. Los octetos relacionados se han agrupado en líneas; éstos son contiguos en los mensajes.

26.12 Resumen

Los protocolos de administración de red permiten que un administrador supervise y controle ruteadores y anfitriones. Un programa cliente de administración de red que se ejecuta en la estación de trabajo del administrador contacta uno o mas servidores, llamados agentes, los cuales corren en las computadoras que serán controladas. Debido a que una red de redes consiste en máquinas y redes heterogéneas, el software de administración TCP/IP se ejecuta como un programa de aplicación y utiliza los protocolos de transporte de la red de redes (por ejemplo, UDP) para realizar la comunicación entre cliente y servidores.

El protocolo de administración de red TCP/IP estándar es el SNMP, Simple Network Management Protocol. El SNMP define un protocolo de administración de bajo nivel que proporciona dos operaciones básicas: obtener un valor de una variable o almacenar un valor dentro de una variable. En el SNMP, todas las operaciones se dan como consecuencia de los valores que se almacenen en las variables. El SNMP define el formato de los mensajes que viajan entre la computadora del administrador y una entidad supervisada.

Un estándar asociado al SNMP define el conjunto de variables que una entidad administrada mantiene. El estándar se conoce como Management Information Base, o MIB. Las variables MIB se describen utilizando ASN.1, un lenguaje formal que proporciona una forma codificada concisa así como una notación precisa que es posible leer para los usuarios para nombres y objetos. ASN.1

utiliza un espacio de nombre jerárquico para garantizar que todos los nombres MIB sean únicos globalmente, en tanto que permiten a los subgrupos asignar partes del espacio de nombres.

PARA CONOCER MÁS

Schoffstall, Fedor, Davin y Case (RFC 1157) contienen el estándar para el SNMP. ISO (Mayo 87a) y (Mayo 87b) contiene el estándar para ASN.1 y especifica la codificación. McCloghrie y Rose (RFC 1213) definen las variables que comprenden MIB-II, pero McCloghrie y Rose (RFC 1211) contiene las reglas SMI para nombrar variables MIB.

Una serie de RFC define al SNMPv2, que es un estándar que se propuso cuando se escribía este libro. Case, McCloghrie, Rose, Waldbusser (RFC 1441) contiene una introducción al SNMPv2. Case, McCloghrie, Rose y Waldbusser [RFC 1450] definen el SNMPv2 MIB. Galvin y McCloghrie (RFC 1446) analizan los protocolos de seguridad SNMPv2. Case, McCloghrie, Rose, Waldbusser (RFC 1448) especifican las operaciones del protocolo.

Una propuesta anterior para un protocolo de administración de red, llamada HEMS, puede encontrarse en Trewitt y Partridge (RFC 1021, 1022, 1023 y 1024). Davin, Case, Fedor y Schoffstall (RFC 1028) especifican un predecesor del SNMP conocido como Simple Gateway Monitoring Protocol (SGMP).

EJERCICIOS

26.1 Capture un paquete SNMP con un analizador de red y decodifique los campos.

26.2 Lea el estándar para que sepa cómo ASN.1 codifica los dos primeros valores de un identificador de objeto en un solo octeto. ¿Por qué lo hace así?

26.3 Lea la especificación para CMIP. ¿Cuántos comandos soporta?

26.4 Suponga que los diseñadores de MIB necesitan definir una variable que corresponda a un arreglo de dos dimensiones. ¿Cómo puede la notación de ASN.1 adaptar las referencias para esta variable?

26.5 ¿Cuáles son las ventajas y las desventajas de definir globalmente nombres ASN.1 únicos para las variables MIB?

26.6 Si usted tiene código cliente SNMP disponible, trate de utilizarlo para leer variables MIB en un ruteador local. ¿Cuál es la ventaja de permitir que cualquier administrador lea variables en todos los ruteadores?

26.7 Lea la especificación de MIB para encontrar la definición de la variable *ipRoutingTable* que corresponde a una tabla de ruteo IP. Diseñe un programa que utilice el SNMP para contactar múltiples ruteadores y observe si alguna entrada en la tabla de ruteo ocasiona un ciclo cerrado de ruteo. Exactamente, ¿qué nombres ASN.1 debería generar un programa?

27

Resumen de las dependencias de protocolos

27.1 Introducción

El TCP/IP ha generado más protocolos de los que es posible tratar en una sola obra. Por ejemplo, sistemas ampliamente conocidos de información distribuida como *gopher* y *World Wide Web*, que proporcionan la capacidad de consultar y acceder a la información en forma remota, lo mismo que interfaces gráficas remotas como el sistema X-window, que permiten a los programas cliente dibujar textos y gráficas en presentaciones de mapas de bits, todos estos sistemas utilizan protocolos TCP/IP. En general, cada uno de ellos define su propio protocolo de aplicación y confía en un TCP o UDP para el transporte de extremo a extremo. De hecho, cualquier programador que construya una aplicación distribuida por medio del TCP/IP define incluso otro protocolo más a nivel de aplicación.

Aunque no es importante comprender los detalles de todos los protocolos, sí lo es saber qué protocolos existen y cómo se pueden usar. En este capítulo, se proporciona un breve resumen de las relaciones entre los principales protocolos que hemos analizado y se muestra cuáles están disponibles para usarse en programas de aplicación.

27.2 Dependencias de protocolos

En el gráfico de la figura 27.1, se muestra las dependencias entre los principales protocolos que hemos tratado. Cada polígono encerrado corresponde a un protocolo y está colocado directamente

arriba de los protocolos que utiliza. Por ejemplo, el protocolo de correspondencia SMTP depende del TCP, que a su vez depende del IP.

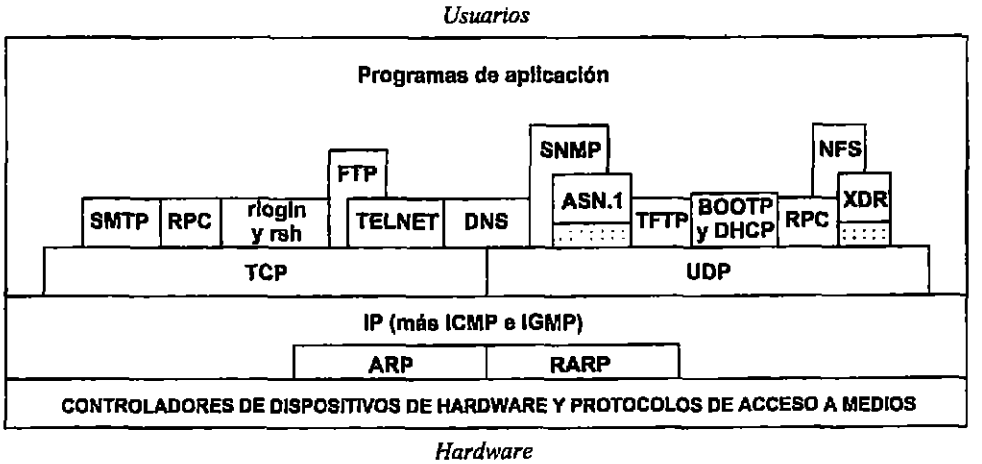


Figura 27.1 Dependencias entre los principales protocolos TCP/IP de más alto nivel. Un *protocolo utiliza estos protocolos que dependen directamente de él.* Los programas de aplicación pueden utilizar todos los protocolos que estén por encima del IP.

Para varias partes del diagrama se necesita una mayor explicación. La capa inferior representa todos los protocolos que proporciona el hardware. Este nivel comprende cada uno de los protocolos de control de hardware, así como los rangos de acceso a medios hacia la ubicación de enlace lógico. A lo largo de la obra, hemos asumido que cualquier sistema de transferencia de paquetes puede incluirse en esta capa en tanto que el IP pueda utilizarlo para transferir datagramas. De este modo, si un sistema se configura para mandar datagramas a través de un túnel, la entrada al túnel se considera como una interfaz de hardware, sin importar su implantación de software.

La segunda capa está integrada por las listas inferiores de ARP y RARP. Por supuesto, no todas las máquinas o tecnologías de red los utilizan. ARP es el más utilizado en Ethernet; RARP se emplea en raras ocasiones salvo en el caso de las máquinas sin disco. Puede haber algunos otros protocolos de enlace de direcciones, pero ninguno tiene un amplio uso.

La tercera capa de la parte inferior contiene al IP. Comprende el protocolo de mensajes de error y control requerido, el ICMP, y el protocolo de administración de grupos opcionales de multidifusión, es decir, el IGMP. Obsérvese que el IP es el único protocolo que ocupa toda la capa. Los protocolos de más bajo nivel entregan información que llega del IP y los de más alto nivel deben utilizar el IP para enviar datagramas. El IP se muestra con una dependencia directa de la capa de hardware, ya que necesita utilizar el enlace de hardware o los protocolos de acceso para transmitir datagramas después de utilizar ARP para direcciones enlazadas.

El TCP y el UDP componen la capa de transporte. Por supuesto, se ha sugerido los nuevos protocolos de transporte, pero ninguno se ha adoptado ampliamente aún.

La capa de aplicación ilustra las complejas dependencias entre los diversos protocolos de aplicación. Recordemos, por ejemplo, que el FTP emplea las definiciones de la terminal virtual de red de TELNET para definir la comunicación en su conexión de control y al TCP para formar conexiones de datos. Así pues, el diagrama muestra que el FTP depende de TELNET y del TCP. El sistema de nombres de dominio (DNS) se vale del UDP y del TCP para la comunicación, de modo que el diagrama muestra ambas dependencias. NFS de Sun depende de los protocolos de representación externa de datos (XDR) y de la llamada de procedimiento remoto (RPC). RPC aparece dos veces porque, lo mismo que el sistema de nombres de dominio, puede utilizar el UDP o el TCP.

El SNMP depende de la *Abstract Syntax Notation* (*Notación de Sintaxis Abstracta* o ASN.1) y también se vale del UDP para mandar datagramas. Como XDR y ASN.1 tan sólo describen las convenciones sintácticas y las representaciones de datos, no utilizan ni el TCP ni el UDP. De este modo, aunque se muestra que SNMP y NFS dependen del UDP, el diagrama contiene un área punteada bajo ASN.1 y XDR pues ninguno de ellos depende del UDP. De hecho, se han omitido muchos detalles en nuestro diagrama. Por ejemplo, se podría argüir que el IP depende del BOOTP/DHCP o que muchos protocolos dependen de DNS porque el software que implanta dichos protocolos requiere un enlace de nombre.

27.3 Acceso de programas de aplicación

La mayor parte de los sistemas restringe los programas de aplicación de acceso a protocolos de nivel inferior. Por lo general, un programa de aplicación puede emplear el TCP o el UDP, o bien, implantar protocolos de más alto nivel que los utilicen (como el FTP). Una aplicación puede necesitar un privilegio especial para abrir puertos específicos, pero esto es completamente diferente del acceso restringido. Algunos sistemas no tienen mecanismos que permitan a un programa de aplicación acceder al IP de manera directa; casi ninguno permite que los programas de aplicación accedan a protocolos como ARP. A pesar de las limitaciones usuales, nuestro diagrama sugiere que las aplicaciones pueden acceder al IP (en uno de los ejercicios se revisa esto con mayor profundidad).

Hay ciertos sistemas que proporcionan mecanismos de propósito especial, los cuales permiten a un programa de aplicación interactuar con los protocolos de capas inferiores. Por ejemplo, el mecanismo conocido como *packet filter* (*filtrador de paquetes*) permite a los programas con privilegio modificar el demultiplexado de las tramas. Utilizando los filtros de paquetes primitivos, un programa de aplicación establece el criterio seguido para capturar paquetes (como el programa de aplicación que especifica que desea capturar todos los paquetes con un valor dado en el campo *tipo* de la trama). Una vez que el sistema operativo acepta el comando de filtro, coloca todos los paquetes que coincidan con el tipo específico en una cola. El programa de aplicación se vale de otra parte del mecanismo de filtro de paquetes para extraer los paquetes de la cola de espera. Para tales sistemas, se debería cambiar el diagrama a fin de mostrar el acceso de aplicación a todos los niveles.

27.4 Resumen

Buena parte de la abundante funcionalidad asociada con el conjunto de protocolos TCP/IP es resultado de una gran variedad de servicios de alto nivel proporcionados por los programas de aplicación. Los protocolos de alto nivel de estos programas utilizan servicios básicos integrados: entrega de datagramas no confiable y transporte de flujo confiable. Por lo general, siguen el modelo cliente-servidor bajo el cual los servidores operan puertos de protocolos bien definidos, de manera que el cliente sabe cómo ponerse en contacto con ellos.

El nivel más alto de protocolos proporciona servicios de usuario como la transferencia de archivos y la correspondencia, lo mismo que el acceso remoto. Construir tales servicios permite obtener, como principales ventajas, una red de redes en la que se proporciona conectividad universal y la simplificación de los protocolos de aplicación. En particular, cuando lo usan dos máquinas que están enlazadas con una red de redes, los protocolos de transporte de extremo a extremo garantizan que el programa cliente en la máquina fuente se comunique de manera directa con un servidor en la máquina destino. Debido a que servicios como el correo electrónico utilizan la conexión de transporte extremo a extremo, no necesitan apoyarse en máquinas intermedias para enviar los mensajes (íntegros).

Hemos visto una gran variedad de niveles de aplicación en protocolos y las complejas dependencias que existen entre ellos. Aunque se ha definido muchos protocolos de aplicación, el correo electrónico queda como el de uso más extendido y la transferencia de archivos cuenta con la mayor parte de los paquetes en Internet.

PARA CONOCER MÁS

Uno de los aspectos que sustenta la designación de capas en los protocolos es la ubicación óptima de la funcionalidad del protocolo. Edge (1979) compara los enfoques de protocolos de extremo a extremo con los de salto a salto. Saltzer, Reed y Clark (1984) arguyen que tienen la ejecución de *más alto nivel en protocolos con el conocimiento* y detección de errores de los de extremo a extremo. En una serie de documentos, Mills propone la aplicación de protocolos para la sincronización de relojes y lo reporta en experimentos (RFC 956, 957 y 958).

EJERCICIOS

27.1 Es posible traducir algunos protocolos de aplicación a otros. Por ejemplo, podría construirse un programa que aceptara una petición FTP, la tradujera a una petición TFTP, trasladara el resultado a un servidor TFTP, para obtener un archivo, y tradujera la respuesta en forma FTP para su transmisión a la fuente original. ¿Cuáles serían las ventajas y desventajas de dicha traducción de protocolos?

27.2 Consideremos la traducción descrita en la pregunta anterior. ¿Qué pares de protocolos de la figu 27.1 serían receptivos de tales traducciones?

27.3 En la figura 27.1 se sugiere que algunos programas de aplicación invocados por los usuarios pueden necesitar acceso al IP sin usar el TCP o el UDP. Encuentre ejemplos de tales programas. (Sugerencia: piense en el ICMP.)

27.4 En el diagrama de la figura 27.1 ¿dónde cabría el EGP?

27.5 DNS permite el acceso mediante el TCP y el UDP. Averigüe si su sistema operativo local permite que un solo proceso acepte conexiones TCP y peticiones UDP.

27.6 Elija una aplicación compleja, como el *sistema Xwindow* y averigüe qué protocolos utiliza.

27.7 En el diagrama de la figura 27.1 ¿dónde cabría el RIP?

27.8 En el diagrama de la figura 27.1 se muestra que el FTP depende de TELNET. ¿El cliente de su FTP local invoca al programa TELNET o FTP contiene una implantación separada para el protocolo TELNET?

27.9 Lea sobre el programa *Mosaic* que utiliza varios protocolos de aplicación. ¿De qué manera se ajustaría un programa como *Mosaic* a la estructura de la figura 27.1?

28

Seguridad de Internet y diseño del muro de seguridad

28.1 Introducción

Como en las cerraduras utilizadas para ayudar a mantener seguras las propiedades, las computadoras y las redes de datos necesitan de ciertas precauciones que ayuden a mantener segura la información. La seguridad en un ambiente de red de redes es importante y difícil. Es importante pues la información tiene un valor significativo —la información puede comprarse y venderse de manera directa o utilizarse directamente para crear productos y servicios nuevos que proporcionan grandes ganancias. La seguridad en una red de redes resulta difícil debido a que implica entender cuándo y cómo pueden confiar los usuarios participantes, las computadoras, los servicios y las redes, uno en otro, también implica entender los detalles técnicos del hardware y los protocolos de red. La seguridad de una red completa puede confiarse a una sola computadora. Algo muy importante, dado que el TCP/IP soporta a una amplia diversidad de usuarios, servicios y redes, y debido a que una red de redes puede abarcar muchas fronteras políticas y organizacionales, los individuos y las organizaciones participantes pueden no estar de acuerdo en un nivel de confiabilidad o en las políticas para el manejo de datos.

En este capítulo se considera una técnica fundamental empleada con frecuencia para proporcionar seguridad entre organizaciones. La técnica es general ya que permite que cada organización determine los servicios y las redes que quieren mantener disponibles al exterior y la amplitud con la que el exterior puede utilizar esos recursos. Comenzaremos por revisar un poco de los conceptos y la terminología básica.

28.2 Recursos de protección

Los términos *seguridad de red* y *seguridad de información* se refieren, en sentido amplio, a la confianza de que la información y los servicios disponibles en una red no puedan accesarlos usuarios no autorizados. Seguridad implica confianza, incluyendo la integridad de los datos, la confianza en el hecho de que no existen accesos no autorizados a los recursos computacionales, de que los recursos están libres de intromisiones o de derivaciones en el cableado y de que los recursos se encuentran libres de interrupciones en el servicio. Por supuesto, así como la propiedad física no está absolutamente segura contra los delitos, ninguna red es absolutamente segura. Las organizaciones hacen un esfuerzo por lograr la seguridad de las redes por la misma razón que se hace para mantener seguros edificios y oficinas: aun cuando una organización no pueda prevenir completamente los delitos, algunas medidas de seguridad básicas pueden disuadir a quienes los cometen, haciendo que los actos ilegales sean mucho más difíciles de realizar.

Proporcionar seguridad para la información requiere protección tanto para los recursos físicos como para los abstractos. Los recursos físicos incluyen dispositivos de almacenamiento pasivo como las cintas magnéticas y los discos, así como dispositivos activos como las computadoras de los usuarios. En un ambiente de red, la seguridad física se extiende a los cables, puentes y ruteadores que comprenden la infraestructura de la red. De hecho, aunque la seguridad física se mencione casi siempre, a menudo juega un papel importante en un plan de seguridad global. Obviamente, la seguridad física puede prevenir derivaciones en el cableado. Una buena seguridad física puede también eliminar ataques como sabotajes (por ejemplo, imposibilitar a un ruteador a que haga que los paquetes sean ruteados a través de una trayectoria alternativa menos segura).

La protección de recursos abstractos como la información es frecuentemente más difícil que la seguridad física pues la información es fugaz. *La integridad de los datos* (es decir, proteger la información de cambios no autorizados) es crucial; así como la *disponibilidad de datos* (por ejemplo, garantizando que desde el exterior no se pueda evitar el acceso legítimo a los datos saturando una red con tráfico). Dado que la información puede copiarse conforme pasa a través de una red, la protección debe prevenir también lecturas no autorizadas. Esto quiere decir que la seguridad de red debe incluir una garantía de *privacía*. Dado que la información puede accesarse y transmitirse a altas velocidades, puede ser difícil diferenciar entre un acceso legítimo y uno ilegítimo cuando se desarrolla una transferencia. Algo muy importante, mientras que la seguridad física con frecuencia clasifica a la gente y los recursos dentro de categorías amplias (por ejemplo, todos los no empleados tienen prohibido utilizar una vía en particular), la seguridad para la información por lo general necesita ser más restrictiva (por ejemplo, algunas partes de los registros de un empleado están disponibles sólo para el personal oficial, otras están disponibles solamente para los jefes de los empleados y otras están disponibles para el personal de la oficina).

28.3 Necesidad de una política de información

Antes de que una organización implante un proyecto de seguridad de red, la organización debe asumir riesgos y desarrollar una política clara, considerando los accesos de información y protección. Las políticas necesitan especificar quiénes tendrán garantizado el acceso a cada parte de la in-

formación, deben establecerse las reglas individuales a seguir, se debe difundir la información hacia todo el conjunto y establecer las formas en que la organización reaccionará ante las transgresiones.

Aun cuando la necesidad de una política parece obvia, muchas organizaciones intentan hacer sus redes seguras sin decidir primero lo que significa la seguridad. En organizaciones que han adoptado una política de información general, los empleados pueden ignorar la política, las motivaciones para adoptar la política o las consecuencias de violar dicha política. Establecer una política de información y educación a los empleados es crucial ya que:

> *Las personas son por lo general el punto más susceptible de cualquier esquema de seguridad. Un trabajador malicioso, descuidado o ignorante de las políticas de información de la organización puede comprometer la seguridad.*

En consecuencia, cada empleado debe conocer la política de información de la organización y ser capaz de responder a preguntas básicas como:

- ¿Qué tan importante es la información para su organización? Por ejemplo, ¿usted trabaja para una compañía que recurre a tratos secretos para obtener ventajas sobre sus competidores?
- ¿Qué significan los derechos de autor y cuál es la política de su organización en relación al fotocopiado de la información? ¿Cómo varía la política si usted utiliza una computadora para hacer copias de información en discos flexibles?
- ¿Qué tanto de la información a la que tiene acceso puede discutir con otros empleados?, ¿con personas del exterior? Por ejemplo, ¿podría llevar al exterior el directorio telefónico de su organización?
- ¿Usted o su organización trabajan con información que proviene de otras organizaciones? ¿Puede discutir esta información con otros? Por ejemplo, ¿está motivado o desmotivado a partir de las discusiones con clientes que hacen pedidos de bienes o servicios? ¿Qué detalles sobre los pedidos de los clientes o de los tratos de negocios tiene permitido comentar con otros clientes?
- ¿Qué información puede importar a la compañía? Por ejemplo, si un amigo de una compañía competidora le entrega a usted una descripción confidencial de los planes para productos nuevos, ¿debe mostrar estos documentos a su jefe?
- ¿Puede utilizar una computadora personal y un módem del trabajo para accesar información desde un servicio de boletín electrónico? Si es así, ¿su organización establece restricciones en cuanto al uso de los datos obtenidos de esta manera?
- ¿Qué son los derechos de propiedad intelectual y cómo lo afectan a usted en su trabajo?

Como se muestra en las preguntas, una política de información debe ser lo suficientemente amplia para cubrir información representada en papel y en una computadora, y debe dirigirse a aspectos como la información "entrante" así como la "que sale" de la organización. Además, una política debe detallar directivas respecto a información intrusa en la organización, originada por clientes en el curso normal de la conducción de negocios, e información que puede deducirse o derivarse en relación a clientes por sus pedidos de bienes o servicios.

Luego que se ha establecido una política de información, lograr el nivel deseado de seguridad puede ser complejo pues hacer esto significa impulsar una política a través de toda la organización. Surgen algunas dificultades cuando se tiene contacto con organizaciones externas, y el enlace de redes hace que esta interacción se dé con frecuencia. En particular, como una red de redes puede abarcar varias organizaciones, las políticas pueden entrar en conflicto. Por ejemplo, consideremos tres organizaciones: *A, B* y *C*. Supongamos que la política de *A* permite que la información se exporte a *B* pero no a *C*. Si la política en *B* permite exportar hacia *C* la información puede fluir desde *A* hacia *C* a través de *B*. Algo muy importante, aunque el efecto final pueda comprometer la seguridad, ningún empleado en ninguna organización debe violar la política de su organización.

28.4 Comunicación, cooperación y desconfianza mutua

El ejemplo anterior muestra que una sola organización no puede garantizar una política de información arbitraria de manera aislada, de hecho, cuando una organización comunica información a otra, la última disposición de la información depende de las políticas de las dos organizaciones y de la política de otras organizaciones por las que pasará la información. El término matemático *transitividad* se ha utilizado para describir la situación: decimos que cuando tres organizaciones intercambian información, la seguridad de la política de seguridad es el *cierre transitivo* de las políticas de seguridad individuales. Como resultado:

> *Una organización no puede conocer el efecto de comunicarse e interactuar con otras a menos que las dos organizaciones acuerden un nivel de confianza recíproca.*

Así, el problema central de la seguridad de red radica en un conflicto fundamental: aunque la comunicación requiere de un grado de acuerdo recíproco entre las partes que se comunican, una red de computadora puede posibilitar la comunicación entre grupos que desconfían unos de otros. El enlace de redes agudiza el problema de la confianza pues introduce terceras partes. En particular, como los datagramas viajan a través de una red de redes de una fuente a un destino distante, pueden pasar por ruteadores y atravesar redes que son propiedad de y están operadas por organizaciones no asociadas ni con la fuente del datagrama ni con su destino. Algo muy importante, ni la localidad emisora ni la receptora controlan cómo se procesan o rutean los datagramas conforme viajan a través de la red de redes entre sus organizaciones.

28.5 Mecanismos para la seguridad de Internet

Los problemas de seguridad en las redes de redes y los mecanismos de software que ayudan a que la comunicación en la red de redes sea segura, se pueden dividir en términos generales en tres conjuntos. El primer conjunto se enfoca a los problemas de *autorización, autenticación* e *integridad*. El segundo se enfoca al problema de la *privacía* y el tercero se orienta hacia el problema de la *disponibilidad* mediante el control de acceso. Como los dos primeros conjuntos se aplican a la seguri-

dad de las computadoras en general y han sido estudiados con mayor detalle, nos concentraremos en considerar brevemente el tercero.

28.5.1 Mecanismos de autenticación

Los *mecanismos de autenticación* resuelven el problema de verificar la identificación. Muchos servidores, por ejemplo, están configurados para rechazar una solicitud a menos que la origine un cliente autorizado. Cuando un cliente hace un primer contacto, el servidor debe verificar que el cliente esté autorizado antes de prestar el servicio. Para validar la autorización, un servidor debe conocer la identidad del cliente. Por ejemplo, una forma débil de autenticación en la red de redes utiliza direcciones IP. Cuando se utiliza la autenticación de direcciones IP, un administrador configura un servidor con una lista de direcciones fuente IP válidas. El servidor examina la dirección IP fuente en cada solicitud entrante y sólo acepta solicitudes que provienen de clientes que están en la lista autorizada.

La autenticación de fuente IP es *débil* porque se puede romper fácilmente. En una red de redes, en la que los datagramas pasan a través de ruteadores y redes intermedias, la autenticación de fuente puede ser atacada desde una de las máquinas intermedias. Por ejemplo, supongamos que un impostor logra controlar un ruteador *R* que está localizado entre un cliente válido y un servidor. Para accesar al servidor, el impostor primero altera las rutas en *R* para dirigir el tráfico hacia el impostor. El impostor, entonces genera una solicitud utilizando la dirección del cliente autorizado como dirección fuente. El servidor aceptará la solicitud y enviará la réplica hacia el cliente autorizado. Cuando se alcanza el ruteador en cuestión, la réplica será dirigida hacia una ruta incorrecta por el impostor, donde podrá interceptarla. Si el impostor envía todo el tráfico, salvo las réplicas para realizar solicitudes ilegítimas, ni el cliente ni el servidor detectarán al intruso. El punto es que:

> *Un esquema de autorización que utiliza una dirección IP de una máquina remota para autenticar su identidad no puede evitar ataques de parte de impostores a través de una red de redes poco segura pues un impostor que logra el control de un ruteador intermedio puede hacer las veces de un cliente autorizado.*

Es interesante el hecho de que los clientes enfrentan el mismo problema que los servidores debido a que un impostor puede también hacer las veces de servidor. Por ejemplo, dijimos que un programa cliente es responsable del envío de correo electrónico hacia un servidor de e-mail remoto. Si la correspondencia contiene información importante, podría ser necesario para el cliente verificar que no se está comunicando con un impostor.

¿Cómo pueden los programas cliente y servidor saber si se están comunicando o no con un impostor? La respuesta radica en proporcionar un servicio confiable. Por ejemplo, una forma de servicio confiable se vale de un sistema de *cifrado de clave pública*. Para utilizar un sistema de clave pública, cada participante debe ser asignado a dos *claves* que son utilizadas para codificar y decodificar mensajes. Cada clave es un entero largo.[1] Un participante publica una clave, llamada *clave pública*, en una base de datos pública y conserva la otra clave en secreto. Un mensaje se codifica mediante una clave que se puede decodificar utilizando la otra. Por ejemplo, si un emisor em-

[1] Para hacer que el cifrado sea difícil de violar, cada clave debe contener muchos dígitos; por ejemplo, algunos esquemas requieren que cada clave contenga 50 dígitos o más.

plea una clave secreta para codificar un mensaje, un receptor puede utilizar la clave pública del emisor para decodificar el mismo. Además, conocer la clave pública no hace más fácil adivinar o calcular la clave secreta. Así, si un mensaje se decodifica de manera correcta por medio de la clave pública de un propietario, debe codificarse por medio de la clave privada del propietario. Un cliente y un servidor que utilicen cifrado de clave pública pueden estar razonablemente seguros de que su interlocutor es auténtico, aun cuando los datagramas transferidos entre ambos pasen a través de una red de redes poco segura.

28.5.2 Mecanismos de privacía

El cifrado también puede manejar problemas de *privacía*. Por ejemplo, si un emisor y un receptor utilizan un esquema de cifrado de clave pública, el emisor puede garantizar que sólo el receptor involucrado pueda leer un mensaje. Si es así, el emisor utiliza la clave pública del receptor para codificar el mensaje y el receptor su clave privada para decodificar el mensaje. Dado que sólo el receptor involucrado tiene la clave privada necesaria, ninguna otra parte puede decodificar el mensaje. Así, la privacía puede reforzarse aun cuando una tercera parte obtenga una copia de los datagramas conforme éstos pasan entre el emisor y el receptor.

Los mensajes se pueden codificar dos veces para autenticar al emisor así como para reforzar la privacía. Luego de que un emisor codifica el mensaje utilizando la clave privada del emisor, el emisor codifica nuevamente el resultado por medio de la clave pública del receptor. El receptor primero aplica su propia clave privada para obtener de nuevo el primer nivel de cifrado y luego aplica la clave pública del emisor para decodificar el mensaje original. En resumen:

> *Mecanismos como el cifrado de clave pública pueden utilizarse para ayudar a resolver los problemas de autenticación, autorización y privacía. Tanto el software del cliente como el del servidor deben ser modificados para usar estos mecanismos.*

28.6 Muros de seguridad y acceso a Internet

Los mecanismos que controlan el *acceso a la red de redes* manejan el problema del filtrado hacia una organización o red en particular de las comunicaciones no previstas. Estos mecanismos pueden ayudar a prevenir a la organización sobre la obtención de información con respecto al exterior, el cambio de información o la interrupción de comunicaciones en la red de redes interna de la organización. A diferencia de los mecanismos de autenticación y privacía que se pueden añadir a programas de aplicación, el control de acceso a la red de redes por lo general requiere cambios en componentes básicos de la infraestructura de la red de redes. En particular, un exitoso control de acceso requiere de una combinación cuidadosa de restricciones en la topología de red, en el almacenamiento intermedio de la información y en el filtrado de paquetes.

Una sola técnica ha emergido como la base para el control de acceso a la red de redes. La técnica instala un bloque conocido como *muro de seguridad* [2] en la entrada hacia la parte de la red de redes que será protegida. Por ejemplo, una organización puede colocar un muro de seguridad en su conexión de la red global de Internet para protegerse de intromisiones indeseables. Un muro de seguridad divide una red de redes en dos regiones, conocidas informalmente como el *interior* y el *exterior*. La figura 28.1 ilustra el concepto.

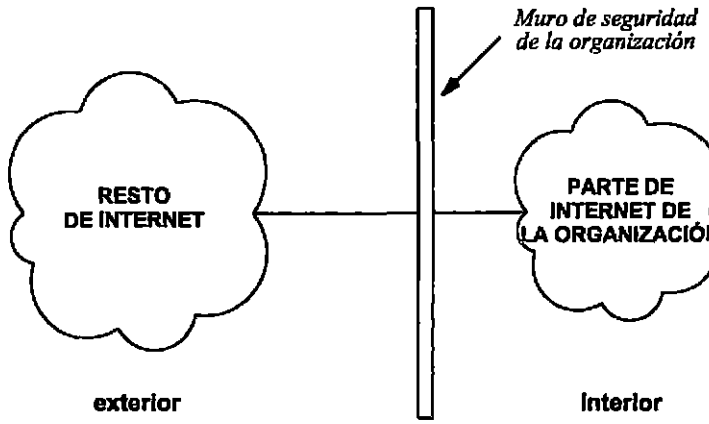


Figura 28.1 Ubicación conceptual del muro de seguridad de una red de redes que protege los ruteadores, las computadoras y los datos de la red interna de una organización contra comunicaciones indeseables que provengan del exterior.

28.7 Conexiones múltiples y vínculos más débiles

Aun cuando la imagen conceptual de la figura 28.1 parece sencilla, los detalles pueden complicar la construcción de un muro de seguridad. En particular la red de redes de una organización puede tener varias conexiones externas. Por ejemplo, si una compañía tiene una columna vertebral de red de área amplia corporativa que conecta a las localidades de la corporación en varias ciudades o países, el administrador de red en una localidad determinada, puede elegir conectar la localidad directamente a un local de negocios o una universidad. Las conexiones externas múltiples plantean un problema de seguridad especial. La organización debe formar un *perímetro de seguridad* instalando un muro de seguridad en cada conexión externa. Algo muy importante: para garantizar que este perímetro es efectivo, la organización debe coordinar todos los muros de seguridad para que utilicen exactamente las mismas restricciones de acceso. De otra manera, podría ser posible evadir las restricciones impuestas a un muro de seguridad entrando a la red de redes de la organización a través de otro.

---

[2] El término *firewall* (muro de seguridad) se deriva de la arquitectura, en la que un muro de seguridad es una separación gruesa, a prueba de fuego, por medio de la cual determinada sección de un edificio es impenetrable para el fuego.

Evadir un muro de seguridad pude ser el resultado de una acción maliciosa o inadvertida. Por supuesto, alguien que trata de lograr el acceso elige atacar en el punto más débil —si la organización tiene una conexión externa descuidada, un intruso encontrará fácil localizar y utilizar una conexión descuidada para alterar el mecanismo de seguridad en una conexión bien resguardada. De hecho, es bien conocida la idea de que un sistema de seguridad es tan fuerte como su parte más débil, y esto se conoce como el *axioma del eslabón más débil.*[3]

El axioma del eslabón más débil nos ayudará a explicar cómo la seguridad de una red de redes en una organización de redes puede comprometerse inadvertidamente. Consideremos una red de redes corporativa que conecta computadoras en todas las localidades de la corporación. Si el administrador de red en una oficina ramal proporciona una salida con acceso hacia una computadora en esta oficina, la computadora puede tener también software que permita accesar a otras computadoras en la organización. Incluso, si alguien, en el exterior, no trata de ser malicioso, puede solicitar u obtener información que debería estar restringida para los empleados. Así, un pequeño descuido en la configuración del muro de seguridad y la curiosidad de un usuario externo puede dejar vulnerable la entrada a la compañía.

Por desgracia, coordinar múltiples muros de seguridad puede ser difícil. Las restricciones necesarias en una localidad pueden no parecer importantes en otras. En consecuencia, los responsables de los muros de seguridad de la red de redes de una organización deben coordinar sus esfuerzos cuidadosamente. Podemos resumir la importancia de un perímetro de seguridad uniforme de la siguiente manera:

> *Una organización que tiene múltiples conexiones externas debe instalar un muro de seguridad en cada conexión externa y coordinar todos los muros de seguridad. Las fallas para restringir en forma idéntica el acceso en todos los muros de seguridad puede hacer que la organización sea vulnerable.*

28.8 Implantación de muro de seguridad y hardware de alta velocidad

¿Cómo debe implantarse un muro de seguridad? En teoría, un muro de seguridad sencillamente bloquea todas las comunicaciones no autorizadas entre computadoras en la organización y computadoras de organizaciones externas. En la práctica, los detalles dependen de las tecnologías de red, la capacidad de conexión, la carga de tráfico y las políticas de la organización. Así, no existe una solución que funcione para todas las organizaciones; construir un muro de seguridad a la medida y efectivo puede ser muy difícil.

Una de las dificultades en la construcción de un muro de seguridad consiste en el poder de procesamiento que se requiere. Un muro de seguridad necesita el suficiente poder computacional para examinar todos los mensajes que entran y salen. Para entender por qué el poder de procesamiento requerido puede ser significativo, pensemos en la conexión entre una corporación y la red global de Internet. Aun cuando una pequeña compañía puede utilizar un enlace lento hacia Internet, la conexión para una corporación mediana o grande debe operar a una alta velocidad a fin de

[3] El nombre proviene del dicho según el cual una cadena es tan fuerte como su eslabón más débil.

proporcionar un nivel adecuado de servicio. Dado que es necesario examinar cada datagrama que viaja entre la red interna y externa, el muro de seguridad de la organización debe manejar los datagramas a la misma velocidad que la conexión. Además, si un muro de seguridad retarda los datagramas en un búfer mientras decide si permite la transferencia, el muro de seguridad puede verse abrumado con las retransmisiones y el búfer se estancará.

Para operar a la velocidad de la red, un muro de seguridad debe tener hardware y software optimizado para la tarea. Por fortuna, la mayor parte de los ruteadores comerciales incluye un mecanismo de filtrado de alta velocidad que puede usarse para realizar muchas de las funciones necesarias. Un administrador puede configurar el filtro en un ruteador para solicitar que el ruteador bloquee datagramas específicos. Analizaremos el detalle de los mecanismos de filtrado y veremos cómo forman los filtros la estructura básica de bloqueo de un muro de seguridad. Más adelante veremos cómo pueden utilizarse todos los filtros junto con otros mecanismos para proporcionar comunicaciones que sean flexibles y seguras.

28.9 Filtros de nivel de paquete

Muchos ruteadores comerciales ofrecen un mecanismo que aumenta el ruteo normal y permite que el administrador tenga un mayor control en el procesamiento de paquetes. Informalmente conocidos *como filtros de paquetes* el mecanismo requiere que el administrador especifique cómo deberá manejar el ruteador cada datagrama. Por ejemplo, el administrador podría elegir para *filtrar* (es decir, para realizar un bloqueo selectivo) todos los datagramas que provengan de una fuente en particular o los utilizados para una aplicación particular mientras selecciona la ruta para otros datagramas hacia su destino.

El término *filtrado de paquetes* se debe a que el mecanismo de filtrado no conserva un registro de las interacciones o una historia de los datagramas previos. Por el contrario, el filtro considera a cada datagrama de manera separada. Cuando un datagrama llega primero, el ruteador pasa el datagrama a través de su filtrado de paquetes antes de realizar cualquier otro procesamiento. Si el filtro rechaza el datagrama, el ruteador lo desecha de inmediato.

Dado que el TCP/IP no dicta un estándar para el filtrado de paquetes. Cada vendedor de ruteadores es libre de seleccionar las características de su filtro de paquetes así como la interfaz que un administrador utiliza para configurar el filtro. Algunos ruteadores permiten al administrador configurar acciones de filtrado separadas para cada interfaz, mientras que otros tienen una sola configuración para todas las interfaces. Por lo general, cuando se especifican datagramas que el filtro debe bloquear, un administrador puede listar cualquier combinación de direcciones IP fuentes, direcciones IP destino, protocolos, número de puerto de protocolo fuente y número de puerto de protocolo destino. Por ejemplo, la figura 28.2 ilustra la especificación de un filtro.

En el ejemplo, el administrador ha elegido bloquear datagramas entrantes destinados a unos cuantos servicios bien conocidos y bloquear un caso de datagramas que salen. El filtro bloquea todos los datagramas que salen y que se originan desde cualquier anfitrión en la red de clase B. *128.5.0.0* está destinado a un servidor de correo electrónico remoto (TCP puerto *25*). El filtro también bloquea datagramas entrantes destinados a FTP (TCP puerto *21*), TELNET (TCP puerto *23*), WHOIS (UDP puerto *43*) TFTP (UDP puerto *69*) o FINGER (TCP puerto *79*).

LLEGADAS A LA INTERFAZ	FUENTE IP	DESTINO IP	PROTOCOLO	PUERTO FUENTE	PUERTO DESTINO
2	*	*	TCP	*	21
2	*	*	TCP	*	23
1	128.5.*.*	*	TCP	*	25
2	*	*	UDP	*	43
2	*	*	UDP	*	69
2	*	*	TCP	*	79

Figura 28.2 Ruteador con dos interfaces y un ejemplo de especificación de filtro de datagramas. Un ruteador que incluye filtro de paquetes forma la base de una estructura de bloqueo de un muro de seguridad.

28.10 Especificación de seguridad y de filtro de paquetes

Aun cuando en el ejemplo de configuración de filtrado en la figura 28.2 se especifica una pequeña lista de servicios que deben bloquearse, un método como éste no funciona bien en un muro de seguridad efectivo. Hay tres razones. En primer lugar, el número de puertos bien conocidos es extenso y creciente. Así, un administrador podría necesitar actualizar la lista continuamente debido a que un simple error podría hacer que el muro de seguridad fuera vulnerable. En segundo lugar, gran parte del tráfico en una red de redes no viaja hacia o desde un puerto bien conocido. Además, para los programadores que pueden elegir números de puerto desde sus aplicaciones de cliente-servidor privadas, los servicios como *Remote Procedure Call* (*RPC*) asignan puertos de manera dinámica. En tercer lugar, listar puertos de servicios bien conocidos hace que el muro de seguridad sea vulnerable al procedimiento mediante *túneles*, una técnica mediante la que se encapsula temporalmente a un datagrama en otro para transferirlo a través de una parte de la red de redes. El procedimiento mediante túneles se utiliza para la seguridad circundante, lo que hace que un anfitrión o ruteador en la entrada acepte datagramas encapsulados desde el exterior, retire una capa de encapsulación y envíe el datagrama hacia el servicio que de otra manera está restringido por el muro de seguridad.

¿Cómo puede un muro de seguridad utilizar con efectividad el filtrado de paquetes? La respuesta está en la idea contraria al filtrado: en lugar de especificar los datagramas que deben filtrarse, un muro de seguridad deberá configurarse para bloquear todos los datagramas excepto los destinados a redes específicas, anfitriones y puertos de protocolo para los que se ha aprobado la comunicación externa. Así, un administrador comienza por asumir que la comunicación no está permitida y que debe examinar las políticas de información de la organización cuidadosamente antes de habilitar cualquier puerto. De hecho, muchos filtros de paquetes permiten a un administrador especificar un conjunto de datagramas para admitirse en lugar de un conjunto de datagramas para bloquearse. Podemos resumir lo siguiente:

Para hacer efectivo un muro de seguridad que se vale del filtrado de datagramas debe restringirse el acceso de todas las fuentes IP, destinos IP, protocolos y puertos de protocolos a excepción de las computadoras, redes y servicios que la organización decida explícitamente poner a disposición del exterior. Un filtro de paquetes que permite a un administrador especificar qué datagramas admitir en lugar de qué datagramas bloquear, puede hacer que las restricciones sean más fáciles de especificar.

28.11 Consecuencia del acceso restringido para clientes

La prohibición impuesta a los datagramas que llegan desde puertos de protocolos desconocidos parece resolver muchos problemas potenciales de seguridad, evitando que desde el exterior se accese arbitrariamente a servidores en la organización. Así, un muro de seguridad tiene una consecuencia interesante: impide también que una computadora arbitraria dentro del muro de seguridad se vuelva un cliente que accese un servicio fuera de dicho muro. Para entender por qué, recordemos que, aun cuando cada servidor opera en un puerto bien conocido, un cliente no. Cuando un programa cliente comienza su ejecución, requiere que el sistema operativo seleccione un número de puerto de protocolo que no está entre los puertos bien conocidos ni se encuentra actualmente en uso en la computadora del cliente. Cuando trata de comunicarse con un servidor del exterior, un cliente generará uno o más datagramas y los enviará hacia el servidor. Cada datagrama que sale tiene el puerto de protocolo del cliente como puerto cliente y el puerto de protocolo bien conocido del servidor como puerto de destino. El muro de seguridad no bloqueará los datagramas cuando salgan. Cuando genere una respuesta, el servidor devolverá el puerto de protocolo. El puerto del cliente se vuelve el puerto de destino y el puerto del servidor se convierte en el puerto fuente. Cuando el datagrama que transporta la respuesta llega al muro de seguridad, será bloqueado ya que el puerto de destino no está aprobado. Así, podemos considerar la siguiente idea importante:

Si el muro de seguridad de una organización restringe los datagramas entrantes excepto para puertos que corresponden a servicios que la organización pone a disposición del exterior, una aplicación arbitraria al interior de la organización no podrá volverse cliente de un servidor exterior a la organización.

28.12 Acceso de servicios a través de un muro de seguridad

Por supuesto, no todas las organizaciones configuran sus muros de seguridad para bloquear todos los datagramas destinados a puertos de protocolo desconocidos (de hecho, no todas las organizaciones tienen un muro de seguridad para proteger sus redes). En los casos en que la seguridad de un muro de este tipo es necesaria para prevenir accesos inesperados, los usuarios en el interior necesitan un mecanismo seguro que proporcione acceso a servicios externos. Este mecanismo forma la segunda pieza mayor de la arquitectura de los muros de seguridad.

En general, una organización puede proporcionar sólo acceso seguro hacia servicios del exterior a través de una computadora segura. En lugar de tratar de hacer que todas las computadoras del sistema en la organización sean seguras (una tarea desalentadora), una organización por lo general asocia una computadora segura con cada muro de seguridad. Debido a que una computadora debe fortificarse poderosamente para servir como un canal de comunicación seguro, a menudo se le conoce como *anfitrión baluarte*. La figura 28.3 ilustra el concepto.

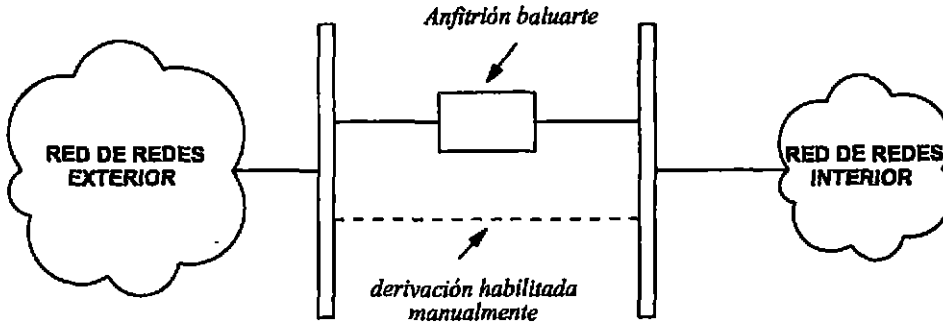


Figura 28.3 Organización conceptual de un anfitrión baluarte incorporado a un muro de seguridad. El anfitrión baluarte proporciona un acceso seguro a los servicios del exterior sin requerir que una organización admita datagramas con destinos arbitrarios.

Para permitir un acceso seguro, el muro de seguridad tiene dos barreras conceptuales. La barrera exterior bloquea todo el tráfico entrante excepto (1) a datagramas destinados a servicios en el anfitrión baluarte que la organización elige para mantener disponibles al exterior, y (2) a datagramas destinados a clientes en el anfitrión baluarte. La barrera de entrada bloquea el tráfico entrante excepto datagramas que se originan en el anfitrión baluarte. La mayor parte de los muros de seguridad también incluye una *derivación manual* que habilita al administrador para derivar temporalmente todo el tráfico, o parte de él, entre un anfitrión dentro de la organización y un anfitrión fuera de la organización (por ejemplo, para probar o depurar la red). En general, las organizaciones que desean una seguridad máxima, nunca habilitan una derivación.

Para entender cómo opera un anfitrión baluarte, consideremos el servicio FTP. Supongamos que un usuario en la organización necesita accesar un servidor externo FTP para obtener la copia de un archivo. Como el muro de seguridad impide que la computadora del usuario reciba datagramas entrantes, el usuario no puede correr el software cliente del FTP directamente. El usuario debe correr el cliente FTP en el anfitrión baluarte. Luego de que el archivo ha sido copiado por el anfitrión baluarte, el usuario puede correr una transferencia de archivo entre el anfitrión baluarte y su computadora local.

¿Cómo puede un usuario en una organización correr software cliente en un anfitrión baluarte? Las organizaciones por lo común siguen uno o dos esquemas básicos. Algunas organizaciones requieren que los usuarios utilicen un servicio de acceso remoto (por ejemplo, TELNET) para ponerse en contacto con el anfitrión baluarte. El usuario entonces invoca al software cliente desde el intérprete de comandos. Otras organizaciones proveen a los usuarios con programas cliente modifi-

cados que corren en sus computadoras pero que establecen contacto de manera automática con el anfitrión baluarte cuando es necesario. Las ventajas del primer método radican en su generalidad, economía y seguridad —sólo el anfitrión baluarte necesita la copia de un programa cliente para cada servicio de Internet y sólo esta copia se requiere para hacer el procedimiento seguro. Debido a que el acceso remoto proporciona acceso hacia todos los comandos en el anfitrión baluarte, un usuario en una computadora cualquiera en la organización sólo necesita software de acceso remoto para habilitar el acceso a cualquier servicio en Internet. La mayor desventaja de este primer método se debe a que pueden ser necesarios mecanismos adicionales para transferir la información desde el anfitrión baluarte hasta la computadora del usuario.

La ventaja del segundo método es su conveniencia —un usuario no necesita contar con un anfitrión baluarte, no necesita ponerse en línea antes de accesar a un servidor remoto y no tiene que aprender la sintaxis del intérprete de comandos en el anfitrión baluarte. La mayor desventaja en el segundo método se debe a los costos —cada computadora en la organización necesita un programa cliente especial para cada servicio, y el anfitrión baluarte requiere de software especial que pueda comunicarlo con los anfitriones en la organización así como con un servidor de Internet.

28.13 Detalles de la arquitectura del muro de seguridad

Ahora que entendemos el concepto básico de muro de seguridad, la implantación deberá parecer una consecuencia directa. Cada una de las barreras mostradas en la figura 28.3 requiere un ruteador que tenga un filtro de paquetes. Las redes interconectan los ruteadores y un anfitrión baluarte. Por ejemplo, una organización que se conecta con la red global de Internet por medio de una serie de líneas seriales puede elegir implantar un muro de seguridad como se muestra en la figura 28.4

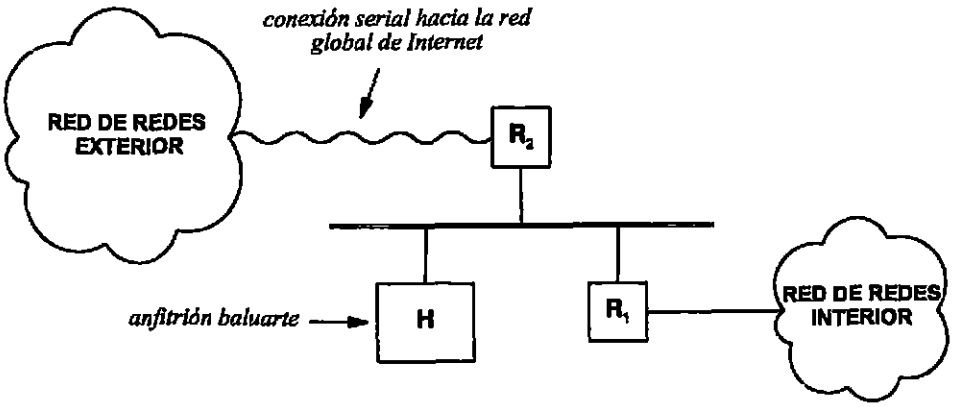


Figura 28.4 Implantación de un muro de seguridad con dos ruteadores y un anfitrión baluarte. Una línea serial conecta a la organización con el resto de Internet.

Como se muestra en la figura, el ruteador R_2 implementa la barrera externa; se filtra todo el tráfico excepto los datagramas destinados al anfitrión baluarte H. El ruteador R_1 implanta la barrera interior que aísla el resto de la red de redes corporativa del exterior; se bloquean todos los datos entrantes excepto los que se originan en el anfitrión baluarte.

Por supuesto, la seguridad de un muro de seguridad completo depende de la seguridad en el anfitrión baluarte. Si un intruso logra accesar al sistema de la computadora que corre en el anfitrión baluarte, logrará accesar hacia la red de redes entera. Más aún, un intruso puede abrir una grieta en la seguridad, ya sea que ésta se presente en el sistema operativo en el anfitrión baluarte o en las aplicaciones que la red está corriendo. Así, los administradores deben ser particularmente cuidadosos cuando eligen y configuran el software para un anfitrión baluarte. En resumen:

> *Aun cuando un anfitrión baluarte es esencial para la comunicación a través de un muro de seguridad, la seguridad de dicho muro depende de la seguridad en el anfitrión baluarte. Un intruso que abra una grieta en la seguridad en el sistema operativo del anfitrión baluarte puede lograr el acceso del anfitrión dentro del muro de seguridad.*

28.14 Red Stub

Podría parecer que la figura 28.4 contiene una red superflua que conecta los dos ruteadores y el anfitrión baluarte. Una red como ésta con frecuencia se conoce como *red stub* porque consta de un cable corto al que sólo se conectan tres computadoras. La cuestión es la siguiente, "¿la red stub es necesaria o la localidad puede colocar el anfitrión baluarte en una de sus redes de producción?" La respuesta depende del tráfico esperado desde el exterior. La red stub aísla a la organización del tráfico de datagramas entrantes. En particular, ya que el ruteador R_2 admite a todos los datagramas destinados al anfitrión baluarte, desde el exterior, se puede enviar un número arbitrario de datagramas a través de la red stub. Si una conexión externa es relativamente lenta para la capacidad de la red stub, un cable separado físicamente podría ser innecesario. Sin embargo, una red stub es por lo general una forma económica para que una organización se proteja contra la interrupción del servicio en una red interna de producción.

28.15 Implantación alternativa de muro de seguridad

La implantación de muro de seguridad en la figura 28.4 funciona bien para una organización que tiene una sola conexión serial con el resto de la red global de Internet. Algunas localidades tienen una topología de interconexiones diferente. Por ejemplo, supongamos que una compañía tiene tres o cuatro clientes grandes, cada uno de los cuales necesita extraer o almacenar grandes volúmenes de información. La compañía desea tener un solo muro de seguridad, pero también desea permitir la conexión con varias localidades.[4] La figura 28.5 ilustra una posible arquitectura de muro de seguridad que se adapta a múltiples conexiones externas.

[4] Un solo muro de seguridad puede ser menos caro y más fácil de administrar que un muro de seguridad por cada conexión, separados entre sí..

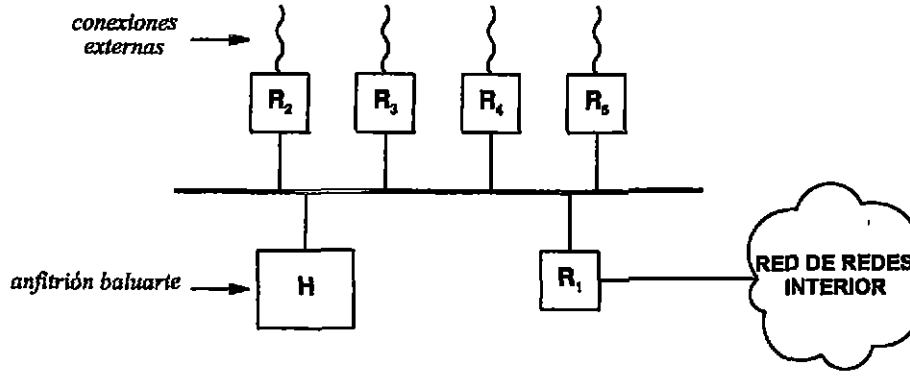


Figura 28.5 Arquitectura alternativa de muro de seguridad que permite varias conexiones externas a través de un solo muro de seguridad. Utilizar un solo muro de seguridad para varias conexiones puede reducir los costos.

Como se muestra en la figura, la arquitectura alternativa extiende un muro de seguridad al proporcionar una salida a la red en la que se ubican las terminales de las conexiones externas. Como se muestra en la figura 28.4, el ruteador R_1 actúa protegiendo a la localidad mediante la restricción de los datagramas entrantes que son enviados desde el anfitrión baluarte. Los ruteadores R_2 a R_5 conectan a una localidad externa con el muro de seguridad.

Para entender por qué los muros de seguridad con conexiones múltiples frecuentemente utilizan un ruteador por conexión, recordemos que todas las localidades desconfían unas de otras. Es decir que la organización que corre el muro de seguridad no confía de cualquier otra organización externa del todo y ninguna de las organizaciones externas confía por completo en otra. El filtro de paquetes en un ruteador en una conexión externa dada puede configurarse para restringir el tráfico en una conexión particular. Como resultado, los propietarios de un muro de seguridad pueden garantizar que, aun cuando todas las conexiones externas compartan una sola red común, ningún datagrama que provenga de una conexión externa pasará hacia otra. Así, la organización que corre el muro de seguridad puede asegurar a sus clientes que su conexión es segura.

En resumen:

Cuando varias localidades externas se conectan a través de un solo muro de seguridad, una arquitectura que cuenta con un ruteador por conexión externa, puede prevenir el flujo indeseable de paquetes desde una localidad externa hacia otra.

28.16 Monitoreo y establecimiento de conexión

El monitoreo es uno de los aspectos más importantes en el diseño de un muro de seguridad. El administrador de red responsable de un muro de seguridad necesita estar consciente de los riesgos en

la seguridad. A menos que se cuente con un reporte de incidentes en un muro de seguridad, el administrador podría no darse cuenta de los problemas que se presentan.

El monitoreo puede ser *activo* o *pasivo*. En el monitoreo activo, un muro de seguridad notifica al administrador todos los incidentes que se presentan. La mayor ventaja del monitoreo activo es la velocidad —un administrador puede tener conocimiento de problemas potenciales de inmediato. La mayor desventaja es que en el monitoreo activo frecuentemente se produce mucha información que un administrador puede no comprender o puede no encontrar en tal información ningún indicio de problemas. Así, la mayoría de los administradores prefiere el monitoreo pasivo o una combinación de monitoreo pasivo con el reporte de unos cuantos incidentes de alto riesgo en lugar del monitoreo activo.

En el monitoreo pasivo, un muro de seguridad lleva un registro de cada incidente en un archivo en disco. Un monitoreo pasivo por lo general registra la información del tráfico normal (es decir, como una simple estadística) y los datagramas que se filtran. Un administrador puede ingresar a la bitácora en cualquier momento; la mayoría de los administradores utiliza un programa de computadora. La mayor ventaja del monitoreo pasivo radica en que elabora registros de eventos —un administrador puede consultar la bitácora para observar las tendencias y, cuando se presente un problema de seguridad, revisar la historia de eventos que conducen a un problema dado. Algo muy importante, un administrador puede analizar la bitácora periódicamente (por ejemplo, cada día) para determinar si los intentos por acceder a la organización se han incrementado o han decrecido con el tiempo.

28.17 Resumen

Los problemas de seguridad se originan porque una red de redes puede conectar organizaciones que no tienen una confianza recíproca. Se dispone de varias técnicas para ayudar a asegurar que la información se mantenga a salvo cuando se envía a través de una red de redes. Un cliente y un servidor pueden utilizar el cifrado para garantizar su identidad; los servidores necesitan de la autenticación para determinar si un cliente está autorizado para accesar un servicio. El cifrado también puede resolver el problema de la privacía pues hace que un emisor y un receptor se puedan comunicar a través de un canal poco seguro sin temor a que la comunicación esté siendo intervenida. Antes de que una organización pueda elegir un mecanismo para reforzar la seguridad, es necesario establecer una política de información.

El mecanismo de muro de seguridad se utiliza para controlar el acceso a la red de redes. Una organización coloca un muro de seguridad en cada conexión externa para garantizar que la red de redes interna de la organización se mantenga libre de tráfico no autorizado. Un muro de seguridad consiste en dos barreras y una computadora segura, llamada anfitrión baluarte. Cada barrera utiliza un filtro para restringir el tráfico de datagramas. El anfitrión baluarte ofrece servicios visibles desde el exterior y corre clientes que accesan servidores externos. La organización utiliza su información y sus políticas de acceso a la red de redes para determinar cómo configurar el filtro. Por lo general, un muro de seguridad bloquea todos los datagramas que llegan desde el exterior, excepto los destinados al anfitrión baluarte.

Un muro de seguridad puede implantarse en una de varias formas; la elección depende de los detalles así como del número de conexiones externas. En muchos casos, cada barrera en un muro

de seguridad se implanta con un ruteador que contiene un filtro de paquetes. Un muro de seguridad puede utilizar también una red stub para mantener fuera el tráfico externo de la red de producción de una organización.

PARA CONOCER MÁS

Muchos RFC se enfocan a aspectos relacionados con la seguridad en las redes de redes y proponen políticas, procedimientos y mecanismos. Crocker, Fraser y Pethia (RFC 1281) analizan la operación segura de Internet, en tanto que Holbrook y Reynolds (RFC 1244), la seguridad de las localidades. Galvin y McCloghrie (RFC 1446) presentan las aportaciones de SNMPv2 a la seguridad. Cheswick y Bellovin (1994) tratan los muros de seguridad y otros temas relacionados con la operación segura del TCP/IP en las redes de redes.

Muchas veces se han considerado protocolos que garanticen la privacía de los mensajes de correo electrónico. Cuatro RFC presentados por Linn (RFC 1421), Kent (RFC 1422), Balenson (RFC 1423) y Kaliski (RFC 1424) presentan un protocolo para *Privacy Enchanced Mail* (*PEM*). Kohl y Neuman (RFC 1510) describen el servicio de autenticación *kerberos* y Borman (RFC 1411) analiza cómo puede usarse *kerberos* para autenticar a TELNET.

El proyecto de seguridad COAST de la Universidad de Purdue ha reunido un archivo extenso de información relacionada con la seguridad de computadoras y redes. El archivo puede accesarse por medio de gopher, ftp o World Wide Web:

gopher://coast.cs.purdue.edu
ftp://coast.cs.purdue.edu/pub
http://www.cs.purdue.edu/coast/coast.html

EJERCICIOS

28.1 Muchas organizaciones que requieren que todas las transferencias de archivos se hagan a través de un anfitrión baluarte hacen que un software de transferencia de archivos los revise antes de admitirlos en la organización. ¿Por qué la organización revisa los archivos? Sugerencia: piense en los programas importantes que corren en una computadora personal.

28.2 Lea la descripción de un filtrador de paquetes, de un ruteador disponible comercialmente. ¿Qué características ofrecen?

28.3 Obtenga una bitácora del tráfico que entra a su localidad que tenga una entrada por datagrama. Analice la bitácora para determinar el porcentaje de tráfico que llega desde o es destinada hacia un puerto de protocolo bien conocido. ¿Le sorprenden los resultados?

28.4 Si está disponible en su computadora un software de cifrado, mida el tiempo necesario para cifrar un archivo de 10 Mbyte, transferirlo a otra computadora y descifrarlo. Compárelo con el tiempo requerido para transferirlo si no se usa cifrado.

28.5 Infórmese si los usuarios de su localidad envían información confidencial por e-mail. ¿Los usuarios entienden que el SMTP transfiere mensajes en ASCII y que nadie que observe el tráfico de red puede ver el contenido de un mensaje de e-mail?

28.6 Investigue entre los empleados de su localidad qué tanto utilizan los módems y las computadoras personales para importar y exportar información. Averigüe si comprenden las políticas de la organización.

28.7 Puede utilizarse un muro de seguridad con otros conjuntos de protocolos como Appletalk o Netware? ¿Por qué sí o por qué no?

28.8 Los militares sólo liberan información para quienes "necesitan saber". ¿Tal esquema debe aplicarse a toda la información de su organización? ¿Por qué sí o por qué no?

28.9 Dé dos razones por las que un grupo de personas que administra las políticas de seguridad de una organización deben separarse de las personas que administran los sistemas de red y las computadoras de una organización.

28.10 Algunas organizaciones utilizan muros de seguridad para aislar grupos de usuarios internamente. Proporcione ejemplos de la forma en que los muros de seguridad pueden mejorar el desempeño de la red interna y de cómo pueden degradar el desempeño de la red.

29

El futuro de TCP/IP (IPng, IPv6)

29.1 Introducción

La evolución de la tecnología TCP/IP está vinculada a la evolución de Internet por varias razones. En primer lugar, Internet es la red de redes del TCP/IP instalada más extensa, de manera que muchos problemas aparecen en Internet antes de que salgan a la superficie en otras redes de redes TCP/IP. En segundo lugar, los investigadores e ingenieros fundadores del TCP/IP provienen de compañías y dependencias gubernamentales que utilizan Internet, de manera que tienden a fundar proyectos que impactan a Internet. En tercer lugar, la mayoría de los investigadores participantes en el TCP/IP tienen conexiones con Internet y la utilizan diariamente. Así pues, tienen una motivación inmediata para resolver problemas que mejorarán el servicio y ampliarán su funcionalidad.

Con millones de usuarios en decenas de miles de localidades alrededor del mundo que dependen de la red global de Internet como parte de su ambiente diario de trabajo, puede parecer que Internet es una infraestructura de producción estable. Hemos pasado de las primeras etapas de desarrollo, en las que los usuarios eran también expertos, a una etapa en la cual pocos usuarios comprenden la tecnología. Sin embargo, a pesar de las apariencias, ni Internet ni el conjunto de protocolos TCP/IP son estáticos. Nuevos grupos conectan sus redes y descubren nuevas formas de utilizar la tecnología. Los investigadores resuelven nuevos problemas de redes y los ingenieros mejoran los mecanismos subyacentes. En pocas palabras, la tecnología continúa evolucionando.

El propósito de este capítulo es considerar el proceso de evolución actual y examinar uno de los más importantes esfuerzos de ingeniería. En particular, veremos una propuesta de revisión del IP. Si la propuesta es aprobada como estándar y adoptada por los vendedores, tendrá un mayor impacto en el TCP/IP y en Internet. Si el nuevo protocolo se hace parte del TCP/IP en los próximos meses, años o décadas es irrelevante; el objetivo es hacer que el lector comprenda el esfuerzo. El

lector deberá estar consciente de que la propuesta no es un estándar final y que los detalles pueden cambiar.

29.2 ¿Por qué cambiar TCP/IP e Internet?

La tecnología básica TCP/IP ha funcionado bien por una década. ¿Por qué debería cambiarse? En términos generales, los procesos que estimulan la evolución del TCP/IP y de la arquitectura de Internet se pueden clasificar dentro de cuatro categorías. Luego de describir cada categoría, examinaremos la propuesta de una nueva versión del IP y veremos cómo cada categoría afecta al diseño.

29.2.1 Nuevas tecnologías de comunicación y computación

Como en la mayoría de los grupos orientados hacia la tecnología, los investigadores e ingenieros que trabajan en los protocolos TCP/IP mantienen un agudo interés por las nuevas tecnologías. Tan pronto como un nueva computadora de alta velocidad está disponible, la utilizan en anfitriones y ruteadores. En cuanto una nueva tecnología de red emerge, la utilizan para transportar datagramas IP. Por ejemplo, además de las LAN y las líneas convencionales de comunicación serial arrendadas, los investigadores del TCP/IP han estudiado la comunicación punto a punto vía satélite, las estaciones múltiples de satélites sincronizados, los paquetes de radio y ATM. Más recientemente, los investigadores han estudiado las redes inalámbricas que se valen de luz infrarroja o las tecnologías de frecuencias de radio de espectro extendido.

29.2.2 Nuevas aplicaciones

Las nuevas aplicaciones constituyen una de las fronteras de investigación y desarrollo de Internet más interesantes y por lo general crean una demanda de infraestructura o servicios que los protocolos actuales no pueden proporcionar. Por ejemplo, el interés creciente en multimedios ha creado una demanda de protocolos que puedan transferir imágenes y sonido eficientemente. De la misma forma, el interés en la comunicación en tiempo real de audio y video ha creado una demanda de protocolos que puedan garantizar la entrega de la información con retardos fijos, así como protocolos que puedan sincronizar audio y video con flujos de datos.

29.2.3 Incrementos en el tamaño y en la carga

La red global de Internet ha tenido varios años de crecimiento exponencial, duplicando su tamaño cada nueve meses o más rápido. A principios de 1994, en promedio, un nuevo anfitrión aparecía cada 30 segundos, y la cantidad se incrementó de manera dramática. Sorpresivamente, la carga de tráfico en Internet ha crecido más rápido que el número de redes. El incremento en el tráfico puede atribuirse a varias causas. En primer lugar, la población de Internet está cambiando su composición respecto al público en general, deja de estar formada por académicos e investigadores. En conse-

cuencia, la gente ahora utiliza Internet luego de sus horas de trabajo para actividades comerciales y de entretenimiento. En segundo lugar, las nuevas aplicaciones que transfieren imágenes y video en tiempo real generan más tráfico que las aplicaciones que transfieren texto. En tercer lugar, las herramientas de búsqueda automatizada generan una cantidad sustancial de tráfico y lo hacen más lento al sondear en las localidades de Internet para encontrar datos.

29.2.4 Nuevas políticas

Conforme se expande hacia nuevas industrias y nuevos países, Internet cambia de forma fundamental: adquiere nuevas autoridades administrativas. Los cambios en la autoridad producen cambios en las políticas administrativas y se establecen nuevos mecanismos para reforzar tales políticas. Como hemos visto, la arquitectura de conexión de Internet y los protocolos que utiliza comprenden un modelo de núcleo centralizado. La evolución continúa conforme se conectan más columnas vertebrales de redes nacionales, produciendo un incremento complejo de políticas que regulan la interacción. Cuando diversas corporaciones interconectan redes TCP/IP privadas enfrentan problemas similares al tratar de definir políticas de interacción y encontrar mecanismos para reforzar estas políticas. Así, muchos de los esfuerzos de investigadores e ingenieros alrededor del TCP/IP continúan enfocados a encontrar formas de adaptarse a nuevos grupos administrativos.

29.3 Motivos para el cambio del IPv4

La versión *4* del protocolo de Internet (*IPv4*) proporciona los mecanismos de comunicación básicos del conjunto TCP/IP y la red global Internet; se ha mantenido casi sin cambio desde su inserción a fines de los años setenta.[1] La antigüedad de la versión *4* muestra que el diseño es flexible y poderoso. Desde el momento en que se diseñó el IPv4, el desempeño de los procesadores se ha incrementado en dos órdenes de magnitud, el tamaño de las memorias se ha incrementado por un factor de 32, el ancho de banda de la columna vertebral de la red Internet se ha incrementado en un factor de 800, las tecnologías LAN han emergido y el número de anfitriones en Internet ha crecido hasta llegar un total de 4 millones. Además, los cambios no ocurren de manera simultánea —el IP se ha adaptado a los cambios de una tecnología antes de adaptarse a los cambios de otras.

A pesar de su diseño, el IPv4 también debe ser reemplazado. En el capítulo 10, se describe las principales motivaciones para actualizar el IP: el inminente agotamiento del espacio de direcciones. Cuando el IP se diseñó, un espacio de 32 bits era más que suficiente. Sólo un puñado de organizaciones utilizaba las LAN; pocas tenían una WAN corporativa. Ahora, sin embargo, muchas corporaciones de tamaño mediano tienen varias LAN y varias de las grandes corporaciones cuentan con una WAN corporativa. En consecuencia, el espacio de direcciones IP de 32 bits que se usa actualmente no puede adaptarse al crecimiento proyectado de la red global de Internet.

Aun cuando la necesidad de un espacio de direcciones extenso está forzando un cambio inmediato en el IP, hay otros factores que también contribuyen. En particular, gran parte de éstos se refieren al soporte de nuevas aplicaciones. Por ejemplo, debido a que el audio y el video en tiempo

---

[1] Las versiones de la *1* a la *3* nunca se asignaron formalmente y la versión número *5* fue asignada al protocolo *ST*.

real necesitan determinadas garantías en los retardos, una nueva versión del IP debe proporcionar un mecanismo que haga posible asociar un datagrama con una reservación de fuente preasignada. Además, como varias de las nuevas aplicaciones de Internet necesitan comunicaciones seguras, una nueva versión del IP deberá incluir capacidades que hagan posible autenticar al emisor.

29.4 El camino hacia una nueva versión del IP

Los grupos en el IETF han estado trabajando para formular una nueva versión del IP por varios años. Como tratan de producir un estándar *abierto*, el IETF ha invitado a toda la comunidad a participar en el proceso de estandarización. En consecuencia, investigadores, fabricantes de computadoras, vendedores de hardware y software de red, programadores, administradores, usuarios, compañías telefónicas y televisoras por cable han especificado sus requerimientos para la próxima versión IP y han comentado todos sus propuestas específicas.

Se han propuesto muchos diseños para servir a un propósito en particular o a una comunidad en especial. Uno de los diseños propuestos haría al IP más sofisticado y el costo por el incremento en la complejidad de procesamiento se elevaría. Otro diseño propone utilizar una modificación del protocolo CLNS de OSI. Un tercer diseño mayor propone conservar la mayor parte de las ideas del IP, y hacer extensiones para adaptarlo a direcciones extensas. El diseño conocido como *SIP (Simple IP)* ha sido la base para una propuesta extendida que incluye ideas de otras propuestas. La versión extendida del SIP ha sido llamada *Simple IP Plus* (*SIPP*) y finalmente emerge como el diseño elegido como base para el próxima IP.

Seleccionar una nueva versión del IP no ha sido fácil. La popularidad de Internet hace que el mercado de productos IP alrededor del mundo se tambalee. Muchos grupos consideran esto como una oportunidad económica y tratan de que la nueva versión del IP les ayude a obtener ganancias sobre sus competidores. Además, se han involucrado algunas personalidades —algunas opiniones técnicas individuales se mantienen fuertemente; otros consideran la participación activa como una manera de hacerse promoción. En consecuencia, las discusiones han generado argumentaciones acaloradas.

29.5 Nombre del próximo IP

Al comienzo de las discusiones sobre el cambio del IP, el IAB publicó una declaración política que se refería a la próxima versión como *IP version 7*, el informe causó una confusión general. La gente preguntaba "¿qué sucedió con la versión *5* y *6*?" ¿El IAB se refiere a la *versión 5*, o se refiere al establecimiento de una política para un futuro a largo plazo. Evidentemente, el error ocurrió porque el protocolo ST estaba asignado como la versión número *5* y uno de los documentos disponibles por el IAB reportaba erróneamente a la versión actual como la versión *6*.

Para evitar la confusión, el IETF cambió el nombre. Retomando el nombre de una popular serie de televisión, el IETF eligió "IP — la próxima generación" y el esfuerzo comenzó a conocerse como *IPng*.

Formalmente, se ha decidido que a la próxima versión del IP se le asigne el número de versión *6*. Así, para distinguirlo de la versión actual del IP (*IPv4*), la próxima generación se llamará *IPv6*. En el pasado, el término *IPng* ha sido utilizado en un contexto amplio para referirse a todas las discusiones y propuestas para una próxima versión del IP, mientras que el término *IPv6* [2] se ha utilizado para referirse a una propuesta específica que proviene del IETF. La literatura actual a menudo trata a los dos términos como sinónimos y los emplea de manera indistinta. Para ayudar a distinguir la discusión general respecto de la propuesta actual, utilizaremos el término *IPv6* para referirnos al protocolo específico que ha sido propuesto e *IPng* para referirnos a todos los esfuerzos relacionados con el desarrollo de una nueva generación del IP.

29.6 Características del IPv6

El protocolo IPv6 propuesto conserva muchas de las características que contribuyeron al éxito del IPv4, de hecho, los diseñadores han caracterizado al IPv6 como si fuera básicamente el mismo que el IPv4 con unas cuantas modificaciones. Por ejemplo, el IPv6 todavía soporta la entrega sin conexión (es decir, permite que cada datagrama sea ruteado independientemente), permite al emisor seleccionar el tamaño de un datagrama y requiere que el emisor especifique el máximo número de saltos que un datagrama puede realizar antes de ser eliminado. Como veremos, el IPv6 también conserva la mayor parte de los conceptos proporcionados por la versión IPv4, incluyendo capacidades de fragmentación y ruteo de fuente.

A pesar de las similitudes conceptuales, el IPv6 cambia la mayor parte de los detalles de protocolo. Por ejemplo, el IPv6 utiliza direcciones largas y añade unas cuantas características nuevas. Algo muy importante, el IPv6 revisa completamente el formato de los datagramas, remplazando el campo de opción de longitud variable del IPv4 por una serie de encabezados de formato fijo. Examinaremos los detalles luego de considerar los cambios mayores y las motivaciones subyacentes para cada uno.

Los cambios introducidos para el IPv6 pueden agruparse en cinco categorías:

- *Direcciones más largas.* El nuevo tamaño de las direcciones es el cambio más notable. El IPv6 cuadruplica el tamaño de las direcciones del IPv4, va de 32 bits a 128 bits. El espacio de direcciones del IPv6 es tan grande que no podrá agotarse en un futuro previsible.
- *Formato de encabezados flexible.* El IPv6 utiliza un formato de datagrama incompatible y completamente nuevo. A diferencia del IPv4, que utiliza un encabezado de datagrama de formato fijo en el que todos los campos excepto las opciones ocupan un número fijo de octetos en un desplazamiento fijo, el IPv6 utiliza un conjunto de encabezados opcionales.
- *Opciones mejoradas.* Como el IPv4, el IPv6 permite que un datagrama incluya información de control opcional. El IPv6 incluye nuevas opciones que proporcionan capacidades adicionales no disponibles en el IPv4.

---

[2] Algunos autores utilizan la abreviatura *IP6*

- *Soporte para asignación de recursos.* El IPv6 reemplaza la especificación del tipo de servicio del IPv4 con un mecanismo que permite la preasignación de recursos de red. En particular, el nuevo mecanismo soporta aplicaciones como video en tiempo real que requieren de una garantía de ancho de banda y retardo.
- *Provisión para extensión de protocolo.* Posiblemente el cambio más significativo en el IPv6 es el cambio de un protocolo que especifica completamente todos los detalles a un protocolo que puede permitir características adicionales. La capacidad de extensión tiene la posibilidad de permitir que el IETF se adapte a los protocolos para cambiar al hardware de red subyacente o a nuevas aplicaciones.

29.7 Forma general de un datagrama IPv6

El IPv6 cambia completamente el formato de datagrama. Como se muestra en la figura 29.1, un datagrama IPv6 tiene un *encabezado base* de tamaño fijo, seguido por ceros o más *encabezados de extensión*, seguidos a su vez por datos.

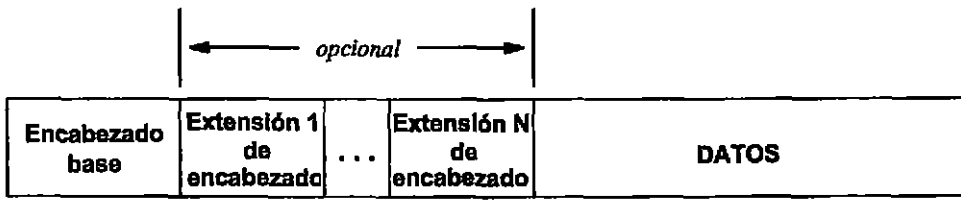


Figura 29.1 Forma general de un datagrama IPv6 con varios encabezados. Sólo el encabezado base es indispensable, los encabezados de extensión son opcionales.

29.8 Formato del encabezado base del IPv6

Es interesante que, aun cuando debe adaptarse a direcciones extensas, un encabezado base IPv6 contiene menos información que un encabezado de datagrama IPv4. Las opciones y algunos de los campos fijos que aparecen en un encabezado de datagrama del IPv4 se han cambiado por encabezados de extensión en el IPv6. En general, el cambio en los encabezados en los datagramas refleja los cambios en el protocolo:

- La alineación se ha cambiado de múltiplos de 32 bits a múltiplos de 64 bits.
- Los campos de longitud de encabezado se han eliminado y el campo de longitud de datagrama ha sido reemplazado por el campo *PAYLOAD LENGTH* (*LONGITUD PAYLOAD*).

- El tamaño de los campos de dirección de fuente y destino se ha incrementado en 16 octetos cada uno.
- La información de fragmentación se ha movido de los campos fijos en el encabezado base, hacia un encabezado de extensión.
- El campo *TIME-TO-LIVE* (*LÍMITE DE SALTO*) ha sido reemplazado por el *HOP LIMIT*.
- El campo *SERVICE TYPE* ha sido reemplazado por el campo *FLOW LABEL* (*ETIQUETA DE FLUJO*).
- El campo *PROTOCOL* ha sido reemplazado por un campo que especifica el tipo del próximo encabezado.

La figura 29.2 muestra el contenido y el formato de un encabezado base IPv6. Varios campos en un encabezado base IPv6 corresponden directamente a los campos en un encabezado IPv4. Como en el IPv4, el campo inicial *VERS* de 4 bits especifica la versión del protocolo; *VERS* siempre contiene el número 6 en un datagrama IPv6. Como en el IPv4, los campos *SOURCE ADDRESS* (*DIRECCIÓN FUENTE*) y *DESTINATION ADDRESS* (*DIRECCIÓN DE DESTINO*) especifican la dirección del emisor y del recipiente. En el IPv6, sin embargo, cada dirección requiere 16 octetos. El campo *HOP LIMIT* corresponde al campo *TIME- TO-LIVE* del IPv4. A diferencia del IPv4, que interpreta un tiempo límite como una combinación de conteo de saltos y tiempo máximo, el IPv6 interpreta el valor como un límite estricto del máximo número de saltos que un datagrama puede realizar antes de ser desechado.

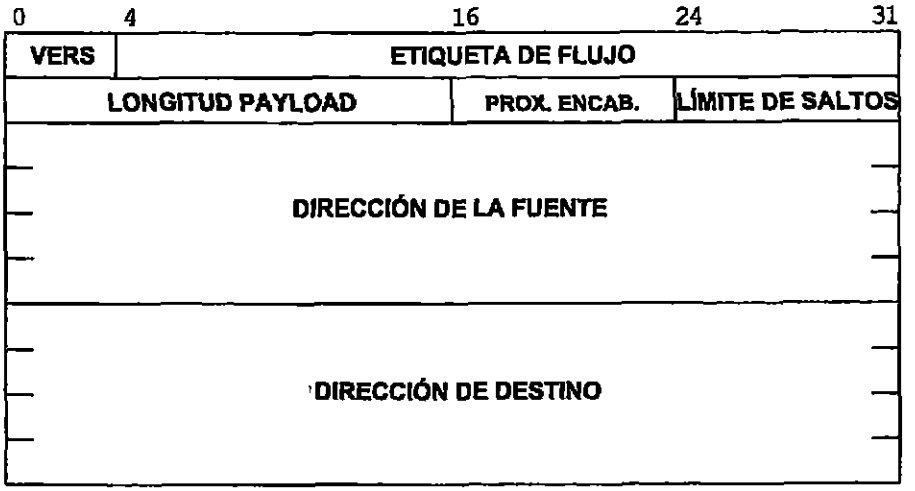


Figura 29.2 Formato del encabezado base de 40 octetos del IPv6. Cada datagrama IPv6 comienza con un encabezado base.

El IPv6 maneja las especificaciones de longitud de datagramas de forma nueva. En primer lugar, debido a que el tamaño del encabezado base se fijó en 40 octetos, dicho encabezado no incluye un campo para la longitud de encabezado. En segundo lugar, el IPv6 reemplaza el campo de longitud de datagrama del IPv4 por un campo *PAY LOAD LENGTH* de 16 bits que especifica el número de octetos transportados en un datagrama, excluyendo al encabezado mismo. Así, un datagrama IPv6 puede contener 64K octetos de datos.

Un nuevo mecanismo en el IPv6 soporta reservación de recursos y permite a un ruteador asociar cada datagrama con una asignación de recursos dados. La abstracción subyacente, un *flujo*, consiste en una trayectoria a través de una red de redes a lo largo de la cual ruteadores intermedios garantizan una calidad de servicio específica. Por ejemplo, dos aplicaciones que necesitan enviar video pueden establecer un flujo en el que el retardo y el ancho de banda estén garantizados. Como alternativa, un proveedor de red puede requerir una suscripción para especificar la calidad de servicio deseado y, luego, utilizar un flujo para limitar el tráfico a una computadora específica o al envío de una aplicación específica. Observe que un flujo puede también utilizarse dentro de una organización determinada para administrar recursos de red y asegurar que todas las aplicaciones puedan compartir recursos de manera justa.

El campo *FLOW LABEL* en el encabezado base contiene información que los ruteadores utilizan para asociar un datagrama con una prioridad y un flujo específicos. El campo está subdividido en dos subcampos como se muestra en la figura 29.3.

4 bits	24 bits
TCLASS	IDENTIFICADOR DE FLUJO

Figura 29.3 Los dos subcampos de una etiqueta de flujo. Cada datagrama IPv6 transporta una etiqueta de flujo, la cual puede usarse para asociar el datagrama con una calidad específica de servicio.

Dentro de la etiqueta de flujo, el campo *TCLASS* de 4 bits especifica la clase de tráfico para el datagrama. Los valores del *0* al *7* se emplean para especificar la sensibilidad al tiempo del tráfico controlado por flujo; los valores del *8* al *15* se utilizan para especificar una prioridad para tráfico que no es de flujo. El campo de 24 bits restantes contiene el campo *FLOW IDENTIFIER* (*IDENTIFICADOR DE FLUJO*). La fuente selecciona un identificador de flujo cuando establece el flujo (por ejemplo, en forma aleatoria). No hay conflicto potencial entre las computadoras debido a que un ruteador utilice la combinación de direcciones fuente de datagramas e identificadores de flujo cuando asocia un datagrama con un flujo específico. En resumen:

Cada datagrama IPv6 comienza con un encabezado base de 40 octetos que incluye campos para las direcciones de fuente y destino, el límite máximo de saltos, la etiqueta de flujo y el tipo del próximo encabezado. Así, un datagrama IPv6 debe contener cuando menos 40 octetos además de los datos.

29.9 Encabezados de extensión del IPv6

El paradigma de un encabezado base fijo seguido por un conjunto de encabezados de extensión opcionales se eligió como un compromiso entre la generalidad y la eficiencia. Para ser totalmente general, el IPv6 necesita incluir mecanismos para soportar funciones como la fragmentación, el ruteo de fuente y la autenticación. Sin embargo, elegir la asignación de campos fijos en el encabezado de datagrama para todos los mecanismos es ineficiente pues la mayor parte de los datagramas no utiliza todos los mecanismos. El gran tamaño de las direcciones IPv6 aumenta la ineficiencia. Por ejemplo, cuando se envía un datagrama a través de una red de área local, un encabezado que contenga campos de dirección vacíos puede ocupar una fracción sustancial de cada trama. Algo muy importante, los diseñadores asumen que no se puede predecir qué recursos serán necesarios.

El paradigma de encabezado de extensión IPv6 funciona en forma similar a las opciones del IPv4 —un emisor puede elegir qué encabezados de extensión incluir en un datagrama determinado y cuáles omitir. Así, los encabezados de extensión proporcionan una flexibilidad máxima. Podemos resumir lo siguiente:

Los encabezados de extensión IPv6 son similares a las opciones IPv4. Cada datagrama incluye encabezados de extensión sólo para los recursos que el datagrama utilice.

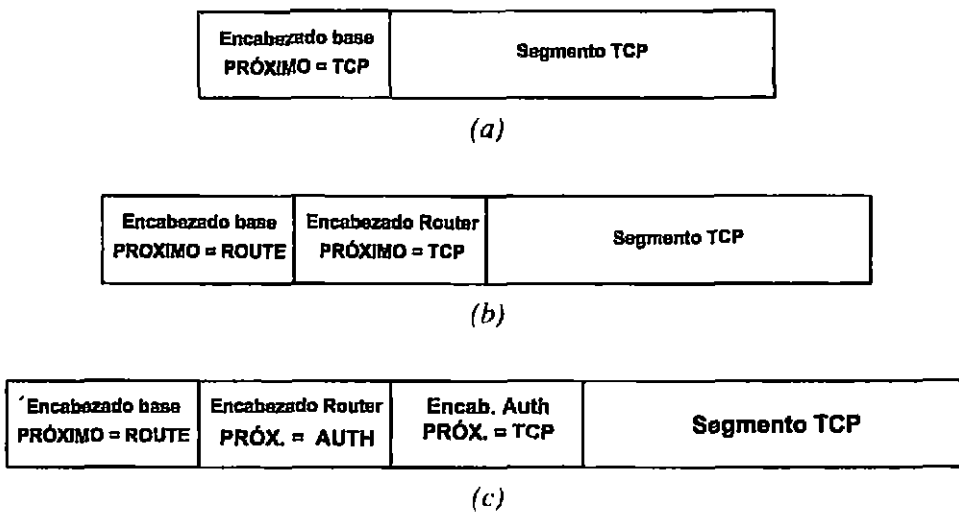


Figura 29.4 Tres datagramas con (a) sólo un encabezado base, (b) un encabezado base y una extensión y (c) un encabezado base más dos extensiones. El campo *NEXT HEADER* (*PRÓXIMO ENCABEZADO*) en cada encabezado especifica el tipo de encabezado siguiente.

29.10 Análisis de un datagrama IPv6

Cada encabezado de base y extensiones contiene un campo *NEXT HEADER* (*PRÓXIMO ENCABEZADO*). El software en los ruteadores intermedios y en el destino final que necesitan procesar el datagrama deben utilizar el valor en el campo *NEXT HEADER* de cada encabezado para analizar el datagrama. Extraer toda la información del encabezado de un datagrama IPv6 requiere de una búsqueda secuencial a través de los encabezados. Por ejemplo, la figura 29.4 muestra el campo *NEXT HEADER* de 3 datagramas que contienen cero, uno y dos encabezados de extensión.

Por supuesto, analizar un datagrama IPv6 que sólo tiene un encabezado base y datos es tan eficaz como analizar un datagrama IPv4. Además, veremos que los ruteadores intermedios con frecuencia necesitan procesar todos los encabezados de extensión.

29.11 Fragmentación y reensamblaje del IPv6

Como el IPv4, el IPv6 prepara el destino final para realizar el reensamblaje de datagramas. Sin embargo, los diseñadores tomaron una decisión poco usual respecto a la fragmentación. Recordemos que el IPv4 requiere un ruteador intermedio para fragmentar cualquier datagrama que sea demasiado largo para la MTU de la red en la que viaja. En el IPv6, la fragmentación está restringida a la fuente original. Antes de enviar tráfico de información, una fuente debe realizar una técnica de *Path MTU Discovery* (*descubrir la MTU de la ruta*) para identificar la MTU (Maximum Transfer Unit) mínima a lo largo de la trayectoria hasta el destino. Antes de enviar un datagrama, la fuente fragmenta el datagrama de manera que cada fragmento sea menor que el Path MTU. Así, la fragmentación es de extremo a extremo; no son necesarias fragmentaciones adicionales en ruteadores intermedios.

El encabezado base IPv6 no contiene campos análogos a los campos utilizados para la fragmentación en un encabezado IPv4. Por el contrario, cuando la fragmentación es necesaria, la fuente inserta un pequeño encabezado de extensión luego del encabezado base en cada fragmento. La figura 29.5 muestra el contenido de un *encabezado de extensión de fragmento*.

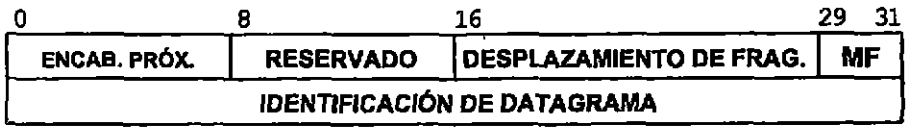


Figura 29.5 Formato de un encabezado de extensión de fragmento.

El IPv6 conserva mucho de la fragmentación del IPv4. Cada fragmento debe ser un múltiplo de *8* octetos, un bit en el campo *MF* marca el último fragmento como el bit del IPv4 *MORE FRAG-*

MENTS, y el campo *DATAGRAM IDENTIFICATION* (*IDENTIFICACIÓN DE DATAGRAMA*) transporta una ID única que el receptor utiliza para el grupo de fragmentos.[3]

29.12 Consecuencia de la fragmentación de extremo a extremo

La motivación para utilizar la fragmentación de extremo a extremo radica en su capacidad para reducir la sobrecarga en los ruteadores y permitir que cada ruteador maneje más datagramas por unidad de tiempo. De hecho, la sobrecarga de CPU requerida por la fragmentación IPv4 puede ser significativa —en un ruteador convencional, la CPU puede alcanzar el 100% de su utilización si el ruteador fragmenta muchos o todos los datagramas que recibe. Sin embargo, la fragmentación de extremo a extremo tiene una consecuencia importante: cambia un supuesto fundamental respecto a Internet.

Para entender la consecuencia de la fragmentación de extremo a extremo, recordemos que el IPv4 está diseñado para permitir a los ruteadores cambiar en cualquier momento. Por ejemplo, si una red o un ruteador falla, el tráfico puede ser redireccionado hacia diferentes trayectorias. La mayor ventaja de este sistema es su flexibilidad —el tráfico puede rutearse hacia una trayectoria alternativa sin interrumpir el servicio y sin informar a la fuente o al destino. En el IPv6, sin embargo, los ruteadores no pueden cambiarse tan fácilmente pues un cambio en una ruta puede cambiar el Path MTU. Si el Path MTU a lo largo de una nueva ruta es menor que el Path MTU a lo largo de la ruta original, un ruteador intermedio debe fragmentar el datagrama original o la fuente original debe ser informada. El problema se puede resumir de la siguiente forma:

> *Un protocolo de red de redes que utilice la fragmentación de extremo a extremo requiere que el emisor descubra el Path MTU para cada destino y que fragmente cualquier datagrama que salga si es mayor que el Path MTU. La fragmentación de extremo a extremo no se adapta al cambio de rutas.*

Para resolver el problema de los cambios de ruta que afectan el Path MTU, el IPv6 permite a los ruteadores intermedios hacer un túnel de IPv6 a través del IPv6. Cuando un ruteador intermedio necesita fragmentar un datagrama, el ruteador no inserta un encabezado de extensión de fragmento ni cambia los campos en el encabezado base. En lugar de ello, el ruteador intermedio crea un datagrama completamente nuevo que encapsula el datagrama original como dato. El ruteador divide el nuevo datagrama en fragmentos reproduciendo el encabezado base e insertando un encabezado de extensión de fragmento en cada uno. Finalmente, el ruteador envía cada fragmento hacia el destino final. En el destino final, el datagrama original puede formarse recolectando los fragmentos entrantes en un datagrama y luego extrayendo la porción de datos. La figura 29.6 ilustra la encapsulación.

---

[3] El IPv6 expande el campo *IDENTIFICATION* a 32 bits para adaptarse a las redes de alta velocidad.

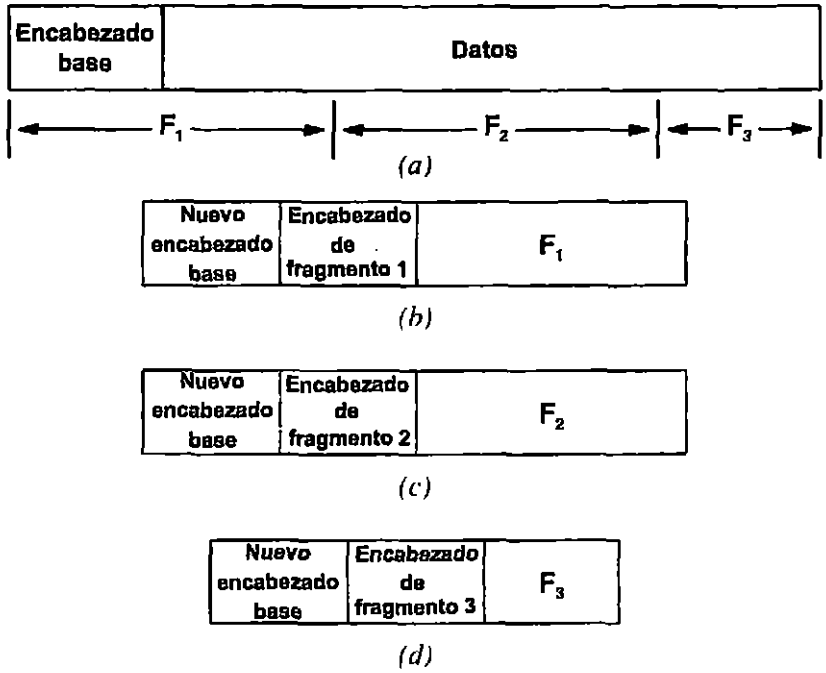


Figura 29.6 (a) Un datagrama IPv6, y de (b) a (d) los tres fragmentos resultantes cuando un ruteador encapsula y fragmenta el datagrama. En el destino se reensamblará el datagrama original incluyendo el encabezado.

29.13 Ruteamiento de origen del IPv6

El IPv6 conserva la capacidad de un emisor para especificar una ruta fuente. A diferencia del IPv4, en el que el ruteo de fuente se proporciona mediante opciones, el IPv6 utiliza un encabezado de extensión separado. Como se muestra en la figura 29.7, los campos de encabezado de ruteo corresponden a los campos de una opción de ruteo de fuente del IPv4. El encabezado contiene una lista de direcciones que especifica ruteadores intermedios a través de los cuales debe pasar el datagrama. El campo *NUM ADDRS* (*NÚMERO DE DIRECCIONES*) especifica el número total de direcciones en la lista y el campo *NEXT ADDRESS* (*DIRECCIÓN PRÓXIMA*) la dirección siguiente hacia la que se enviará el datagrama.

29.14 Opciones del IPv6

Podría parecer que los encabezados de extensión IPv6 reemplazan por completo a las opciones IPv4. Sin embargo, los diseñadores propusieron 2 encabezados de extensión adicionales para adaptarse a cualquier tipo de información no incluida en otros encabezados de extensión. Los encabeza-

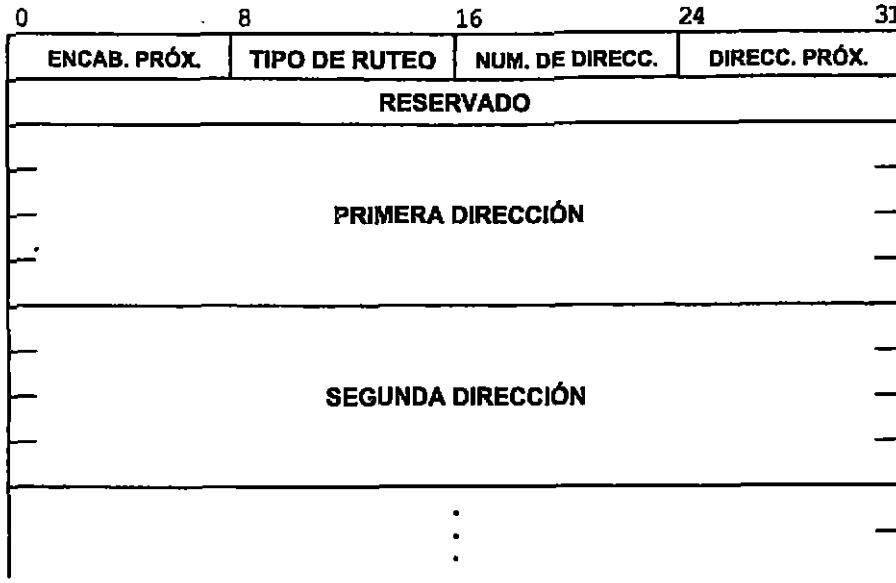


Figura 29.7 Formato de un encabezado de ruteo IPv6. Los campos corresponden a los de una opción de ruta de fuente de IPv4.

dos adicionales consisten en un *Hop By Hop Extension Header* (*extensión de cabeza salto por salto*) y un *End To End Extension Header* (*extensión de cabeza extremo a extremo*). Como lo indican los nombres, los dos encabezados de opción separan el conjunto de opciones que serán examinados en cada salto por el conjunto que será interpretado en el destino.

Aun cuando cada uno de los 2 encabezados de opción tiene un código de tipo único, ambos encabezados utilizan el formato que se ilustra en la figura 29.8.

Como sucede normalmente, el campo *NEXT HEADER* proporciona el tipo de encabezado que sigue. Dado que un encabezado de opción no tiene un tamaño fijo, el campo con el nombre *HEADER LEN* (*LONGITUD DE ENCABEZADO*) especifica la longitud total del encabezado. El

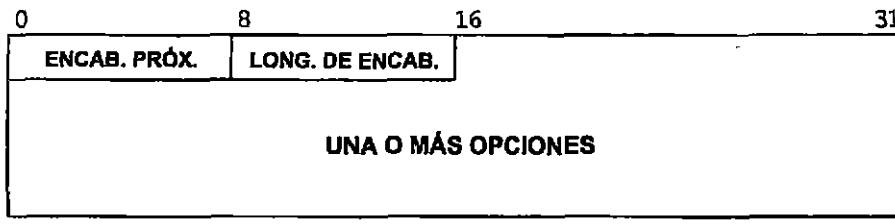


Figura 29.8 Formato del encabezado de una extensión opcional IPv6. Los encabezados de las opciones *salto por salto* y *extremo a extremo* utilizan el mismo formato; el campo *NEXT HEADER* del encabezado anterior distingue entre los dos tipos.

área con el nombre *ONE OR MORE OPTIONS* (*UNA O MÁS OPCIONES*) representa una secuencia de opciones individuales. La figura 29.9 ilustra cómo está codificada cada opción individual con un tipo, longitud y valor;[4] las opciones no están alineadas ni tienen rellenos.

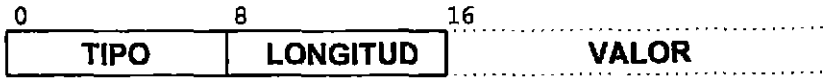


Figura 29.9 Codificación de una opción individual en el encabezado de la extensión opcional IPv6. Cada opción consiste en un tipo de un octeto y una longitud de un octeto seguidos por cero o más octetos de datos para la opción.

Como se muestra en la figura, las opciones IPv6 tienen la misma forma que las opciones IPv4. Cada opción comienza con un campo de un octeto *TYPE* (*TIPO*) seguido de un campo *LENGTH* (*LONGITUD*) de un octeto. Si la opción requiere de datos adicionales, los octetos que comprenden *VALUE* (*VALOR*) se siguen de LENGTH.

Los dos bits de orden superior de cada opción de campo *TYPE* especifican cómo deberán disponer un anfitrión o un ruteador del datagrama si no comprende las opciones:

Bits en tipo	Significado
00	Saltar esta opción
01	Desechar el datagrama; no enviar mensaje ICMP
10	Desechar datagrama; enviar mensaje ICMP a la fuente
11	Desechar datagrama; no enviar ICMP para multidifusión

29.15 Tamaño del espacio de dirección del IPv6

En el IPv6, cada dirección ocupa 16 octetos, 4 veces el tamaño de una dirección IPv4. El amplio espacio de direcciones garantiza que el IPv6 puede tolerar cualquier esquema de asignación de direcciones razonable. De hecho, si los diseñadores deciden cambiar el esquema de direccionamiento más tarde, el espacio de direcciones es lo suficientemente extenso como para adaptarse a una reasignación.

Es difícil comprender el tamaño del espacio de direcciones del IPv6. Una forma de entenderlo es relacionando la magnitud con el tamaño de la población: el espacio de direcciones es tan grande que cada persona en el planeta puede tener direcciones suficientes como para poseer una red de redes tan grande como la Internet actual. Otra forma de entender el tamaño es relacionarlo con el agotamiento de direcciones. Por ejemplo, consideremos qué se necesitaría para asignar todas las posibles direcciones. Un entero de 16 octetos puede manejar 2^{128} valores. Así, el espacio de di-

[4] En la jerga, una codificación de tipo, longitud y valor se conoce en ocasiones como *codificación TLV*.

recciones es mayor que 3.4×10^{38}. Si las direcciones se asignaran a razón de un millón de direcciones por milisegundo, tomaría alrededor de 20 años asignar todas las direcciones posibles.

29.16 Notación hexadecimal con dos puntos del IPv6

Aun cuando resuelve el problema de tener una capacidad insuficiente, el gran tamaño de direcciones plantea un problema nuevo: los usuarios que manejan redes de redes deben leer, introducir y manipular estas direcciones. Obviamente, la notación binaria no es práctica. Sin embargo, la notación decimal con puntos utilizada por el IPv4 tampoco hace las direcciones lo suficientemente compactas. Para entender por qué, consideremos el ejemplo de un número de 128 bits expresado en notación decimal con puntos:

```
104.230.140.100.255.255.255.255.0.0.17.128.150.10.255.255
```

Para ayudar a hacer la dirección ligeramente más compacta y fácil de introducir, los diseñadores del IPv6 proponen utilizar una *notación hexadecimal con dos puntos* (abreviado *colon hex*) en la cual el valor de cada cantidad de 16 bits se representa en forma hexadecimal separado por dos puntos. Por ejemplo, cuando el valor mostrado arriba en notación decimal se traduce a la notación hexadecimal con dos puntos e impresa, utilizando el mismo espaciado, se convierte en:

```
68E6:8C64:FFFF:FFFF:0:1180:96A:FFFF
```

La notación hexadecimal con dos puntos tiene la ventaja obvia de requerir menos dígitos y menos caracteres separadores que la notación decimal con puntos. Además, la notación hexadecimal con dos puntos incluye dos técnicas que la hacen muy útil. En la primera, la notación hexadecimal con dos puntos permite la *compresión 0* mediante la cual una cadena de ceros repetidos se reemplaza por un par de dos puntos. Por ejemplo, la dirección:

```
FF05:0:0:0:0:0:0:B3
```

puede escribirse:

```
FF05::B3
```

Para asegurar que la compresión cero produce una interpretación sin ambigüedades, la propuesta especifica que puede aplicarse sólo una en cualquier dirección. La compresión cero es especialmente útil cuando se emplea el esquema de asignación de direcciones propuesto ya que muchas direcciones contendrán cadenas contiguas de ceros. En segundo lugar, la notación hexadecimal con dos puntos incorpora sufijos decimales con punto; como veremos, esta combinación tiene el propósito de utilizarse durante la transición del IPv4 al IPv6. Por ejemplo, la siguiente cadena es una notación hexadecimal con dos puntos válida:

```
0:0:0:0:0:0:128.10.2.1
```

Obsérvese que, aun cuando los números están separados por dos puntos, cada uno especifica el valor de una cantidad de 16 bits, los números en la porción decimal con puntos especifican el valor de un octeto. Por supuesto, la compresión cero puede utilizarse con el número de arriba para producir una cadena hexadecimal con dos puntos equivalente que se vería muy similar a una dirección IPv4:

```
::128.10.2.1
```

29.17 Tres tipos básicos de dirección IPv6

Como el IPv4, el IPv6 asocia una dirección con una conexión de red específica, no con una computadora específica. Así, la asignación de direcciones es similar para el IPv4: un ruteador IPv6 tiene dos o más direcciones, y un anfitrión IPv6, con una conexión de red, necesita sólo una dirección. El IPv6 también conserva (y extiende) la jerarquía de direcciones del IPv4 en la que una red física es asignada a un prefijo. Sin embargo, para hacer la asignación de direcciones y la modificación más fácil, el IPv6 permite que varios prefijos sean asignados a una red dada y que una computadora tenga varias direcciones simultáneas asignadas hacia una interfaz determinada.

Además de permitir varias direcciones simultáneas por conexión de red, el IPv6 expande y, en algunos casos, unifica las direcciones especiales del IPv4. En general, una dirección de destino en un datagrama cae dentro de una de tres categorías:

Unidifusión	La dirección de destino especifica una sola computadora (anfitrión o ruteador); el datagrama deberá rutearse hacia el destino a lo largo de la trayectoria más corta.
Grupo	El destino es un conjunto de computadoras en el que todas comparten un solo prefijo de dirección (por ejemplo, si están conectadas a la misma red física); el datagrama deberá rutearse hacia el grupo a través de la trayectoria más corta y, después, entregarse exactamente a un miembro del grupo (por ejemplo, el miembro más cercano).
Multidifusión	El destino es un conjunto de computadoras, posiblemente en múltiples localidades. Una copia del datagrama deberá entregarse a cada miembro del grupo que emplee hardware de multidifusión o de difusión si están disponibles.

29.18 Dualidad de difusión y multidifusión

El IPv6 no emplea el término *difusión* o *difusión dirigida* para referirse a la entrega a todas las computadoras en una red física o a una subred IP lógica. En cambio, utiliza el término *multidifusión*, y trata a la difusión como una forma especial de multidifusión. La elección puede parecer ex-

traña para cualquiera que conozca el hardware de red ya que la mayor parte de las tecnologías de hardware soporta la difusión así como la multidifusión. De hecho, un ingeniero de hardware es probable que vea a la multidifusión como una forma restringida de difusión —el hardware envía un paquete de multidifusión hacia todas las computadoras en la red exactamente como un paquete de difusión, y el hardware de interfaz en cada computadora filtra todos los paquetes de multidifusión excepto los que el software ha definido para que los acepte el hardware de interfaz.

En teoría, la elección entre la multidifusión y una forma limitada de difusión es irrelevante pues se puede simular una con la otra. Esto es, la difusión y la multidifusión son dos formas diferentes que proporcionan la misma funcionalidad. Para entender por qué, consideremos cómo simular una con la otra. Si la difusión está disponible, un paquete puede entregarse a un grupo enviándolo a todas las máquinas y haciendo que el software en cada computadora decida si acepta o descarta el paquete entrante. Si la multidifusión está disponible, un paquete puede ser entregado a todas las máquinas haciendo que todas las máquinas escuchen el mismo grupo de multidifusión similar de *todos los anfitriones* (tratado en el capitulo 17).

29.19 Una elección de ingeniería y difusión simulada

Saber que la difusión y la multidifusión son teóricamente equivalentes no ayuda a la elección entre éstas. Para ver por qué los diseñadores del IPv6 eligieron la multidifusión como la abstracción central en lugar de la difusión, consideremos las abstracciones en lugar de observar el hardware subyacente. Una aplicación necesita comunicarse con otra aplicación o con otro grupo de aplicaciones. La comunicación directa se maneja mejor vía unidifusión; la comunicación en grupo se maneja mejor por medio de la multidifusión o la difusión. Para proporcionar la mayor flexibilidad, los miembros de un grupo no deben determinar las conexiones de red, ya que puede haber miembros que residan en localidades arbitrarias. Utilizar la difusión para la comunicación de todo el grupo no conduce a manejar una red de redes tan extensa como la red global de Internet.

No es sorprendente, pues, que los diseñadores predefinieran las direcciones de multidifusión que corresponden a las redes del IPv4 y a las direcciones de difusión de subred. Así, además de sus propias direcciones de unidifusión, cada anfitrión es requerido para aceptar paquetes direccionados hacia el grupo de multidifusión *todos los nodos* y hacia el grupo de multidifusión *todos los anfitriones* para su entorno local; también, existe la dirección *todos los ruteadores*.

29.20 Asignación propuesta de espacio de dirección IPv6

La cuestión sobre cómo dividir el espacio de direcciones ha generado muchas discusiones. Hay dos temas centrales: cómo administrar la asignación de direcciones y cómo transformar una dirección en una ruta. El primer tema se enfoca en el problema práctico de construir una jerarquía de autoridad. A diferencia de la Internet actual, la cual utiliza una jerarquía de dos niveles de prefijos de red (asignados por la autoridad de Internet) y sufijos de anfitrión (asignados por la organización), el gran espacio de direcciones en el IPv6 permite una jerarquía de multiniveles o jerarquías múltiples. El segundo tema se enfoca en la eficiencia computacional. Independientemente de la jerarquía de

autoridad que asigne direcciones, un ruteador debe examinar cada datagrama y elegir una trayectoria hacia el destino. Para mantener bajo el costo de los ruteadores de alta velocidad, el tiempo de procesamiento requerido para elegir una trayectoria debe mantenerse bajo.

Como se muestra en la figura 29.10, los diseñadores del IPv6 proponen asignar clases de direcciones en forma similar al esquema utilizado por el IPv4. Aun cuando los ocho primeros bits de una dirección son suficientes para especificar su tipo, el espacio de direcciones no se divide en secciones de igual tamaño.

Prefijo binario	Tipo de dirección	Parte del espacio de dirección
0000 0000	Reservado (compatible con IPv4)	1 /256
0000 0001	Reservado	1 /256
0000 001	Direcciones NSAP	1/128
0000 010	Direcciones IPX	1/128
0000 011	Reservado	1/128
0000 100	Reservado	1/128
0000 101	Reservado	1/128
0000 110	Reservado	1/128
0000 111	Reservado	1 /28
0001	Reservado	1/16
001	Reservado	1/8
010	Unidifusión proveedor asignado	1/8
011	Reservado	1/8
100	Reservado (geográfico)	1/8
101	Reservado	1/8
110	Reservado	1/8
1110	Reservado	1/16
1111 0	Reservado	1/32
1111 10	Reservado	1/64
1111 110	Reservado	1/128
1111 1110	Disponible para uso local	1/ 256
1111 1111	Utilizado para multidifusión	1/256

Figura 29.10 División propuesta de las direcciones IPv6 en tres tipos, los cuales son análogos a las clases IPv4. Como en el IPv4, el prefijo de una dirección determina su tipo de dirección.

29.21 Codificación y transición de la dirección IPv4

Observe de la figura 29.10 que alrededor del *72%* del espacio de direcciones ha sido reservado para usos futuros, no incluyendo la sección reservada para direcciones geográficas. Aun cuando el

prefijo *0000 0000* tiene el nombre *reservado* en la figura, los diseñadores planean usar una pequeña fracción de direcciones en esta sección para codificar direcciones IPv4. En particular, cualquier dirección que comience con 80 bits puestos a cero, seguidos por 16 bits puestos a 1 o 16 bits puestos todos a cero, contiene una dirección IPv4 en los 32 bits de orden inferior. La codificación será necesaria durante la transición del IPv4 al IPv6 por dos razones. En primer lugar, una computadora puede elegir actualizar su software de IPv4 como IPv6 antes de tener asignada una dirección IPv6 válida. En segundo lugar, una computadora que corra software IPv6 puede necesitar comunicarse con una computadora que corra sólo software IPv4.

Teniendo una forma de codificar una dirección IPv4 en una dirección IPv6 no se resuelve el problema de lograr que las dos versiones interoperen. Además de la codificación de direcciones, es necesaria la traducción. Para utilizar un traductor, una computadora IPv6 genera un datagrama que contiene la codificación IPv6 de la dirección de destino IPv4. La computadora IPv6 envía el datagrama hacia un traductor, el cual utiliza IPv4 para comunicarse con el destino. Cuando el traductor recibe una réplica desde el destino, traduce el datagrama IPv4 a IPv6 y lo envía de regreso a la fuente IPv6.

Parecería como si el protocolo de traducción de direcciones fallara debido a que las capas superiores de los protocolos verifican la integridad de las direcciones. En particular, el TCP y el UDP utilizan un *pseudo encabezado* en su cálculo para la suma de verificación. El pseudo encabezado incluye la dirección del protocolo de la fuente y el destino, cambiar estas direcciones puede afectar el cálculo. Sin embargo los diseñadores planearon cuidadosamente que el TCP o el UDP en una máquina IPv4 se pudieran comunicar con el correspondiente protocolo de transporte en una máquina IPv6. Para evitar errores en la suma de verificación, la codificación IPv6 de una dirección IPv4 ha sido elegida de manera que el complemento a uno de los 16 bits de la suma de verificación para una dirección IPv4 y la codificación IPv6 de la dirección sean idénticos. El punto es el siguiente:

> *Además de seleccionar detalles técnicos de un nuevo protocolo de Internet, el IETF que trabaja en el IPng se ha enfocado en encontrar una forma de transición del protocolo actual al protocolo nuevo. En particular, la propuesta actual para el IPv6 permite codificar una dirección IPv4 en lugar de una dirección IPv6, de manera que la traducción de la dirección no cambie la suma de la verificación del pseudo encabezado.*

29.22 Proveedores, suscriptores y jerarquía de direcciones

Un ejemplo ayudará a entender cómo concibieron los diseñadores el uso de las direcciones IPv6. Consideremos la compañía *Network Access Provider* (*NAP*). Dicha compañía ofrece a sus clientes conectividad hacia Internet, a tales clientes los llamaremos *suscriptores*. Para permitir que cada proveedor asigne direcciones, la autoridad de Internet asigna a cada proveedor un identificador único. El proveedor puede entonces asignar a cada proveedor un identificador único y utilizar ambos identificadores cuando asigne un bloque de direcciones. El suscriptor puede asignar entonces un ID único para cada red física y a cada computadora en una red un ID de nodo único. La figura 29.11 ilustra la posible división de una dirección en subcampos.

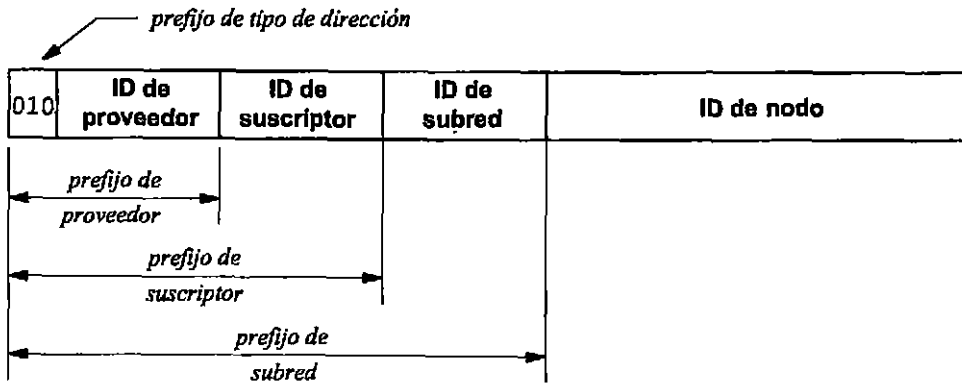


Figura 29.11 Jerarquía de direcciones IPv6 para una dirección asignada por un proveedor de acceso a red. La autoridad de Internet asigna a cada proveedor una ID única, el proveedor asigna una ID única a cada suscriptor y el suscriptor asigna una única ID a cada subred en cada nodo.

Como se muestra en la figura 29.11, cada prefijo sucesivamente más largo tiene un nombre. La cadena inicial *010* identifica la dirección como el tipo de asignación del proveedor. Para cada dirección, el *prefijo de proveedor* incluye el tipo de dirección más el ID del proveedor. El *prefijo de suscriptor* cubre el prefijo del proveedor más el ID del suscriptor. Por último, el *prefijo de subred* incluye el prefijo de suscriptor más la información de subred.

Los campos en la figura 29.11 no están dibujados a escala. Por ejemplo, aun cuando los prefijos de direcciones parecen grandes en la figura, ocupan sólo tres de 128 bits. Los diseñadores recomiendan que el campo *ID del nodo* contenga por lo menos 48 bits para permitir que se utilice el direccionamiento de tipo 802 de IEEE. Así, será posible para un nodo IPv6 usar su dirección Ethernet como su ID de nodo.

29.23 Jerarquía adicional

Aunque el formato de direcciones mostrado arriba implica una jerarquía de cuatro niveles, una organización puede introducir niveles adicionales dividiendo el campo *Subnet ID* en varios campos. Por ejemplo, una organización puede elegir subdividir su subred en áreas y asignar subredes dentro de las áreas. Hacer esto es similar al esquema de direccionamiento de subred del IPv4, en el que la porción del anfitrión de una dirección es dividida en dos partes. El amplio espacio de direccionamiento del IPv6 permite la división en muchas partes.

29.24 Resumen

Ni la red global Internet ni los protocolos TCP/IP son estáticos. A través de su Engineering Task Force, la Internet Architecture Board se mantiene activa y realiza esfuerzos que hacen que la tecnología evolucione y mejore. Los procesos que conducen al cambio se manifiestan como un incremento en el tamaño y en la carga que obliga a mejorar los recursos para mantener el servicio, como aplicaciones nuevas que demandan más de la tecnología subyacente y como tecnologías nuevas que hacen posible proporcionar nuevos servicios.

Un esfuerzo para definir la próxima generación de protocolo de Internet (IPng) ha generado una gran polémica y varias propuestas. Ha surgido un acuerdo de IETF para adoptar una propuesta conocida como *Simple IP Plus* como estándar para el IPng. Debido que se deberá asignar el número de versión *6*, el protocolo propuesto se conoce a menudo como IPv6 para distinguirlo del protocolo actual, IPv4.

El IPv6 conserva muchos de los conceptos básicos del IPv4, pero cambia la mayor parte de los detalles. Como el IPv4, el IPv6 proporciona un servicio de entrega de datagramas sin conexión, con el mejor esfuerzo. Sin embargo el formato del datagrama IPv6 es completamente diferente del formato IPv4, y el IPv6 proporciona características nuevas como la autenticación, un mecanismo para flujos controlados de datagramas y soporte para seguridad.

El IPv6 revisa cada datagrama como una serie de encabezados seguidos por datos. Un datagrama siempre comienza con un encabezado base de 40 octetos, el cual contiene la dirección de fuente y destino y un identificador de flujo. El encabezado base puede estar seguido de ceros o por más encabezados de extensión, seguidos por datos. Los encabezados de extensión son opcionales —el IPv6 los utiliza para manejar gran parte de la información que el IPv4 codifica en opciones.

Una dirección IPv6 tiene una longitud de 128 bits, lo que hace que el espacio de dirección sea tan largo que cada persona en el planeta podría tener una red de redes tan extensa como la Internet actual. El IPv6 divide las direcciones en tipos en forma análoga a como el IPv4 las divide en clases. Un prefijo de la dirección determina la localización y la interpretación de los campos de dirección restantes. Muchas direcciones IPv6 serán asignadas por proveedores de servicio de red autorizados, dichas direcciones tienen campos que contienen un ID de proveedor, un ID de suscriptor, un ID de subred y un ID de nodo.

PARA CONOCER MÁS

Han aparecido muchos RFC que contienen información relacionada con el IPng, incluyendo análisis sobre requerimientos, procedimientos y propósitos específicos. Bradner y Mankin (RFC 1550) hacen una invitación para propuestas y discusiones; muchos RFC responden. Por ejemplo, Britton y Tavs (RFC 1678) y Fleischman (RFC 1687) hacen comentarios sobre el IPng en redes corporativas amplias. Gross (RFC 1719) trata una dirección general, Partridge y Kastenholz (RFC 1726) analizan los criterios técnicos para elegir la tecnología IPng, y Brazdziunas (RFC 1680) comenta sobre el soporte IPng para ATM. Bellovin (RFC 1675) comenta la seguridad en el IPng.

Bradner y Mankin (RFC 1752) resumen propuestas y contienen las recomendaciones de los administradores de área de IETF para el IPng. Los lectores deben estar conscientes de que las pro-

puestas para el IPv6 no son ya un estándar probado a fondo y que más adelante algunos detalles probablemente cambien.

EJERCICIOS

29.1 El IPv6 propuesto no tiene suma de verificación del encabezado. ¿Cuáles son las ventajas y las desventajas de este método?

29.2 ¿Cómo deberían ordenarse los encabezados de extensión para minimizar el tiempo de procesamiento?

29.3 Aun cuando las direcciones del IPv6 son asignadas jerárquicamente, un ruteador no necesita analizar una dirección completamente para seleccionar una ruta. Construya un algoritmo y una estructura de datos para obtener un ruteo eficiente. Sugerencia: considere un enfoque de mayor semejanza.

29.4 Demuestre que los 128 bits de direcciones son más de lo que se necesitaría y que 96 bits proporcionan capacidad suficiente.

29.5 Suponga que su organización trata de adoptar el IPv6. Construya el esquema de direccionamiento que utilizaría la organización para asignar a cada anfitrión una dirección. ¿Seleccionaría una asignación jerárquica dentro de su organización? ¿Por qué sí o por qué no?

29.6 ¿Cuál es la mayor ventaja de codificar una dirección Ethernet en una dirección IPv6? ¿Cuál es la mayor desventaja?

29.7 Si usted tuviera que seleccionar tamaños para los campos ID de proveedores, suscriptores y subredes de una dirección IPv6, ¿de qué tamaño haría cada uno? ¿Por qué?

29.8 Lea sobre los encabezados de autenticación y seguridad del IPv6. ¿Por qué se proponen dos encabezados?

Apéndice 1

Guía de RFC

Introducción

La mayor parte de la información escrita sobre el TCP/IP e Internet, así como sobre su arquitectura, protocolos e historia, puede encontrarse en una serie de reportes conocidos como *Request For Comments (Solicitud de comentarios)* o *RFC*. Este conjunto de notas informales y con una coordinación poco rigurosa tiene una riqueza de información y un colorido poco frecuentes. Antes de considerar los aspectos más serios de los RFC, tomaremos unos cuantos minutos para prestar atención al aspecto de mayor colorido. Un buen lugar para comenzar es con el poema de Cerf *Twas the Night Before Start-up* (RFC 968), una parodia humorística que describe algunos de los problemas que se encuentran cuando se inicia una nueva red. Aprender a no tomarse las cosas tan en serio ha hecho crecer el esfuerzo de Internet. Cualquiera que recuerde su primer encuentro con Internet, lleno de la jerga de redes y el *Jabberwocky* de Lewis Carrol, plagado de giros extraños del idioma inglés, comprenderá exactamente por qué D. L. Covill los reunió en *ARPAWOCKY* (RFC 527). Cualquiera que haya leído *The Art Of Computer Programming* puede reír en el RFC 473, preguntando dónde se pueden ejecutar programas MIX en ARPANET. Podemos imaginar a Pickens lleno de orgullo cuando responde con el RFC 485, *MIX y MAXIMAL en UCSB*. La Universidad de California en Santa Bárbara no estaba sola al ofrecer MIX al mundo. En RFC 494, Walden proporciona una lista de todos los anfitriones en la red que soportan programación MIX.

Otros RFC parecen igualmente frívolos. Explorando entre las descripciones de ideas que provocarían cambios dramáticos en las redes, encontramos notas como el RFC 416, escritas a principios de noviembre de 1972: *The ARC System will be Unavailable for Use During Thanksgiving Week*. Dice exactamente lo que usted piensa que dice. O considere el humor irónico de Crispin en RFC 748, el cual describe la *TELNET Randomly-Lose Option* (opción propuesta para TELNET que suprime caracteres aleatoriamente). Sin notas como ésta no parecen insignificantes, piense en

los sesenta y siete RFC listados como *nunca publicados*. Todos éstos fueron asignados a un número y tuvieron un autor pero nunca salieron a la luz. Todos se mantiene como huecos en el esquema de numeración, preservados como pequeños restos de ideas que se evaporaron o de trabajos que quedaron incompletos.

Incluso, luego de que se han retirado RFC delirantes, absurdos e inútiles, los documentos restantes no cumplen con la mayor parte de los estándares para los escritos científicos. A diferencia de las revistas científicas escolares que se concentran en identificar artículos de importantes archivos de interés, presentándolos cuidadosamente y clasificándolos para la posteridad, los RFC proporcionan un registro de las conversaciones en curso entre los principales involucrados en el diseño, construcción, medición y uso de la red global de Internet. Los lectores comprenden enseguida que los RFC muestran el pensamiento de los investigadores que están al frente de la innovación tecnológica, no las opiniones estudiadas de escolares que tienen completamente dominado un tema. Los autores no siempre están seguros de las consecuencias de sus propuestas o, incluso, de su contenido, pero se dan cuenta claramente de que las publicaciones son más difíciles de entender sin una discusión comunitaria.

A pesar de las inconsistencias de los RFC, que a veces los hacen difíciles de entender para los principiantes, el mecanismo de los RFC ha evolucionado y ahora trabaja extraordinariamente bien. Dado que los RFC están disponibles electrónicamente, la información se difunde entre la comunidad con rapidez. Dado que abarcan una amplia variedad de intereses, contribuyen tanto los practicantes como los diseñadores. Como registran conversaciones informales, los RFC capturan discusiones y no sólo conclusiones finales. Incluso, los desacuerdos y las propuestas contrarias son útiles pues muestran lo que los diseñadores consideraban antes de establecer un protocolo determinado (los lectores interesados en la historia de una idea o protocolo en particular pueden utilizar los RFC para seguirlos desde sus inicios hasta su estado actual).

Importancia de los documentos de requerimientos para anfitriones y compuertas

A diferencia de la mayor parte de los RFC, los cuales se concentran en una sola idea o protocolo, tres RFC especiales cubren un amplio rango de protocolos. Los documentos especiales se titulan *Requirements for Internet Gateways* y *Requirements for Internet Host* (partes 1 y 2).

Los documentos de requerimientos fueron publicados a finales de 1980, luego de muchos años de experiencia con los protocolos TCP/IP, y se consideran la mayor revisión para estándares de protocolos. En esencia, cada uno de los documentos de requerimientos revisa muchos protocolos. Hacen énfasis en debilidades conocidas o ambigüedades de los RFC que definen los protocolos, establecen convenciones que han sido adoptadas por los vendedores, documentan problemas que se presentan en la práctica y listan soluciones que se han aprendido y acumulado mediante la experiencia con respecto a estos problemas. Los RFC para protocolos individuales *no* han sido actualizados y, por ello, no incluyen cambios y actualizaciones de los documentos de requerimientos. Así, los lectores deben ser cuidadosos y deben consultar siempre los documentos de requerimientos cuando estudien un protocolo en particular.

Cómo obtener un RFC en Internet

Los RFC están disponibles de manera electrónica en cualquier depósito del mundo. Consulte a su administrador de red local para encontrar la localidad más cercana. Si no puede hallar una, utilice las instrucciones que se dan a continuación para accesar el INTERnet Network Informatios Center (INTERNIC).

El nombre de dominio de Internet para el anfitrión que proporciona el archivo es:

ds.internic.net

Para conseguir la copia de un archivo de texto de un RFC directamente del archivo, utilice el Protocolo de Transferencia de Archivo (FTP) en una computadora conectada a Internet. Luego de invocar a un cliente FTP, proporcione los comandos de recuperación. Primero, envíe el comando *user* para identificarse con el servidor remoto. Proporcione el nombre de usuario *anonymous* y la clave de acceso *guest* cuando se le indique. Una vez que el servidor haya reconocido su nombre y su clave de acceso, emplee el comando *get* para recuperar el archivo llamado:

get rfc/rfc*N*.txt *LocalFile*

donde *N* es el número del RFC deseado,[1] y *LocalFile* es el nombre de un archivo en el que el programa *ftp* debe almacenar la copia. Por ejemplo, para obtener una copia del RFC 822, anote el comando:

get rfc/rfc822.txt *LocalFile*

El archivo recuperado contendrá texto en ASCII con un carácter de alimentación separando cada página y otro separando cada línea nueva (alimentación de línea). Excepto para los caracteres de alimentación de línea y de página, el archivo completo contiene texto que se puede imprimir con una impresora convencional. No se incluyen dibujos ni ningún otro tipo de gráficos en particular.

La siguiente secuencia de comandos de UNIX ilustra cómo se puede elaborar un programa que se valga del FTP para recuperar un RFC.

---

[1] Algunos RFC sólo están disponibles en postcript; sus nombres terminan con *.ps*.

```
---------------------------------------------------------------------
#! /bin/sh
#
# rfc - UNIX (Bourne) programa para obtener copias de uno o más RFC
#           conservando una memoria inmediata local para solicitudes
#           subsecuentes
#
# use: rfc number [ number...]
#
PATH=/bin:/usr/bin:/usr/ucb
PUB=/usr/pub/RFC
INTERNIC=ds.internic.net
for i in $*
do          if test ! -r $PUB/RFC.$i -o $i = "index"
            then echo Retrieving RFC $i from $INTERNIC > &2
#
# invoca a FTP bajo UNIX y proporciona comandos de recuperación como
#           entrada.
#
                        ftp -n $INTERNIC > /dev/null 2&1 <<!
user anonymous guest
get rfc/rfc$i.txt $PUB/RFC.$i
quit
!
    fi
#
# Habiendo obtenido el archivo, proporciona una copia al usuario si
#           la recuperación fue exitosa.
#
            if test -r $PUB/RFC.$i
            then cat $PUB/RFC.$i
            else echo Could not retrieve RFC $i > 1&2
            fi
done
---------------------------------------------------------------------
```

El texto del programa mostrado arriba no hace más que utilizar el FTP para recuperar un RFC. Deja una copia del RFC en el directorio */usr/pub/RFC*. La ventaja de conservar la copia local de un RFC es que las solicitudes subsecuentes son mucho más rápidas que la primera ya que no se utiliza el FTP ni se debe pasar información a través de Internet. Si el programa encuentra una de las RFC solicitadas en la memoria inmediata, sólo presentará una copia al usuario. Observe que el programa no busca en la memoria intermedia cuando recupera el archivo especial *-index* pues el índice contiene una lista de todos los RFC y cambia cuando aparecen RFC nuevos.

Cómo obtener un RFC a través del correo electrónico

INTERNIC y muchas otras localidades operan servidores de información que responden a los mensajes de correo electrónico. Esto es, se envía un mensaje de correo electrónico a una dirección de e-mail especial, un programa de computadora lee el correo entrante, consulta la información de su base de datos y responde por medio de e-mail. La dirección de e-mail del servidor de información de INTERNIC es:

mailserv@ds . internic . net

La base de datos contiene documentos de texto de los RFC junto con otro tipo de información. Para obtener un RFC, un mensaje e-mail debe incluir la línea:

send rfc*N* . *txt*

donde *N* es el número de un RFC. para más información, envíe un mensaje que contenga sólo la línea:

help

Cómo obtener una copia en papel de un RFC

Las personas que no pueden accesar las redes electrónicas tampoco pueden obtener copias de un RFC. En Estados Unidos, el número para llamar sin cargos es 1-800-444-4345. Antes de llamar, utilice este apéndice para hacer una lista de los RFC que necesite.

Explorando los RFC

Hay varios índices que pueden ayudar a explorar los RFC. En primer lugar, el archivo *rfc/rfc-index.txt* contiene una lista precisa de todos los RFC en orden cronológico inverso. Se conserva en el archivo junto con los archivos de texto para los RFC. Cualquiera puede obtener el índice utilizando el FTP o e-mail; los usuarios que desean explorar a través de los RFC por lo regular obtienen el índice para comprobar si están enterados de las últimas versiones de RFC. En segundo lugar, muchos RFC contienen resúmenes o índices de otros RFC. Por ejemplo, el RFC 899 contiene un índice de todos los RFC, del 800 al 899, en orden cronológico inverso. En tercer lugar, los lectores con frecuencia necesitan saber qué RFC contiene la última versión de un protocolo oficial de Internet o qué protocolos son oficiales y cuáles no lo son. Para adaptarse a tales necesidades, la IAB publica de manera periódica un RFC titulado *INTERNET OFFICIAL PROTOCOL STANDARDS*. en el cual se proporciona la lista de todos los protocolos que han sido adoptados como estándares TCP/IP, junto con el número de RFC o los RFC más recientes que describen cada protocolo. Además, el RFC 1602, *The Internet Standards Process - Revision 2*, describe el proceso de estandariza-

ción de Internet y define el significado de los términos *proposed standard, draft standard, Internet standard, required, recommended* e *historic.*

La *Internet Assigned Numbers Authority (IANA)* en el Instituto de Ciencias de la Información de la Universidad del Sur de California, publica información sobre constantes de protocolos en RFC con el título *Internet Numbers*. Los RFC de Internet Numbers contienen valores utilizados en varios campos de los protocolos oficiales (por ejemplo, el RFC Internet Numbers especifica que el campo *protocol* en el encabezado de un datagrama IP debe contener el valor *6* cuando el datagrama contiene un segmento TCP).

A pesar de los índices disponibles, explorar los RFC puede ser difícil, especialmente cuando el lector está buscando información relacionada con un tema determinado. Leer una lista cronológica de todos los RFC es tedioso, pero no hay mecanismos que permitan encontrar grupos relacionados de RFC. Para empeorar el problema, la información sobre un tema dado puede aparecer diseminada a lo largo de varios años. Explorar a través de un índice cronológico de RFC puede ser particularmente difícil dado que los títulos no proporcionan una identificación suficiente sobre la información en los RFC. (¿Cómo puede uno adivinar que bajo el título *Leaving Well Enough Alone* hay un RFC en relación con el FTP?) Por último, como hay varios RFC con un solo título, (por ejemplo, Internet Numbers) la búsqueda puede resultar confusa ya que el lector no concluye fácilmente si un documento es obsoleto, sin comprobarlo de manera directa en el archivo.

RFC agrupados por tema

La sección final de este apéndice ayuda a encontrar información en los RFC ya que contiene una lista de los primeros 1750 RFC agrupados por tema. Los lectores pueden encontrar un índice por temas anterior en RFC 1000, el cual también incluye una lista ordenada de manera cronológica de los primeros 1000 RFC. Aun cuando es un documento extenso, el RFC 1000 es recomendable como una fuente autorizada y como una crítica valiosa, su introducción es especialmente fascinante. Al recordar que los RFC se originaron junto con ARPANET, la introducción captura el espíritu de aventura y la energía que todavía caracteriza a Internet.

RFC organizados por categorías mayores y subtemas
(Para consultar una versión anterior, ver RFC 1000)

1. Administrativos

1a. Números asignados a Internet (valores oficiales utilizados para los protocolos)

1700, 1340, 1117, 1062, 1060, 1020, 1010, 997, 990, 960, 943, 923, 900, 870, 820, 790, 776, 770, 762, 758, 755, 750, 739, 717, 604, 503, 433, 349, 322, 317, 204, 179, 175, 167.

1b. Estándares IAB oficiales y otras listas de protocolos

1720, 1610, 1600, 1540, 1500, 1410, 1360, 1280, 1250, 1200, 1140, 1130, 1100, 1083, 1011, 991, 961, 944, 924, 901, 880, 840, 694, 661, 617, 582, 580, 552.

774, 766 - Internet Protocol Handbook Table of Contents

1c. Notas de encuentros y actas

1636 - Report of IAB Workshop on Security in the Internet Architecture - February 8-10, 1994

1210 - Network and Infrastructure User Requirements for Transatlantic Research Collaboration - Brussels, July 16-18, and Washington July 24-25, 1990

1152 - Workshop report: Internet research steering group workshop on very-high-speed networks

1077 - Critical issues in high bandwidth networking

1019 - Report of the Workshop on Environments for Computational Mathematics

1017 - Network requirements for scientific research: Internet task force on scientific computing

898 - Gateway Special Interest Group Meeting Notes

808, 805, 469 - Computer Mail Meeting Notes

910, 807 - Multimedia Mail Meeting Notes

585 - ARPANET Users Interest Working Group Meeting

549, 396, 282, 253 - Graphics Meeting Notes

371 - International Computer Communications Conference

327 - Data and File Transfer Workshop Notes

316 - Data Management Working Group Meeting Report

164, 131, 108, 101, 082, 077, 063, 037, 021 - Network Working Group Meeting

1d. Anuncios de encuentros e información general sobre grupos

1588 - WHITE PAGES MEETING REPORT
1160, 1120 - Internet Activities Board
828 - Data Communications: IFIP's International "Network" of Experts
631 - Call for Papers: International Meeting on Minicomputers and Data Communication
584 - Charter for ARPANET Users Interest Working Group
537 - Announcement of NGG Meeting
526 - Technical Meeting - Digital Image Processing Software Systems
504 - Workshop Announcement
483 - Cancellation of the Resource Notebook Framework Meeting
474, 314, 246, 232, 134 - Network Graphics Working Group
471 - Announcement of a (Tentative) Workshop on Multi-Site Executive Programs
461 - Telnet Meeting Announcement
457 - TIPUG
456 - Memorandum
454 - File Transfer Protocol Meeting Announcement
453 - Meeting Announcement to Discuss a Network Mail System
374 - IMP System Announcement
359 - The Status of the Release of the New IMP System (2600)
343, 331 - IMP System Change Notification
324 - RJE Protocol Meeting
323 - Formation of Network Measurement Group (NMG)
320 - Workshop on Hard Copy Line Graphics
309 - Data and File Transfer Workshop Announcement
299 - Information Management System
295 - Report of the Protocol Workshop
291, 188, 173 - Data Management Meetings
245, 234, 207, 140, 116, 099, 087, 085, 075, 043, 035 - Network Working Group Meetings
222 - System Programmer's Workshop
212 - NWG Meeting on Network Usage
157 - Invitation to the Second Symposium on Problems in the Optimization of Data Communication Systems
149 - The Best Laid Plans...
130 - Response to RFC 111: Pressure from the chairman
111 - Pressure from the Chairman
048 - A Possible Protocol Plateau
046 - ARPA Network Protocol Notes

1e. Listas de distribución

402, 363, 329, 303, 300, 211, 168, 155 - ARPA Network Mailing Lists
069 - Distribution List Change for MIT
052 - Updated Distribution List

1f. Documentos sobre políticas

1603 - IETF Working Group Guidelines and Procedures
1371 - Choosing a "Common IGP" for the IP Internet (The IESG's Recommendation to the IAB)
1124 - Policy issues in interconnecting networks
1087 - Ethics and the Internet
1052 - IAB recommendations for the development of Internet network management standards
1039 - DoD statement on Open Systems Interconnection protocols
980 - Protocol Document Order Information
952, 810, 608 - Host Table Specification
945 - A DoD Statement on the NRC Report
902 - ARPA-Internet Protocol Policy
849 - Suggestions for Improved Host Table Distribution
678 - Standard file formats
602 - The Stockings Were Hung by the Chimney With Care
115 - Some Network Information Center Policies on Handling Documents
053 - An Official Protocol Mechanism

1g. Administración de solicitud de comentarios

1543, 1111 - Instructions to RFC Authors
1150 - F.Y.I. on F.Y.I.: Introduction to the F.Y.I. notes
1000 - Request For Comments reference guide
999, 899, 800, 699 - Requests for Comments Summary
825 - Request for Comments on Requests for Comments
629 - Scenario for Using the Network Journal
628 - Status of RFC Numbers and a Note on Pre-assigned Journal Numbers
598, 200, 170, 160, 100, 084 - RFC Index

1h. Otros

1718, 1539, 1391 - The Tao of IETF: A Guide for New Attendees of the Internet Engineering Task Force
1690 - Introducing the Internet Engineering and Planning Group (IEPG)
1689 - A Status Report on Networked Information Retrieval: Tools and Groups
1640 - The Process for Organization of Internet Standards Working Group (POISED)

1602, 1310 - The Internet Standards Process
1601, 1358 - I. Architecture Board (IAB)
1527 - What Should We Plan Given the Dilemma of the Network?
1481 - IAB Recommendation for an Intermediate Strategy to Address the Issue of Scaling
1438 - Internet Engineering Task Force Statements Of Boredom (SOBs)
1435 - IESG Advice from Experience with Path MTU Discovery
1401 - Correspondence between the IAB and DISA on the use of DNS *throughout the Internet*
1396 - The Process for Organization of Internet Standards Working Group (POISED)
1380 - IESG Deliberations on Routing and Addressing
1311 - Introduction to the STD Notes
1297 - NOC Internal Integrated Trouble Ticket System Functional Specification Wishlist (''NOC TT REQUIREMENTS'')
1287 - Towards the Future Internet Architecture
1272 - Internet Accounting: Background
1261 - Transition of NIC Services
1174 - IAB Recommended Policy on Distributing Internet Identifier Assignment and IAB Recommended Policy Change to Internet "Connected" Status
1166 - Internet Numbers
637 - *Change of Network Address for SU-DSL*
634 - Change in Network Address for Haskins Lab
616 - Latest Network Maps
609 - Statement of Upcoming Move of NIC/NLS Service
590 - MULTICS Address Change
588 - London Node is Now Up
551 - NYU, ANL, and LBL Joining the Net
544 - Locating On-Line Documentation at SRI-ARC
543 - Network Journal Submission and Delivery
518 - ARPANET Accounts
511 - Enterprise Phone Service to NIC From ARPANET Sites
510 - Request for Network Mailbox Addresses
440 - Scheduled network software maintenance
432 - Network Logical Map
423, 389 - UCLA Campus Computing Network Liaison Staff for ARPA Network
421 - A Software Consulting Service for Network Users
419 - MIT-DMS on Vacation
416 - The ARC System will be Unavailable for Use During Thanksgiving Week

405 - Correction to RFC 404
404 - Host Address Changes Involving Rand and ISI
403 - Desirability of a Network 1108 Service
386 - Letter to TIP Users - 2
384 - Official Site IDENTS for Organizations in the ARPA Networks
381 - Three Aids to Improved Network Operation
365 - Letter to all TIP users
356 - ARPA Network Control Center
334 - Network Use on May 8
305 - Unknown Host Numbers
301 - BBN IMP No. 5 and NCC Schedule for March 4, 1972
289 - What we hope is an official list of host names
276 - NIC Course
249 - Coordination of Equipment and Supplies Purchase
223 - Network Information Center Schedule for Network Users
185 - NIC Distribution of Manuals and Handbooks
154 - Exposition Style
136 - Host Accounting and Administrative Procedures
118 - Information Required for Each Service Available to the Network
095 - Distribution of NWG/RFC's Through the NIC
016 - MIT

2. Documentos sobre requerimientos y revisiones de protocolos mayores

2a. Requerimientos para anfitriones

1127 - Perspective on the Host Requirements RFCs
1123 - Requirements for Internet hosts - application and support
1122 - Requirements for Internet hosts - communication layers

2b. Requerimientos para compuertas

1009 - Requirements for Internet gateways

3. Nivel de interfaz de red (ver también la sección 8)

3a. Asignación de direcciones (ARP, RARP)

1735 - NBMA Address Resolution Protocol (NARP)
1433 - Directed ARP
1329 - Thoughts on Address Resolution for Dual MAC FDDI Networks
1293 - Inverse Address Resolution Protocol
1027 - Using ARP to implement transparent subnet gateways
925 - Multi-LAN Address Resolution Protocol
903 - A Reverse Address Resolution Protocol
826 - Address Resolution Protocol

3b. Protocolo Internet en otras redes (encapsulación)

1626 - Default IP MTU for use over ATM AAL5
1577 - Classical IP and ARP over ATM
1490, 1294 - Multiprotocol Interconnect over Frame Relay
1483 - Multiprotocol Encapsulation over ATM Adaptation Layer 5
1390, 1188, 1103 - Transmission of IP and ARP over FDDI Networks
1374 - IP and ARP on HIPPI
1241 - A Scheme for an Internet Encapsulation Protocol: Version 1
1226 - Internet Protocol Encapsulation of AX.25 Frames
1221, 907 - Host Access Protocol (HAP) Specification
1209 - The Transmission of IP Datagrams over the SMDS Service
1201, 1051 - Transmitting IP Traffic over ARCNET Networks
1149 - Standard for the transmission of IP datagrams on avian carriers
1088 - Standard for the transmission of IP datagrams over NetBIOS networks
1055 - Nonstandard for transmission of IP datagrams over serial lines: SLIP
1044 - Internet Protocol on Network System's HYPERchannel: Protocol specification
1042 - Standard for the transmission of IP datagrams over IEEE 802 networks
948 - Two Methods for the Transmission of IP Datagrams Over IEEE 802.3 Networks
895 - A Standard for the Transmission of IP Datagrams over Experimental Ethernet Networks
894 - A Standard for the Transmission of IP Datagrams over Ethernet Networks
893 - Trailer Encapsulations
877 - A Standard for the Transmission of IP Datagrams Over Public Data Networks

3c. Otros

1326 - Mutual Encapsulation Considered Dangerous

4. Nivel Internet

4a. Protocolo Internet (IP)

1624, 1141 - Computation of the Internet Checksum via Incremental Update
1191 - Path MTU Discovery
1190 - Experimental Internet Stream Protocol, Version 2 (ST-II)
1071 - Computing the Internet checksum
1063 - IP MTU discovery options
1025 - TCP and IP bake off
815 - IP Datagram Reassembly Algorithms

791, 760 - Internet Protocol (IP)
781 - A Specification of the Internet Protocol IP Timestamp Option

4b. Protocolo de Mensajes de Control de Internet (ICMP)

1256 - ICMP Router Discovery Messages
1018 - Some comments on SQuID
1016 - Something a host could do with source quench: The Source Quench Introduced Delay (SQuID)
792, 777 - Internet Control Message Protocol (ICMP)

4c. Protocolo de Administración de Grupos de Internet (IGMP)

1112, 1054, 988 - Host extensions for IP multicasting

4d. Algoritmos de porteo y ruteo (BGP, GGP, RIP, OSPF)

1745 - BGP4/IDRP for IP---OSPF Interaction
1723, 1388 - RIP Version 2 Carrying Additional Information
1722 - RIP Version 2 Protocol Applicability Statement
1721, 1387 - RIP Version 2 Protocol Analysis
1702 - Generic Routing Encapsulation over IPv4 networks
1701 - Generic Routing Encapsulation (GRE)
1656 - BGP-4 Protocol Document Roadmap and Implementation Experience
1655, 1268, 1164 - Application of the Border Gateway Protocol in the Internet
1654 - A Border Gateway Protocol 4 (BGP-4)
1587 - The OSPF NSSA Option
1586 - Guidelines for Running OSPF Over Frame Relay Networks
1585 - MOSPF: Analysis and Experience
1584 - Multicast Extensions to OSPF
1583, 1247, 1131 - OSPF Version 2
1582 - Extensions to RIP to Support Demand Circuits
1581 - Protocol Analysis for Extensions to RIP to Support Demand Circuits
1520 - Exchanging Routing Information Across Provider Boundaries in the CIDR Environment
1519, 1338 - Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy
1517 - Applicability Statement for the Implementation of Classless Inter-Domain Routing (CIDR)
1504 - Appletalk Update-Based Routing Protocol: Enhanced Appletalk Routing
1482 - Aggregation Support in the NSFNET Policy Routing Database
1479 - Inter-Domain Policy Routing Protocol Specification: Version 1
1478 - An Architecture for Inter-Domain Policy Routing
1477 - IDPR as a Proposed Standard

1465 - Routing coordination for X.400 MHS services within a multi protocol / multi network environment Table Format V3 for static routing
1403, 1364 - BGP OSPF Interaction
1397 - Default Route Advertisement In BGP2 And BGP3 Versions Of The Border Gateway Protocol
1383 - An Experiment in DNS Based IP Routing
1370 - Applicability Statement for OSPF
1322 - A Unified Approach to Inter-Domain Routing
1267, 1163 - A Border Gateway Protocol 3 (BGP-3)
1266 - Experience with the BGP Protocol
1265 - BGP Protocol Analysis
1264 - Internet Routing Protocol Standardization Criteria
1254 - Gateway Congestion Control Survey
1246 - Experience with the OSPF Protocol
1245 - OSPF Protocol Analysis
1222 - Advancing the NSFNET Routing Architecture
1195 - Use of OSI IS-IS for Routing in TCP/IP and Dual Environments
1142 - OSI IS-IS Intra-domain Routing Protocol
1136 - Administrative Domains and Routing Domains: A model for routing in the Internet
1133 - Routing between the NSFNET and the DDN
1126 - Goals and functional requirements for inter-autonomous system routing
1125 - Policy requirements for inter Administrative Domain routing
1105 - Border Gateway Protocol (BGP)
1104 - Models of policy based routing
1102 - Policy routing in Internet protocols
1093 - NSFNET routing architecture
1092 - EGP and policy based routing in the new NSFNET backbone
1075 - Distance Vector Multicast Routing Protocol
1074 - NSFNET backbone SPF based Interior Gateway Protocol
1058 - Routing Information Protocol
1046 - Queuing algorithm to provide type-of-service for IP links
985 - Requirements for Internet Gateways
975 - Autonomous Confederations
970 - On Packet Switches With Infinite Storage
911 - EGP Gateway under Berkeley Unix
904, 890, 888, 827 - Exterior Gateway Protocol
875 - Gateways, Architectures, and Heffalumps
823 - Gateway Gateway Protocol

4e. IP de la próxima generación (IPng)

1753 - IPng Technical Requirements Of the Nimrod Routing and Addressing Architecture
1752 - The Recommendation for the IP Next Generation Protocol
1726 - Technical Criteria for Choosing IP:The Next Generation (IPng)
1710 - Simple Internet Protocol Plus White Paper
1707 - CATNIP: Common Architecture for the Internet
1705 - Six Virtual Inches to the Left: The Problem with IPng
1688 - IPng Mobility Considerations
1687 - A Large Corporate User's View of IPng
1686 - IPng Requirements: A Cable Television Industry Viewpoint
1683 - Multiprotocol Interoperability In IPng
1682 - IPng BSD Host Implementation Analysis
1680 - IPng Support for ATM Services
1679 - PN Working Group Input to the IPng Requirements Solicitation
1678 - IPng Requirements of Large Corporate Networks
1677 - Tactical Radio Frequency Communication Requirments for IPng
1676 - INFN Requirements for an IPng
1675 - Security Concerns for IPng
1674 - A Cellular Industry View of IPng
1673 - Electric Power Research Institute Comments on IPng
1672 - Accounting Requirements for IPng
1671 - IPng White Paper on Transition and Other Considerations
1670 - Input to IPng Engineering Considerations
1669 - Market Viability as a IPng Criteria
1668 - Unified Routing Requirements for IPng
1667 - Modeling and Simulation Requirements for IPng
1622 - Pip Header Processing
1621 - Pip Near-term Architecture
1606 - A Historical Perspective On The Usage Of IP Version 9
1550 - IP: Next Generation (IPng) White Paper Solicitation
1526 - Assignment of System Identifiers for TUBA/CLNP Hosts
1475 - TP/IX: The Next Internet
1454 - Comparison of Proposals for Next Version of IP
1385 - EIP: The Extended Internet Protocol A Framework for Maintaining Backward Compatibility
1375 - Suggestion for New Classes of IP Addresses
1365 - An IP Address Extension Proposal
1347 - TCP and UDP with Bigger Addresses (TUBA), A Simple Proposal for Internet Addressing and Routing
1335 - A Two-Tier Address Structure for the Internet: A Solution to the Problem of Address Space Exhaustion

4f. Otros

1744 - Observations on the Management of the Internet Address Space
1716 - Towards Requirements for IP Routers
1715 - The H Ratio for Address Assignment Efficiency
1631 - The IP Network Address Translator (Nat)
1620 - Internet Architecture Extensions for Shared Media
1597 - Address Allocation for Private Internets
1560 - The MultiProtocol Internet
1518 - An Architecture for IP Address Allocation with CIDR
1476 - RAP: Internet Route Access Protocol
1467, 1367 - Schedule for IP Address Space Management Guidelines
1466, 1366 - Guidelines for Management of IP Address Space
1393 - Traceroute Using an IP Option
1363 - A Proposed Flow Specification
1349 - Type of Service in the Internet Protocol Suite
1219 - On the Assignment of Subnet Number
986 - Working Draft - Guidelines for the Use of Internet-IP Addressing in the ISO Connectionless-Mode Network
981 - An Experimental Multiple-Path Routing Algorithm
963 - Some Problems with the Specification of the Military Standard Internet Protocol
950 - Internet Standard Subnetting Procedure
947 - Multi-Network Broadcasting Within the Internet
940, 917, 932, 936 - Internet Subnets Protocol
922, 919 - Broadcasting Internet datagrams in the presence of subnets
871 - A Perspective on the ARPANET Reference Model
831 - Backup Access to the European Side of SATNET
817 - Modularity and Efficiency in Protocol Implementation
816 - Fault Isolation and Recovery
814 - Name, Addresses, Ports, and Routes
796 - Address Mapping
795 - Service Mappings
730 - Extensible Field Addressing

5. Nivel anfitrión

5a. Protocolo de datagramas de usuario (UDP)

768 - User Datagram Protocol

5b. Protocolo de control de transmisión (TCP)

1644 - T/TCP -- TCP Extensions for Transactions Functional Specification
1379 - Extending TCP for Transactions -- Concepts
1337 - TIME-WAIT Assassination Hazards in TCP
1323, 1185 - TCP Extensions for High Performance
1263 - TCP Extensions Considered Harmful
1146, 1145 - TCP alternate checksum options
1144 - Compressing TCP/IP headers for low-speed serial links
1110 - Problem with the TCP big window option
1106 - TCP big window and NAK options
1078 - TCP port service Multiplexer (TCPMUX)
1072 - TCP extensions for long-delay paths
983 - ISO Transport Services on Top of the TCP
964 - Some Problems with the Specification of the Military Standard Transmission Control Protocol
962 - TCP-4 prime
896 - Congestion Control in IP/TCP Internetworks
889 - Internet Delay Experiments
879 - The TCP Maximum Segment Size and Related Topics
872 - TCP-ON-A-LAN
813 - Window and acknowlegement strategy in TCP
794 - Pre-Emption
793, 761, 675 - Transmission Control Protocol
721 - Out of Band Control Signals in a Host to Host Protocol
700 - A Protocol Experiment

5c. Protocolos punto a punto

1717 - The PPP Multilink Protocol (MP)
1663 - PPP Reliable Transmission
1662, 1549 - PPP in HDLC Framing
1661, 1548 - The Point-to-Point Protocol (PPP)
1638, 1220 - Point-to-Point Protocol Extensions for Bridging
1619 - PPP over SONET/SDH
1618 - PPP over ISDN
1598 - PPP in X.25
1570 - PPP LCP Extensions
1552 - The PPP Internetwork Packet Exchange Control Protocol (IPXCP)
1547 - Requirements for an Internet Standard Point-to-Point Protocol
1378 - The PPP AppleTalk Control Protocol (ATCP)
1377 - The PPP OSI Network Layer Control Protocol (OSINLCP)
1376 - The PPP DECnet Phase IV Control Protocol (DNCP)

1334 - PPP Authentication Protocols
1333 - PPP Link Quality Monitoring
1332, 1172 - The Point-to-Point Protocol (PPP) Initial Configuration Options
1331, 1171, 1134 - The Point-to-Point Protocol for the Transmission of Multi-Protocol Datagrams Over Point-to-Point Links

5d. Protocolos de datagramas confiables (RDP, VMTP)

1151, 908 - Reliable Data Protocol (RDP)
1045 - VMTP: Versatile Message Transaction Protocol: Protocol specification

5e. Protocolos de transacción y sistemas operativos distribuidos

955 - Towards a Transport Service for Transaction Processing Applications
938 - Internet Reliable Transaction Protocol Functional and Interface Specification
722 - Thoughts on Interactions in Distributed Services
713 - MSDTP -- Message Services Data Transmission Protocol
712 - A Distributed Capability Computing System DCCS
708 - Elements of a Distributed Programming System
707 - A High-Level Framework for Network-Based Resource Sharing
684 - A Commentary on Procedure Calling as A Network Protocol
677 - The Maintenance of Duplicate Databases
674 - Procedure Call Documents--Version 2
672 - A Multi-Site Data Collection Facility
671 - A Note on Reconnection Protocol
645 - Network Standard Data Specification Syntax
615 - Proposed Network Standard Data Pathname Syntax
610 - Further Datalanguage Design Concepts
592 - Some Thoughts on System Design to Facilitate Resource Sharing
578 - Using MIT-MATHLAB MACSYMA From MIT-DMS Muddle - An Experiment in Automated Resource Sharing
515 - Specifications for Datalanguage, Version 0/9
500 - The Integration of Data Management Systems on a Computer Network
441 - Inter-Entity Communication - An Experiment
437 - *Data Reconfiguration Service at UCSB*
203 - Achieving Reliable Communication
076 - Connection-by-Name: User-Oriented Protocol
062 - A System for Interprocess Communication in a Resource Sharing Computer Network
061 - A Note on Interprocess Communication in a Resource Sharing Computer Network
051 - Proposal for a Network Interchange Language
031 - Binary Message Forms in Computer Networks

5f. Protocolos para computadoras personales (NETBIOS)

1002 - Protocol standard for a NetBIOS service on a TCP/UDP transport: Detailed specifications
1001 - Protocol standard for a NetBIOS service on a TCP/UDP transport: Concepts and methods

5g. Otros

1469 - IP Multicast over Token-Ring Local Area Networks
1458 - Requirements for Multicast Protocols
1312, 1159 - Message Send Protocol
1301 - Multicast Transport Protocol
998, 969 - NETBLT: A Bulk Data Transfer Protocol
979 - PSN End-to-End Functional Specification
966 - A Multicast Extension to the Internet Protocol
869 - Host Monitoring Protocol
741 - Specifications for the Network Voice Protocol NVP
643 - Cross Net Debugger
162 - NETBUGGER3

6. Nivel de aplicación

6a. Protocolo telnet (TELNET)

1647 - TN3270 Enhancements
1646 - TN3270 Extensions for LUname and Printer Selection
1576 - TN3270 Current Practices
1205 - 5250 Telnet Interface
1184 - Telnet Linemode Option
854, 764 - Telnet Protocol Specification
818 - The Remote User Telnet Service
782 - A Virtual Terminal Management Model
728 - A Minor Pitfall in the Telnet Protocol
703, 702, 701, 679, 669 - Survey of New-Protocol Telnet Servers
688 - Tentative Schedule for the New Telnet Implementation for the TIP
681 - Network Unix
600 - Interfacing an Illinois Plasma Terminal to the ARPANET
596 - Second Thoughts on Telnet Go-Ahead
595 - Some Thoughts in Defense of the Telnet Go-Ahead
593 - Telnet and FTP Implementation Schedule Change
576 - Proposal for Modifying Linking
570 - Experimental Input Mapping Between NVT ASCII and UCSB Online System

562 - Modifications to the Telnet Specification
559 - Comments on the New Telnet Protocol and Its Implementation
529 - A Note on Protocol Synch Sequences
513 - Comments on the New Telnet Specifications
495 - Telnet Protocol Specification
466 - Telnet Logger/Server for Host LL-67
452 - Telnet Command at Host LL
435 - Telnet Issues
426 - Reconnection Protocol
393 - Comments on Telnet Protocol Changes
377 - Using TSO Via ARPA Network Virtual Terminal
357 - An Echoing Strategy for Satellite Links
355, 346 - Satellite Considerations
340 - Proposed Telnet Changes
339 - MLTNET - A "Multi-Telnet" Subsystem for TENEX
328 - Suggested Telnet Protocol Changes
318 - Ad Hoc Telnet Protocol
216 - Telnet Access to UCSB's On-Line System
215 - NCP, ICP, and Telnet: The Terminal IMP Implementation
206 - A User Telnet Description of an Initial Implementation
205 - NETCRT - A Character Display Protocol
190 - DEC PDP-10 - IMLAC Communication System
158 - Proposed Telnet Protocol
139 - Discussion of Telnet Protocol
137 - Telnet Protocol - A Proposed Document
135, 110 - Conventions for Using an IBM 2741 Terminal as a User Console for Access to Network Server Hosts
103 - Implementation of Interrupt Keys
097 - A First Cut at a Proposed Telnet Protocol
091 - A Proposed User-User Protocol

6b. Opciones Telnet

1572, 1408 - Telnet Environment Option
1571 - Telnet Environment Option Interoperability Issues
1416, 1409 - Telnet Authentication Option
1412 - Telnet Authentication : SPX
1411 - Telnet Authentication: Kerberos Version 4
1372, 1080 - Telnet remote flow control option
1143 - Q method of implementing Telnet option negotiation
1116 - Telnet Linemode option
1097 - Telnet subliminal-message option

1096 - Telnet X display location option
1091 - Telnet terminal-type option
1079 - Telnet terminal speed option
1073 - Telnet window size option
1053 - Telnet X.3 PAD option
1043 - Telnet Data Entry Terminal option: DODIIS implementation
1041 - Telnet 3270 regime option
946 - Telnet Terminal Location Number Option
933 - Output Marking Telnet Option
930 - Telnet Terminal Type Option
927 - TACACS User Identification Telnet Option
885 - Telnet End of Record Option
884 - Telnet Terminal Type Option
861 - Telnet Extended Options - List Option
860 - Telnet Timing Mark Option
859 - Telnet Status Option
858 - Telnet Suppress Go Ahead Option
857 - Telnet Echo Option
856 - Telnet Binary Transmission
855 - Telnet Option Specifications
779 - Telnet Send-Location Option
749 - Telnet SUPDUP-OUTPUT Option
748 - Telnet Randomly-Lose Option
736 - Telnet SUPDUP Option
735 - Revised Telnet Byte Macro Option
747 - Recent Extensions to the SUPDUP Protocol
746 - The SUPDUP Graphics Extension
732 - Telnet Data Entry Terminal Option
731 - Telnet Data Entry Terminal Option
729 - Telnet Byte Macro Option
727 - Telnet Logout Option
726 - Remote Controlled Transmission and Echoing Telnet Option
719 - Discussion on RCTE
718 - Comments on RCTE from the Tenex Implementation Experience
698 - Telnet Extended ASCII Option
659 - Announcing Additional Telnet Options
658 - Telnet Output Line Feed Disposition
657 - Telnet Output Vertical Tab Disposition Option
656 - Telnet Output Vertical Tab Stops Option
655 - Telnet Output Form Feed Disposition Option
654 - Telnet Output Horizontal Tab Disposition Option

653 - Telnet Output Horizontal Tab Stops Option
652 - Telnet Output Carriage Return Disposition Option
651 - Revised Telnet Status Option
587 - Announcing New Telnet Options
581 - Corrections to RFC 560 - Remote Controlled Transmission and Echoing Telnet Option
563 - Comments on the RCTE Telnet Option
560 - Remote Controlled Transmission and Echoing Telnet Option

6c. Protocolos de acceso y transferencia de archivos

1639, 1545 - FTP Operation Over Big Address Records (FOOBAR)
1635 - How to Use Anonymous FTP
1579 - Firewall-Friendly FTP
1440 - SIFT/UFT: Sender-Initiated/Unsolicited File Transfer
1415 - FTP-FTAM Gateway Specification
1350, 783 - The TFTP Protocol Revision 2
1282, 1258 - BSD Rlogin
1235 - The Coherent File Distribution Protocol
1094 - NFS: Network File System Protocol specification
1068 - Background File Transfer Program (BFTP)
1037 - NFILE - a file access protocol
959, 765, 542, 354, 265, 172, 114 - The File Transfer Protocol
949 - FTP Unique-Named Store Command
913 - Simple File Transfer Protocol
906 - Bootstrap Loading Using TFTP
775 - Directory Oriented FTP Commands
743 - FTP Extension: XRSQ/XRCP
737 - FTP Extension: XSEN
697 - CWD Command of FTP
691 - One More Try on the FTP
686 - Leaving Well Enough Alone
683 - FTPSRV -- Tenex Extension for Paged Files
662 - Performance Improvement in ARPANET File Transfers from Multics
640 - Revised FTP Reply Codes
630 - FTP Error Code Usage for More Reliable Mail Service
624 - Comments on the File Transfer Protocol
614 - Response to RFC 607 - Comments on the FTP
607 - NIC-21255 Comments on the File Transfer Protocol
571 - Tenex FTP Problem
535 - Comments on File Access Protocol
532 - The UCSD-CC Server-FTP Facility

520 - Memo to FTP Group (Proposal for File Access Protocol)
506 - An FTP Command Naming Problem
505 - Two Solutions to a File Transfer Access Problem
501 - Un-Muddling "Free File Transfer"
487 - Host-Dependent FTP Parameters
486 - Data Transfer Revisited
480 - Host-Dependent FTP Parameters
479 - Use of FTP by the NIC Journal
478 - FTP Server-Server Interaction - II
468 - FTP Data Compression
463 - FTP Comments and Response to RFC 430
448 - Print Files in FTP
438 - FTP Server-Server Interaction
430 - Comments on File Transfer Protocol
418 - Server File Transfer Under TSS/360 at NASA/Ames Research Center
414 - File Transfer Protocols (FTP): Status and Further Comments
412 - User FTP Documentation
385 - Comments on the File Transfer Protocol (RFC 354) 310 - Another Look at Data and File Transfer Protocols
294 - The Use of "Set Data Type" Transaction in the File Transfer Protocol
281 - A Suggested Addition to File Transfer Protocol
269 - Some Experience with File Transfer
264, 171 - The Data Transfer Protocol
250 - Some Thoughts on File Transfer
242 - Data Descriptive Language for Shared Data
238 - Comments on DTP and FTP Protocols
163 - Data Transfer Protocols
141 - Comments on RFC 114 (A File Transfer Protocol)
133 - File Transfer and Error Recovery

6d. Sistema de Nombres de Dominio (DNS)

1713 - Tools for DNS debugging
1712 - DNS Encoding of Geographical Location
1706, 1637, 1348 - DNS NSAP Resource Records
1591 - Domain Name System Structure and Delegation
1537 - Common DNS Data File Configuration Error
1536 - Common DNS Implementation Errors and Suggested Fixes
1535 - A Security Problem and Proposed Correction With Widely Deployed DNS Software
1480, 1386 - The US Domain
1464 - Using the Domain Name System To Store Arbitrary String Attributes

1394 - Relationship of Telex Answerback Codes to Internet Domains
1183 - New DNS RR Definitions
1101 - DNS encoding of network names and other types
1035 - Domain names - implementation and specification
1034 - Domain names - concepts and facilities
1033 - Domain administrators operations guide
1032 - Domain administrators guide
1031 - MILNET name domain transition
973 - Domain System Changes and Observations
953, 811 - Hostname Server
921, 897 - Domain Name System Implementation Schedule
920 - Domain Requirements
883 - Domain Names - Implementation and Specification
882 - Domain Names - Concepts and Facilities
881 - The Domain Names Plan and Schedule
830 - A Distributed System for Internet Name Service
819 - The Domain Naming Convention for Internet User Applications
799 - Internet Name Domains
756 - The NIC Name Server – A Datagram-Based Information Utility
752 - A Universal Host Table

6e Sistema de mensajes y correo (SMTP, MIME, X.400)

1741 - MIME Content Type for BinHex Encoded Files
1740 - MIME Encapsulation of Macintosh files - MacMIME
1734 - POP3 AUTHentication command
1733 - DISTRIBUTED ELECTRONIC MAIL MODELS IN IMAP4
1732 - IMAP4 COMPATIBILITY WITH IMAP2 AND IMAP2BIS
1731 - IMAP4 Authentication mechanisms
1730 - INTERNET MESSAGE ACCESS PROTOCOL (IMAP) - VERSION 4
1725, 1460, 1225, 1082, 1081 - Post Office Protocol - version 3
1711 - Classifications in E-mail Routing
1685 - Writing X.400 O/R Names
1664 - Using the Internet DNS to Distribute RFC1327 Mail Address Mapping Tables
1653, 1427 - SMTP Service Extension for Message Size Declaration
1652, 1426 - SMTP Service Extension for 8bit-MIMEtransport
1651, 1425 - SMTP Service Extensions
1649 - Operational Requirements for X.400 Management Domains in the GO-MHS Community
1648 - Postmaster Convention for X.400 Operations
1642 - UTF-7 - A Mail-Safe Transformation Format of Unicode

1641 - Using Unicode with MIME
1616 - X.400(1988) for the Academic and Research Community in Europe
1615 - Migrating from X.400(84) to X.400(88)
1590 - Media Type Registration Procedure
1563, 1523 - The text/enriched MIME Content-type
1557 - Korean Character Encoding for Internet Messages
1556 - Handling of Bi-directional Texts in MIME
1555 - Hebrew Character Encoding for Internet Messages
1544 - The Content-MD5 Header Field
1524 - A User Agent Configuration Mechanism For Multimedia Mail Format Information
1522, 1342 - Representation of Non-ASCII Text in Internet Message Headers
1521, 1341 - MIME (Multipurpose Internet Mail Extensions): Mechanisms for Specifying and Describing the Format of Internet Message Bodies
1506 - A tutorial on gatewaying between X.400 and Internet mail
1505, 1154 - Encoding Header Field for Internet Messages
1502 - X.400 Use of Extended Character Sets
1496 - Rules for downgrading messages from X.400/88 to X.400/84 when MIME content-types are present in the messages
1495, 1327, 1148, 1138 - Mapping between X.400(1988) / ISO 10021 and RFC 822
1494 - Equivalences between 1988 X.400 and RFC-822 Message Bodies
1437 - The Extension of MIME Content-Types to a New Medium
1428 - Transition of Internet Mail from Just-Send-8 to 8Bit-SMTP/MIME
1405 - Mapping between X.400(1984/1988) and Mail-11 (DECnet mail)
1357 - A Format for E-mailing Bibliographic Records
1344 - Implications of MIME for Internet Mail Gateways
1343 - A User Agent Configuration Mechanism For Multimedia Mail Format Information
1339 - Remote Mail Checking Protocol
1328 - X.400 1988 to 1984 downgrading
1211 - Problems with the Maintenance of Large Mailing Lists
1204 - Message Posting Protocol (MPP)
1203, 1176, 1064 - Interactive Mail Access Protocol: Version 2
1168 - Intermail and Commercial Mail Relay Services
1153 - Digest message format
1137 - Mapping between full RFC 822 and RFC 822 with restricted encoding
1090 - SMTP on X.25
1056, 993, 984 - PCMAIL: A distributed mail system for personal computers
1049 - Content-type header field for Internet messages
1047 - Duplicate messages and SMTP

1026 - Addendum to RFC 987: (Mapping between X.400 and RFC-822)
987 - Mapping Between X.400 and RFC 822
977 - Network News Transfer Protocol
976 - UUCP Mail Interchange Format Standard
974 - Mail Routing and the Domain System
934 - Proposed Standard for Message Encapsulation
915 - Network Mail Path Service
886 - Proposed Standard for Message Header Munging
850 - Standard for Interchange of USENET Messages
841 - Specification for Message Format for Computer Based Message Systems
822 - Standard for the Format of ARPA Internet Text Messages
821, 788 - Simple Mail Transfer Protocol
806 - Specification for Message Format for Computer Based Message Systems
780, 772 - Mail Transfer Protocol
786 - Mail Transfer Protocol - ISI TOPS-20 MTP-NIMAIL Interface
785 - Mail Transfer Protocol - ISI TOPS-20 File Definitions
784 - Mail Transfer Protocol - ISI TOPS-20 Implementation
771 - Mail Transition Plan
763 - Role Mailboxes
757 - A Suggested Solution to the Naming, Addressing, and Delivery Problem for ARPANET Message Systems
754 - Out-of-Net Host Addresses for Mail
753 - Internet Message Protocol
751 - Survey of FTP Mail and MLFL
744 - MARS - a Message Archiving and Retrieval Service
733 - Standard for the Format of ARPA Network Text Messages
724 - Proposed Official Standard for the Format of ARPA Network Messages
720 - Address Specification Syntax for Network Mail
706 - On the Junk Mail Problem
680 - Message Transmission Protocol
644 - On the Problem of Signature Authentication for Network Mail
577 - Mail Priority
574 - Announcement of a Mail Facility at UCSB
561 - Standardizing Network Mail Headers
555 - Responses to Critiques of the Proposed Mail Protocol
539, 524 - A Proposed Mail Protocol
498 - On Mail Service to CCN
491 - What is "Free"?
475 - FTP and the Network Mail System

458 - Mail Retrieval via FTP
333 - A Proposed Experiment with a Message Switching Protocol
278, 224, 221, 196 - A Mail Box Protocol

6f. Faxs y mapas de bits

809 - UCL Facsimile System
804 - Facsimile Formats
803 - Dacom 450/500 Facsimile Date Transcoding
798 - Decoding Facsimile Data From the Rapicom 450
797 - Bitmap Formats
769 - Rapicom 450 Facsimile File Format

6g. Gráficas y Sistemas Window

1198 - FYI on the X Window System
1013 - X Window System Protocol, version 11: Alpha update April 1987
965 - A Format for a Graphical Communication Protocol
553 - Draft Design for a Text/Graphics Protocol
493 - Graphics Protocol
401 - Conversion of NGP-0 Coordinates to Device Specific Coordinates
398 - UCSB Online Graphics
387 - Some Experiences in Implementing Network Graphics Protocol Level 0
351 - Information Form for the ARPANET Graphics Resources Notebook
336 - Level 0 Graphics Input Protocol
296 - DS-1 Display System
292 - Graphics Protocol - Level 0 only
285 - Network Graphics
268 - Graphics Facilities Information
199 - Suggestions for a Network Data-Telnet Graphics Protocol
192 - Some Factors Which a Network Graphics Protocol Must Consider
191 - Graphics Implementation and Conceptualization at ARC
186 - A Network Graphics Loader
184 - Proposed Graphic Display Modes
181, 177 - A Device Independent Graphical Display Description
178 - Network Graphics Attention Handling
125, 086 - Proposal for a Network Standard Format for a Data Stream to Control Graphics Display
094 - Some Thoughts on Network Graphics

6h. Administración de datos

304 - A Data Management System Proposal for the ARPA Network
195 - Data Computers - Data Descriptions and Access Language

194 - The Data Reconfiguration Service - Compiler/Interpreter Implementation Notes
166 - Data Reconfiguration Service - An Implementation Specification
144 - Data Sharing on Computer Networks
138 - Status Report on Proposed Data Reconfiguration Service
083 - Language-Machine for Data Reconfiguration

6i. Entrada de trabajos a distancia (NETRJE, NETRJS)

740, 599, 589, 325, 189, 088 - CCN Network Remote Job Entry Program - NETRJS
725 - An RJE Protocol for a Resource Sharing Network
499 - Harvard's Network RJE
490 - Surrogate RJS for UCLA-CCN
477, 436 - Remote Job Service at UCSB
407 - Remote Job Entry
368 - Comments on "Proposed Remote Job Entry Protocol"
360 - Proposed Remote Job Entry Protocol
338 - EBCDIC/ASCII Mapping for Network RJE
307 - Using Network Remote Job Entry
283 - NETRJT - Remote Job Service Protocol for TIPS
105 - Network Specification for Remote Job Entry and Remote Job Output Retrieval at UCSB

6j. Llamada a procedimiento remoto (RPC)

1057 - RPC: Remote Procedure Call Protocol specification version 2
1050 - RPC: Remote Procedure Call Protocol specification

6k. Hora y fecha (NTP)

1708 - NTP PICS PROFORMA For the Network Time Protocol Version 3
1589 - A Kernel Model for Precision Timekeeping
1361 - Simple Network Time Protocol (SNTP)
1305, 1119 - Network Time Protocol
1165 - Network Time Protocol (NTP) over the OSI Remote Operations Service
1129 - Internet time synchronization: The Network Time Protocol
1128 - Measured performance of the Network Time Protocol in the Internet system
1059 - Network Time Protocol (version 1) specification and implementation
958, 957, 956 - Network Time Protocol
868 - Time Server Protocol
867 - Daytime Protocol
778 - DCNET Time Server Protocol

738 - Time Server
685 - Response Time in Cross-network Debugging
034 - Some Brief Preliminary Notes on the ARC Clock
032 - Some Thoughts on SRI's Proposed Real Time Clock
028 - Time Standards

6l. Presentación y representación (XDR)

1489 - Registration of a Cyrillic Character Set
1468 - Japanese Character Encoding for Internet Messages
1456 - Conventions for Encoding the Vietnamese Language VISCII: VIetnamese Standard Code for Information Interchange VIQR: VIetnamese Quoted-Readable Specification
1314 - A File Format for the Exchange of Images in the Internet
1278 - A String Encoding of Presentation Address
1197 - Using ODA for Translating Multimedia Information
1014 - XDR: External Data Representation standard
1003 - Issues in defining an equations representation standard

6m. Administración de red (SNMP, CMOT, MIB)

1749 - IEEE 802.5 Station Source Routing MIB using SMIv2
1748, 1743, 1231 - IEEE 802.5 MIB using SMIv2
1742, 1243 - AppleTalk Management Information Base II
1724, 1389 - RIP Version 2 MIB Extension
1697 - Relational Database Management System (RDBMS) Management Information Base (MIB) using SMIv2
1696 - Modem Management Information Base (MIB) using SMIv2
1695 - Definitions of Managed Objects for ATM Management Version 8.0 using SMIv2
1694, 1304 - Definitions of Managed Objects for the SIP Interface Type
1666 - Definitions of Managed Objects for SNA NAUs using SMIv2
1665 - Definitions of Managed Objects for SNA NAUs using SMIv2
1660, 1318 - Definitions of Managed Objects for Parallel-printer-like Hardware Devices
1659, 1317 - Definitions of Managed Objects for RS-232-like Hardware Devices
1658, 1316 - Definitions of Managed Objects for Character Stream Devices
1657 - Definitions of Managed Objects for the Fourth Version of the Border Gateway Protocol (BGP-4) using SMIv2
1650 - Definitions of Managed Objects for the Ethernet-like Interface Types using SMIv2
1643, 1623, 1398, 1284 - Definitions of Managed Objects for the Ethernet-like Interface Types

1628 - UPS Management Information Base
1612 - DNS Resolver MIB Extensions
1611 - DNS Server MIB Extensions
1604, 1596 - Definitions of Managed Objects for Frame Relay Service
1595 - Definitions of Managed Objects for the SONET/SDH Interface Type
1592, 1228 - SNMP-DPI - Simple Network Management Protocol Distributed Program Interface
1593 - SNA APPN Node MIB
1573, 1229 - Extensions to the Generic-Interface MIB
1567 - X.500 Directory Monitoring MIB
1566 - Mail Monitoring MIB
1565 - Network Services Monitoring MIB
1559, 1289 - DECnet Phase IV MIB Extensions
1525, 1493, 1286 - Definitions of Managed Objects for Bridges
1516, 1368 - Definitions of Managed Objects for IEEE 802.3 Repeater Devices
1515 - Definitions of Managed Objects for IEEE 802.3 Medium Attachment Units (MAUs)
1514 - Host Resources MIB
1513 - Token Ring Extensions to the Remote Network Monitoring MIB
1512, 1285 - FDDI Management Information Base
1503 - Algorithms for Automating Administration in SNMPv2 Managers
1474 - The Definitions of Managed Objects for the Bridge Network Control Protocol of the Point-to-Point Protocol
1473 - The Definitions of Managed Objects for the IP Network Control Protocol of the Point-to-Point Protocol
1472 - The Definitions of Managed Objects for the Security Protocols of the Point-to-Point Protocol
1471 - The Definitions of Managed Objects for the Link Control Protocol of the Point-to-Point Protocol
1461 - SNMP MIB extension for MultiProtocol Interconnect over X.25
1452 - Coexistence between version 1 and version 2 of the Internet-standard Network Management Framework
1451 - Manager to Manager Management Information Base
1450 - Management Information Base for version 2 of the Simple Network Management Protocol (SNMPv2)
1449 - Transport Mappings for version 2 of the Simple Network Management Protocol (SNMPv2)
1448 - Protocol Operations for version 2 of the Simple Network Management Protocol (SNMPv2)
1447 - Party MIB for version 2 of the Simple Network Management Protocol (SNMPv2)

1446 - Security Protocols for version 2 of the Simple Network Management Protocol (SNMPv2)
1445 - Administrative Model for version 2 of the Simple Network Management Protocol (SNMPv2)
1444 - Conformance Statements for version 2 of the Simple Network Management Protocol (SNMPv2)
1443 - Textual Conventions for version 2 of the Simple Network Management Protocol (SNMPv2)
1442 - Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2)
1441 - Introduction to version 2 of the Internet-standard Network Management Framework
1420, 1298 - SNMP over IPX
1419 - SNMP over AppleTalk
1414 - Ident MIB
1407, 1233 - Definitions of Managed Objects for the DS3/E3 Interface Type
1406, 1232 - Definitions of Managed Objects for the DS1 Interface Type
1382 - SNMP MIB Extension for the X.25 Packet Layer
1381 - SNMP MIB Extension for X.25 LAPB
1369 - Implementation Notes and Experience for The Internet Ethernet MIB
1354 - IP Forwarding Table MIB
1353 - Definitions of Managed Objects for Administration of SNMP Parties
1352 - SNMP Security Protocols
1351 - SNMP Administrative Model
1346 - Resource Allocation, Control, and Accounting for the Use of Network Resources
1315 - Management Information Base for Frame Relay DTEs
1303 - A Convention for Describing SNMP-based Agents
1271 - Remote Network Monitoring Management Information Base
1270 - SNMP Communications Services
1269 - Definitions of Managed Objects for the Border Gateway Protocol (Version 3)
1253, 1252, 1248 - OSPF Version 2 Management Information Base
1239 - Reassignment of Experimental MIBs to Standard MIBs
1230 - IEEE 802.4 Token Bus MIB
1227 - SNMP MUX Protocol and MIB
1224 - Techniques for Managing Asynchronously Generated Alerts
1215 - A Convention for Defining Traps for use with the SNMP
1214 - OSI Internet Management: Management Information Base
1213, 1158, 1156, 1066 - Management Information Base for network management of TCP/IP-based internets

1212 - Concise MIB Definitions
1189, 1095 - Common Management Information Services and Protocol over TCP/IP (CMOT)
1187 - Bulk Table Retrieval with the SNMP
1418, 1283, 1161 - SNMP over OSI
1157, 1098, 1067 - Simple Network Management Protocol (SNMP)
1109 - Report of the second Ad Hoc Network Management Review Group
1089 - SNMP over Ethernet
1076 - HEMS monitoring and control language
1155, 1065 - Structure and identification of management information for TCP/IP-based internets
1028 - Simple Gateway Monitoring Protocol
1024 - HEMS variable definitions
1023 - HEMS monitoring and control language
1022 - High-level Entity Management Protocol (HEMP)
1021 - High-level Entity Management System (HEMS)

6n. Servicios de directorio (X.500)

1684 - Introduction to White Pages services based on X.500
1632, 1292 - A Catalog of Available X.500 Implementations
1617 - Naming and Structuring Guidelines for X.500 Directory Pilots
1609 - Charting Networks in the X.500 Directory
1608 - Representing IP Information in the X.500 Directory
1564 - DSA Metrics (OSI-DS 34 (v3))
1562 - Naming Guidelines for the AARNet X.500 Directory Service
1558 - A String Representation of LDAP Search Filters
1491 - A Survey of Advanced Usages of X.500
1488 - The X.500 String Representation of Standard Attribute Syntaxes
1487 - X.500 Lightweight Directory Access Protocol
1485 - A String Representation of Distinguished Names (OSI-DS 23 (v5))
1484 - Using the OSI Directory to achieve User Friendly Naming (OSI-DS 24 (v1.2))
1431 - DUA Metrics
1430 - A Strategic Plan for Deploying an Internet X.500 Directory Service
1384 - Naming Guidelines for Directory Pilots
1373 - PORTABLE DUAs
1309 - Technical Overview of Directory Services Using the X.500 Protocol
1308 - Executive Introduction to Directory Services Using the X.500 Protocol
1279 - X.500 and Domains
1277 - Encoding Network Addresses to Support Operation Over Non-OSI Lower Layers

1276 - Replication and Distributed Operations extensions to provide an Internet Directory using X.500
1275 - Replication Requirements to provide an Internet Directory using X.500
1274 - The COSINE and Internet X.500 Schema
1255, 1218 - A Naming Scheme for c=US
1249 - DIXIE Protocol Specification
1202 - Directory Assistance Service
1107 - Plan for Internet directory services

6o. Servicios de información (WWW, Gopher, WAIS)

1738 - Uniform Resource Locators (URL)
1737 - Functional Requirements for Uniform Resource Names
1729 - Using the Z39.50 Information Retrieval Protocol in the Internet Environment
1728 - Resource Transponders
1727 - A Vision of an Integrated Internet Information Service
1714 - Referral Whois Protocol (RWhois)
1630 - Universal Resource Identifiers in WWW: A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as used in the World-Wide Web
1625 - WAIS over Z39.50-1988
1614 - Network Access to Multimedia Information
1436 - The Internet Gopher Protocol (a distributed document search and retrieval protocol)
954, 812 - Whois Protocol

6p. Protocolos de arranque y configuración (BOOTP, DHCP)

1542, 1532 - Clarifications and Extensions for the Bootstrap Protocol
1541, 1531 - Dynamic Host Configuration Protocol
1534 - Interoperation Between DHCP and BOOTP
1533, 1497, 1395, 1084, 1048 - DHCP Options and BOOTP Vendor Extensions
951 - Bootstrap Protocol

6q. Otros

1703, 1569 - Principles of Operation for the TPC.INT Subdomain: Radio Paging -- Technical Procedures"
1692 - Transport Multiplexing Protocol (TMux)
1645, 1568 - Simple Network Paging Protocol - Version 2
1530 - Principles of Operation for the TPC.INT Subdomain: General Principles and Policy

1529 - Principles of Operation for the TPC.INT Subdomain: Remote Printing -- Administrative Policies
1528 - Principles of Operation for the TPC.INT Subdomain: Remote Printing -- Technical Procedures
1546 - Host Anycasting Service
1492 - An Access Control Protocol, Sometimes Called TACACS
1486 - An Experiment in Remote Printing
1459 - Internet Relay Chat Protocol
1429 - Listserv Distribute Protocol
1413, 931, 912 - Identification Protocol
1307 - Dynamically Switched Link Control Protocol
1288, 1196, 1194, 742 - The Finger User Information Protocol
1193 - Client Requirements for Real-Time Communication Services
1179 - Line Printer Daemon Protocol
978 - Voice File Interchange Protocol (VFIP)
972 - Password Generator Protocol
937, 918 - Post Office Protocol
909 - Loader Debugger Protocol
891 - DCN Local Net Protocol
887 - Resource Location Protocol
866 - Active Users Protocol
865 - Quote of the Day Protocol
864 - Character Generator Protocol
863, 348 - Discard Protocol
862, 347 - Echo Protocol
767 - Document Formats
759 - Internet Message Protocol
734 - SUPDUP Protocol
666 - Specification of the Unified User-Level Protocol
621 - NIC User Directories at SRI-ARC
569 - Network Standard Text Editor
470 - Change in Socket for TIP News Facility
451 - Tentative Proposal for a Unified User Level Protocol
109 - Level III Server Protocol for the Lincoln Laboratory NIC 360/67 Host
098, 079 - Logger Protocol
029 - Note in Response to Bill English's Request for Comments

7. Documentación de programas

496 - A TNLS Quick Reference Card is Available
494 - Availability of MIX and MIXAL in the Network
488 - NLS Classes at Network Sites
485 - MIS and MIXAL at UCSB
431 - Update on SMFS Login and Logout
411 - New Multics Network Software Features
409 - TENEX Interface to UCSB's Simple-Minded File System
399 - SMFS Login and Logout
390 - TSO Scenario Batch Compilation and Foreground Execution
382 - Mathematical Software on the ARPA Network
379 - Using TSO at CCN
373 - Arbitrary Character Sets
350 - User Accounts for UCSB On-Line System
345 - Interest Mixed Integer Programming (MPSX on 360/91 at CCN)
321 - CBI Networking Activity at MITRE
311 - New Console Attachments to the UCSB Host
251 - Weather Data
217 - Specification Changes for OLS, RJE/RJOR, and SMFS
174 - UCLA-Computer Science Graphics Overview
122 - Network Specifications for UCSB's Simple-Minded File System
121 - Network On-Line Operators
120 - Network PL1 Subprograms
119 - Network FORTRAN Subprograms
074 - Specifications for Network Use of the UCSB On-Line System

8. Redes específicas (ver también sección 3)

8a. ARPANET

1005, 878, 851, 802 - The ARPANET 1822L Host Access Protocol
852 - The ARPANET Short Blocking Feature
789 - Vulnerabilities of Network Control Protocols: An Example
745 - JANUS interface specifications
716 - Interim Revision to Appendix F of BBN 1822
704 - IMP/Host and Host/IMP Protocol Change
696 - Comments on the IMP/HOST and HOST/IMP Protocol Changes
695 - Official Change in Host-Host Protocol
692 - Comments on IMP/Host Protocol Changes
690 - Comments on the Proposed Host/IMP Protocol Changes
687 - IMP/Host and Host/IMP Protocol
667 - BBN Host Ports

660 - Some Changes to the IMP and the IMP/Host Interface
642 - Ready Line Philosophy and Implementation
638, 633 - IMP/TIP Preventive Maintenance Schedule
632 - Throughput Degradation for Single Packet Message
627 - ASCII Text File of Hostnames
626 - On a possible Lockup Condition in IMP Subnet due to Message Sequencing
625 - On Line Hostnames Service
623 - Comments on On-line Host Name Service
622 - Scheduling IMP/TIP Down Time
620 - Request for Monitor Host Table Updates
619 - Mean Round-Trip Times in the ARPANET
613 - Network Connectivity: A Response to RFC 603
611 - Two Changes to the IMP/Host Protocol
606 - Host Names On-Line
594 - Speedup of Host-IMP Interface
591 - Addition to the Very Distant Host Specification
568, 567 - Cross-Country Network Bandwidth
548 - Hosts Using the IMP Going Down Message Specification
547 - Change to the Very Distant Host Specification
533 - Message-ID Numbers
528 - Software Checksumming in the IMP and Network Reliability
521 - Restricted Use of IMP DDT
508 - Real-Time Data Transmission on the ARPANET
476, 434 - IMP/TIP Memory Retrofit Schedules
449, 442 - The Current Flow-Control Scheme for IMPSYS
447, 445 - IMP/TIP Preventive Maintenance Schedule
417 - LINK Usage Violation
410 - Removal of the 30-second Delay When Hosts Come Up
406 - Scheduled IMP Software Releases
395 - Switch Settings on IMPs and TIPs
394 - Two Proposed Changes to the IMP-HOST Protocol
369 - Evaluation of ARPANET Services (January through March, 1972)
335 - New Interface-IMP/360
312 - Proposed Change in IMP-to-Host Protocol
297 - TIP Message Buffers
280 - A Draft Set of Host Names
274 - Establishing a Local Guide for Network Usage
273, 237 - The NIC's View of Standard Host Names
271 - IMP System Change Notification
270 - Correction to the BBN Report No. 1822

263 - "Very Distant" Host Interface
254 - Scenarios for Using ARPANET Computers
247 - Proffered Set of Standard Host Names
241 - Connecting Computers to NLC Ports
239 - Host Mnemonics Proposed in RFC 226
236 - Standard Host Names
233 - Standardization of Host Call Letters
230 - Toward Reliable Operation of Minicomputer-based Terminals on a TIP
229 - Standard Host Names
228 - Clarification
226 - Standardization of Host Mnemonics
218 - Changing the IMP Status Reporting
213 - IMP System Change Notification
209 - Host/IMP Interface Documentation
208 - Address Tables
073, 067 - Proposed Change to Host/IMP Spec to Eliminate Marking
071 - Reallocation in Case of Input Error
070 - A Note On Padding
064 - Getting Rid of Marking
041 - IMP/IMP Teletype Communication
025 - No High Link Numbers
019 - Two Protocol Suggestions to Reduce Congestion at Swap-Bound Nodes
017 - Some Questions Re: HOST-IMP Protocol
012 - IMP-HOST Interface Flow Diagrams
007 - HOST-IMP Interface
006 - Conversation with Bob Kahn

8b. Protocolos frontales de anfitrión

929, 928, 705, 647 - Host-Front End Protocol

8c. NCP de ARPANET (antecesor obsoleto del TCP/IP)

801 - NCP/TCP Transition Plan
773 - Comments on NCP/TCP Mail Service Transition Strategy
714 - A Host/Host Protocol for an ARPANET-type Network
689 - Tenex NCP Finite State Machine for Connections
663 - A Lost Message Detection and Recovery Protocol
636 - TIP/TENEX Reliability Improvements
635 - An Assessment of ARPANET Protocols
534, 516, 512 - Lost Message Detection
492, 467 - Proposed Change to Host-Host Protocol Resynchronization of Connection Status

489 - *Comment on Resynchronization of Connection Status Proposal*
425 - "But my NCP Costs $500 a day..."
210 - Improvement of Flow Control
176 - Comments on Byte Size for Connections
165 - A Proferred Official Initial Connection Protocol
147 - The Definition of a Socket
142 - Time-out Mechanism in the Host-Host Protocol
132, 124, 107, 102 - Output of the Host-Host Protocol Glitch Cleaning Committee
129 - A Request for Comments on Socket Name Structure
128 - Bytes
117 - Some Comments on the Official Protocol
072 - Proposed Moratorium on Changes to Network Protocol
068 - Comments on Memory Allocation Control Commands (CEASE, ALL, GVB, RET) and RFNM
065 - Comments on Host-Host Protocol Document Number 1
060 - A Simplified NCP Protocol
059 - *Flow Control-Fixed Versus Demand Allocation*
058 - Logical Message Synchronization
057, 054 - An Official Protocol Proffering
056 - Third Level Protocol
055 - A Prototypical Implementation of the NCP
050, 049, 047, 045, 044, 040, 039, 038, 036, 033 - New Host-Host Protocol
042 - Message Data Types
023 - Transmission of Multiple Control Messages
022 - Host-Host Control Message Formats
018 - Comments Re: Host-Host control link
015 - Network Subsystem for Time Sharing Hosts
011 - Implementation of the Host-Host Software Procedures in GORDO
009, 001 - Host Software
008 - ARPA Network Functional Specifications
005 - DEL
002 - Links

8d. Protocolo de conexión inicial de ARPANET

202 - Possible Deadlock in ICP
197 - Initial Connection Protocol - Revised
161 - A Solution to the Race Condition in the ICP
151, 148, 143, 127, 123 - A Proferred Official ICP
150 - The Use of IPC Facilities
145 - *Initial Connection Protocol Control Commands*

093 - Initial Connection Protocol
080 - Protocol and Data Formats
066 - 3rd Level Ideas and Other Noise

8e. USENET

1036 - Standard for interchange of USENET messages

8f. Otros

1553 - Compressing IPX Headers Over WAN Media (CIPX)
1132 - Standard for the transmission of 802.2 packets over IPX networks
935 - Reliable Link Layer Protocols
916 - Reliable Asynchronous Transfer Protocol
914 - Thinwire Protocol
824 - The Cronus Virtual Local Network

9. Mediciones

9a. General

1404 - A Model for Common Operational Statistics
1273 - A Measurement Study of Changes in Service-Level Reachability in the Global TCP/IP Internet: Goals, Experimental Design, Implementation, and Policy Considerations
1262 - Guidelines for Internet Measurement Activities
557 - Revelations in Network Host Measurements
546 - Tenex Load Averages for July 1973
462 - Responding to User Needs
415 - TENEX Bandwidth
392 - Measurement of Host Costs for Transmitting Network Data
352 - TIP Site Information Form
308 - ARPANET Host Availability Data
286 - Network Library Information System
214, 193 - Network Checkout
198 - Site Certification - Lincoln Labs
182 - Compilation of List of Relevant Site Reports
180 - File System Questionnaire
156 - Status of the Illinois Site (Response to RFC 116)
153 - SRI ARC-NIC Status
152 - SRI Artificial Intelligence Status Report
126 - Ames Graphics Facilities at Ames Research Center
112 - User/Server Site Protocol Network HOST Questionnaire
104 - Link 191
106 - USER/SERVER Site Protocol Network Host Questionnaire

9b. Evaluaciones

971 - A Survey of Data Representation Standards
876 - Survey of SMTP Implementations
848 - Who Provides the "Little" TCP Services?
847 - Summary of Smallberg Surveys
844 - Who Talks ICMP, too? Survey of 18 February 1983
846, 845, 843, 842, 839, 838, 837, 836, 835, 834, 833, 832 - Who Talks TCP?
787 - Connectionless Data Transmission Survey/Tutorial
565 - Storing Network Survey Data at the Datacomputer
545 - Of What Quality be the UCSB Resource Evaluators?
530 - A Report on the SURVEY Project
523 - SURVEY is in Operation Again
519 - Resource Evaluation
514 - Network Make-Work
464 - Resource Notebook Framework
460 - NCP Survey
459 - Network Questionnaire
450 - Multics Sampling Timeout Change
446 - Proposal to Consider a Network Program Resource Notebook
096 - An Interactive Network Experiment to Study Modes of Access to the Network Information Center
090 - CCN as a Network Service Center
081 - Request for Reference Information
078 - NCP Status Report: UCSB/Rand

9c. Estadísticas

1030 - On testing the NETBLT Protocol over divers networks
996 - Statistics Server
618 - A Few Observations on NCP Statistics
612, 601, 586, 579, 566, 556, 538, 522, 509, 497, 482, 455, 443, 422, 413, 400, 391, 378 - Traffic Statistics
603, 597, 376, 370, 367, 366, 362, 353, 344, 342, 332, 330, 326, 319, 315, 306, 298, 293, 288, 287, 267, 266 - Network Host Status
550 - NIC NCP Experiment
388 - NCP Statistics
255, 252, 240, 235 - Site Status

10. Privacía, seguridad y autentificación

10a. General

1751 - A Convention for Human-Readable 128-bit Keys
1750 - Randomness Recommendations for Security
1704 - On Internet Authentication
1511 - Common Authentication Technology Overview
1510 - The Kerberos Network Authentication Service (V5)
1509 - Generic Security Service API : C-bindings
1508 - Generic Security Service Application Program Interface
1507 - DASS - Distributed Authentication Security Service
1457 - Security Label Framework for the Internet
1455 - Physical Link Security Type of Service
1424 - Privacy Enhancement for Internet Electronic Mail: Part IV: Key Certification and Related Services
1423, 1115 - Privacy Enhancement for Internet Electronic Mail: Part III: Algorithms, Modes, and Identifiers
1422, 1114 - Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management
1421, 1113, 989 - Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures
1355 - Privacy and Accuracy Issues in Network Information Center Databases
1281 - Guidelines for the Secure Operation of the Internet
1244 - Site Security Handbook
1170 - Public Key Standards and Licenses
1135 - Helminthiasis of the Internet
1040 - Privacy enhancement for Internet electronic mail: Part I: Message encipherment and authentication procedures
1038 - Draft revised IP security option
1108 - U.S. Department of Defense Security Options for the Internet Protocol
1004 - Distributed-protocol authentication scheme

10b. Algoritmos Message-Digest

1321 - The MD5 Message-Digest Algorithm
1320, 1186 - The MD4 Message Digest Algorithm
1319 - The MD2 Message-Digest Algorithm

11. Experiencias y demostraciones de redes

1306 - Experiences Supporting By-Request Circuit-Switched T3 Networks
968 - 'Twas the Night Before Start-up
967 - All Victims Together
573 - Data and File Transfer - Some Measurement Results
527 - ARPAWOCKY

525 - MIT-Mathlab Meets UCSB-OLS
439 - PARRY Encounters the Doctor
420 - CCA ICC Weather Demo
372 - Notes on a Conversation with Bob Kahn on the ICCC
364 - Serving Remote Users on the ARPANET
302 - Exercising the ARPANET
231 - Service Center Standards for Remote Usage - A User's View
227 - Data Transfer Rates (RAND/UCLA)
113 - Network Activity Report: UCSB and Rand
089 - Some Historic Moments in Networking
004 - Network Timetable

12. Documentación de localidades

30, 27, 24, 10, 3 - Documentation Conventions

13. Estándares de protocolo de otros grupos de interés para Internet

13a. ANSI

183 - The EBCDIC Codes and Their Mapping to ASCII
020 - ASCII Format for Network Interchange

13b. NRC

942 - Transport Protocols for Department of Defense Data Networks
939 - Executive Summary of the NRC Report on Transport Protocols for Department of Defense Data Networks

13c ISO

1698 - Octet Sequences for Upper-Layer OSI to Support Basic Communications Applications
1629, 1237 - Guidelines for OSI NSAP Allocation in the Internet
1575, 1139 - An Echo Function for CLNP (ISO 8473)
1574 - Essential Tools for the OSI Internet
1561 - Use of ISO CLNP in TUBA Environments
1554 - ISO-2022-JP-2: Multilingual Extension of ISO-2022-JP
1330 - Recommendations for the Phase I Deployment of OSI Directory Services (X.500) and OSI Message Handling Services (X.400) within the ESnet Community
1238, 1162 - CLNS MIB - for use with Connectionless Network Protocol (ISO 8473) and End System to Intermediate System (ISO 9542)
1223 - OSI CLNS and LLC1 Protocols on Network Systems HYPERchannel
1008 - Implementation guide for the ISO Transport Protocol

1007 - Military supplement to the ISO Transport Protocol
995 - End System to Intermediate System Routing Exchange Protocol for Use in Conjunction with ISO 8473
994 - Final Text of DIS 8473, Protocol for Providing the Connectionless Mode Network Service
982 - Guidelines for the Specification of the Structure of the Domain Specific Part (DSP) of the ISO Standard NSAP Address
941 - Addendum to the Network Service Definition Covering Network Layer Addressing
926 - Protocol for Providing the Connectionless-Mode Network Services
905 - ISO Transport Protocol Specification (ISO DP 8073)
892 - ISO Transport Protocol
873 - The Illusion of Vendor Support

14. Interoperabilidad con otras aplicaciones y protocolos

14a. Traducción de protocolos y puentes

1086 - ISO-TP0 bridge between TCP and X.25
1029 - More fault tolerant approach to address resolution for a Multi-LAN system of Ethernets

14b. Procedimiento mediante túneles y estratificación por capas

1634, 1551, 1362 - Novell IPX Over Various WAN Media (IPXWAN)
1613 - cisco Systems X.25 over TCP (XOT)
1538 - Advanced SNA/IP : A Simple SNA Transport Protocol
1434 - Data Link Switching: Switch-to-Switch Protocol
1356 - Multiprotocol Interconnect on X.25 and ISDN in the Packet Mode
1240 - OSI Connectionless Transport Services on top of UDP - Version: 1
1234 - Tunneling IPX Traffic through IP Networks
1085 - ISO presentation services on top of TCP/IP based internets
1070 - Use of the Internet as a subnetwork for experimentation with the OSI network layer
1006 - ISO transport services on top of the TCP: Version: 3

14c. Transformación de nombres, direcciones e identificadores

1439 - The Uniqueness of Unique Identifiers
1236 - IP to X.121 Address Mapping for DDN
1069 - Guidelines for the use of Internet-IP addresses in the ISO Connectionless-Mode Network Protocol

15. Misceláneos

15a. General

1746 - Ways to Define User Expectations
1739 - A Primer On Internet and TCP/IP Tools
1709 - K-12 Internetworking Guidelines
1681 - On Many Addresses per Host
1627 - Network 10 Considered Harmful (Some Practices Shouldn't be Codified)
1691 - The Document Architecture for the Cornell Digital Library
1633 - Integrated Services in the Internet Architecture: an Overview
1607 - A VIEW FROM THE 21ST CENTURY
1605 - SONET to Sonnet Translation
1580 - Guide to Network Resource Tools
1578 - FYI on Questions and Answers: Answers to Commonly Asked "Primary and Secondary School Internet User" Questions
1501 - OS/2 User Group
1498 - On the Naming and Binding of Network Destinations
1462 - FYI on "What is the Internet?"
1594, 1325, 1206, 1177 - FYI on Questions and Answers - Answers to Commonly Asked "New Internet User" Questions
1470, 1147 - FYI on a Network Management Tool Catalog: Tools for Monitoring and Debugging TCP/IP Internets and Interconnected Devices
1453 - A Comment on Packet Video Remote Conferencing and the Transport/Network Layers
1432 - Recent Internet Books
1417, 1295 - NADF Standing Documents: A Brief Overview
1402, 1290 - There's Gold in them thar Networks! Searching for Treasure in all the Wrong Places
1400 - Transition and Modernization of the Internet Registration Service
1392 - Internet Users' Glossary
1359 - Connecting to the Internet What Connecting Institutions Should Anticipate
1345 - Character Mnemonics & Character Sets
1336, 1251 - Who's Who in the Internet: Biographies of IAB, IESG and IRSG Members
1324 - A Discussion on Computer Network Conferencing
1313 - Today's Programming for KRFC AM 1313 Internet Talk Radio
1302 - Building a Network Information Services Infrastructure
1300 - Remembrances of Things Past

1296 - Internet Growth (1981-1991)
1291 - Mid-Level Networks: Potential Technical Services
1259 - Building The Open Road: The NREN As Test-Bed For The National Public Network
1257 - Isochronous Applications Do Not Require Jitter-Controlled Networks
1242 - Benchmarking Terminology for Network Interconnection Devices
1217 - Memo from the Consortium for Slow Commotion Research (CSCR)
1216 - Gigabit Network Economics and Paradigm Shifts
1208 - A Glossary of Networking Terms
1207 - Answers to Commonly asked "Experienced Internet User" Questions
1199, 1099 - Request for Comments Summary RFC Numbers 1100-1199
1192 - Commercialization of the Internet Summary Report
1181 - RIPE Terms of Reference
1180 - A TCP/IP Tutorial
1178 - Choosing a Name for Your Computer
1173 - Responsibilities of Host and Network Managers A Summary of the "Oral Tradition" of the Internet
1169 - Explaining the Role of GOSIP
1167 - Thoughts on the National Research and Education Network
1121 - Act one - the poems
1118 - Hitchhikers guide to the Internet
1015 - Implementation plan for interagency research Internet
992 - On communication support for fault tolerant process groups
874 - A Critique of X.25
531 - Feast or famine? A response to two recent RFC's about network information
473 - MIX and MIXAL?
472 - Illinois' reply to Maxwell's request for graphics information NIC 14925
429 - Character generator process
408 - NETBANK
361 - Deamon processes on host 106
313 - Computer based instruction
256 - IMPSYS change notification
225 - Rand/UCSB network graphics experiment
219 - User's view of the datacomputer
187 - Network/440 protocol concept
169 - Computer networks
146 - Views on issues relevant to data sharing on computer networks
013 - No Title

15b. Bibliografías

1463 - FYI on Introducing the Internet--A Short Bibliography of Introductory Internetworking Readings for the Network Novice
1175 - FYI on Where to Start - A Bibliography of Internetworking Information
1012 - Bibliography of Request For Comments 1 through 999
829 - Packet Satellite Technology Reference Sources
290 - Computer Network and Data Sharing: A Bibliography
243 - Network and Data Sharing Bibliography

16. No publicados

16a. Nunca publicados

1061, 853, 723, 715, 711, 710, 709, 693, 682, 676, 673, 670, 668, 665, 664, 650, 649, 648, 646, 641, 639, 605, 583, 575, 572, 564, 558, 554, 541, 540, 536, 517, 507, 502, 484, 481, 465, 444, 428, 427, 424, 397, 383, 380, 375, 358, 341, 337, 284, 279, 277, 275, 272, 262, 261, 260, 259, 258, 257, 248, 244, 220, 201, 159, 092, 026, 014

16b. Ya no publicados

1752, 1747, 1736, 1719, 1699, 1693, 1599, 1499, 1399, 1299, 1260, 1182

Apéndice 2

Glosario de abreviaturas y términos de enlace de redes

Terminología TCP/IP

Como la mayor parte de los grandes proyectos, el TCP/IP tiene su lenguaje propio. Una curiosa combinación de jerga de la red, nombres de protocolos, nombres de proyectos y nombres de agencias gubernamentales hacen a este lenguaje difícil de entender y de recordar. Para quienes no conocen el ambiente de las redes, un análisis entre conocedores suena como un parloteo sin sentido en el que se pronuncian siglas cada vez que es posible. El problema se complica pues algunos términos están definidos de manera muy imprecisa y el gran volumen está hundido en aplicaciones muy específicas.

Este glosario ayuda a resolver el problema puesto que proporciona definiciones breves para términos utilizados en Internet. No pretende ser un tutorial para principiantes. Por el contrario, nos concentramos en proporcionar una referencia concisa de modo que sea fácil y rápido buscar el significado de términos o siglas para quienes tienen conocimientos generales sobre redes. Los lectores encontrarán este glosario sustancialmente más útil como referencia luego de haber estudiado el texto que viene antes.

Glosario de términos y abreviaturas en orden alfabético

10Base-T

Nombre técnico para el par trenzado de Ethernet.

576

Tamaño mínimo de datagrama que todos los anfitriones y ruteadores deben manejar.

802.3

Estándar IEEE para Ethernet.

822

Formato estándar del TCP/IP para los mensajes de correo electrónico. Los expertos por lo general se refieren a él como "mensajes 822". El nombre proviene del RFC 822 que contiene las especificaciones. El formato 822 se conoció anteriormente como formato 733.

AAL

Abreviatura de *ATM Adaptation Layer* (*Capa de Adaptación ATM*).

ACK

Abreviatura de *acknowledgement* (*reconocimiento o acuse de recibo*).

acknowledgement (reconocimiento o acuse de recibo)

Respuesta enviada por un receptor para indicar que recibió con éxito la información que le fue enviada. Los acuses de recibo se pueden implantar en cualquier nivel, incluyendo el nivel físico (utilizando el voltaje en uno o más cables para coordinar la transferencia), en el nivel de enlace (para indicar la transmisión exitosa a través de un solo enlace de hardware) o en niveles elevados (por ejemplo, para permitir que un programa de aplicación, en el destino final, responda a un programa de aplicación en la fuente).

active open (apertura activa)

Operación que realiza un cliente para establecer una conexión TCP con un servidor en una dirección conocida.

address mask (máscara de dirección)

Máscara de bits utilizada para seleccionar bits de una dirección IP a fin de direccionar subredes. La máscara es de 32 bits de longitud y selecciona la porción de red de la dirección IP así como uno o más bits de la parte local.

address resolution (resolución de dirección)

Conversión de una dirección de protocolo en su correspondiente dirección física (por ejemplo, la conversión de una dirección IP en una dirección Ethernet). Dependiendo de la red subyacente, la resolución puede requerir difusión en una red local. Ver ARP.

Advanced Networks and Services

Compañía propietaria y operadora de la columna vertebral de la red de Internet en 1995.

agent (agente)

En la administración de redes, un agente es el software servidor que corre en un anfitrión o ruteador que se está administrando.

ANSI

(*American National Standards Institute*) Grupo que define los estándares de Estados Unidos para la industria del procesamiento de información. ANSI participa en la definición de los estándares de protocolos de red.

anonymous FTP (FTP anónimo)

Sesión FTP que utiliza el nombre de identificación *anónimo* para accesar archivos públicos. Un servidor que permite FTP anónimo por lo general acepta la clave de acceso *guest* (*visitante*).

ANS

Abreviatura de *Advanced Networks and Services.*

ANSNET

Red de área amplia que formaba la red de columna vertebral de Internet en 1995.

ARP

(*Address Resolution Protocol*) Protocolo TCP/IP utilizado para asignar una dirección IP de alto nivel a una dirección de hardware físico de bajo nivel. ARP se utiliza a través de una sola red física y está limitada a redes que soportan difusión de hardware.

ARPA

(*Advanced Research Projects Agency*) Institución gubernamental que fundó la ARPANET y, después, la red global Internet. El grupo dentro de ARPA responsable de ARPANET fue la IPTO (*Information Processing Techniques Office*), llamada más tarde ISTO (*Information Systems Technology Office*). ARPA se conoció como *DARPA* por varios años.

ARPANET

Red pionera de gran alcance fundada por ARPA (después DARPA) y construida por BBN. Sirvió de 1969 a 1990 como base para las primeras investigaciones de red y como columna vertebral de red durante el desarrollo de Internet. ARPANET consiste en nodos individuales conmutadores de paquetes interconectados por líneas arrendadas.

ARQ

(*Automatic Repeat reQuest*) Cualquier protocolo que utilice acuses de recibo positivos y negativos con técnicas de retransmisión para asegurar la confiabilidad. El emisor repite la solicitud de manera automática si no recibe una respuesta.

ASN.1

(*Abstract Syntax Notation.1*) Protocolo estándar de presentación de ISO utilizado por SNMP para representar mensajes.

Assigned Numbers (números asignados)

Documento RFC que especifica (por lo general en forma numérica) los valores utilizados por los protocolos TCP/IP.

ATM

(*Asynchronous Transfer Mode*) Tecnología de red orientada a la conexión que utiliza pequeñas celdas de tamaño fijo en la capa de nivel inferior. ATM tiene la ventaja potencial de ser capaz de soportar voz, video y datos con una sola tecnología subyacente.

ATM Adaptation Layer (AAL)

Uno de varios protocolos definidos por ATM que especifica cómo envía y recibe una aplicación información en una red ATM. La transmisión de datos emplea AAL5.

ATMARP

Protocolo utilizado por un anfitrión para la resolución de direcciones cuando se hacen envíos IP en una red ATM.

AUI

Abreviatura de *Attachment Unit Interface*, el conector utilizado en las redes Ethernet de cable grueso.

authority zone (zona de autoridad)

Parte de la jerarquía de nombres de dominio en la que un solo servidor de nombre es la autoridad.

autonomous system (sistema autónomo)

Grupo de ruteadores y redes bajo una entidad administrativa que cooperan de cerca para ampliar la accesibilidad de la red (y el ruteo) comunicándose entre sí mediante el protocolo de compuerta interior de su elección. Los ruteadores dentro de un sistema autónomo tienen un alto grado de confiabilidad. Antes de que dos sistemas autónomos se puedan comunicar, un ruteador en cada sistema envía información de accesibilidad hacia un ruteador en el otro.

backbone network (red de columna vertebral de la red)

Cualquier red que forme la interconexión central para una red de redes. Una columna vertebral de red nacional es una WAN; una columna vertebral de red corporativa puede ser una LAN.

base header (encabezado base)

En la propuesta IPng, es el encabezado que se encuentra al comienzo de cada datagrama.

baseband (banda de base)

Característica de cualquier tecnología de red, como Ethernet que utiliza una sola frecuencia portadora y requiere que todas las estaciones estén conectadas a la red para participar en todas las transmisiones. Comparar con broadband (de banda ancha).

bastion host (anfitrión baluarte)

Computadora segura que forma parte del sistema de muro de seguridad y corre aplicaciones que se comunican con el exterior de una organización.

baud

Literalmente, el número de veces que una señal puede cambiar por segundo en una línea de transmisión. Por lo común, la línea de transmisión utiliza sólo dos señales de transmisión (por ejemplo, dos voltajes), lo que hace que la cantidad de baudios sea igual al número de bits por segundo que pueden ser transferidos. La técnica de transmisión subyacente puede utilizar parte del ancho de banda, pero podría suceder que el usuario no transfiriera datos según la razón de transferencia de bits especificados para la línea. Por ejemplo, dado que las líneas asíncronas requieren 10 bits por vez para enviar un carácter de 8 bits, una línea de transmisión asíncrona a 9600 bauds puede enviar sólo 960 caracteres por segundo.

Berkeley broadcast

Dirección de difusión IP no estándar que utiliza ceros en la porción del anfitrión en lugar de unos. El nombre se debe a que la técnica fue introducida y difundida en UNIX BSD de Berkeley.

best-effort delivery (entrega con el mejor esfuerzo)

Característica de las tecnologías de red que no proporcionan confiabilidad en el nivel de enlace. El IP funciona bien en el hardware de entrega con el mejor esfuerzo pues no asume que la red subyacente proporciona confiabilidad. El protocolo UDP proporciona el servicio de entrega con el mejor esfuerzo para los programas de aplicación.

BGP

Abreviatura de *Border Gateway Protocol.*

big endian

Formato para el almacenamiento o la transmisión de datos binarios, en el cual el octeto (bit) más significativo se encuentra primero. El orden de los octetos de red del estándar TCP/IP es de tipo big endian. Comparar con *little endian.*

BISYNC

(*BInary SYNchronous Communication*) Uno de los primeros protocolos de bajo nivel desarrollados por IBM y utilizado para transmitir datos a través de un enlace de comunicación síncrono. A diferencia de la mayor parte de los protocolos de nivel de enlace modernos, BISYNC está orientado al octeto (byte-oriented), lo que significa que utiliza caracteres especiales para marcar el comienzo y el fin de las tramas. BISYNC se conoce a menudo como BSC, especialmente en los productos comerciales.

BNC

Tipo de conector utilizado con Ethernet de cable delgado.

BOOTP

Abreviatura de *BOOTstrap Protocol*, protocolo que utiliza un anfitrión para obtener información de arranque, incluyendo la dirección IP, de un servidor.

Border Gateway Protocol (BGP)

Protocolo de compuerta o pasarela exterior utilizado en NSFnet. Han aparecido cuatro versiones mayores de BGP.

bps

(*bits per second*) Medida de la razón de transmisión de datos.

bridge (puente)

Computadora que conecta dos o más redes y envía paquetes entre ellas. Los puentes operan a nivel de red física. Por ejemplo, un puente Ethernet conecta dos cables Ethernet físicos y envía de un cable al otro sólo los paquetes que no son locales. Los puentes difieren de los repetidores pues almacenan y envían paquetes completos, mientras que los repetidores envían todas las señales eléctricas. Los puentes difieren de los ruteadores puesto que los puentes se valen de direcciones físicas, mientras que los ruteadores utilizan direcciones IP.

broadband (de banda ancha)

Característica de cualquier tecnología de red que multiplexa varias portadoras de red independientes en un solo cable (usualmente utilizando el multiplexado por división de frecuencia). Por ejemplo, un cable de banda ancha de 50 Mbps puede dividirse en 5 portadoras de 10 Mbps, cada una de ellas tratada como una red independiente. La ventaja de la banda ancha es que se emplea menos cable, la desventaja es el alto costo de los equipos de conexión. Ver baseband (banda de base).

broadcast (difusión)

Sistema de entrega que proporciona la copia de un paquete dado a todos los anfitriones conectados para la difusión del paquete. La difusión puede implantarse con hardware (por ejemplo, en una red Ethernet) o con software (digamos, con Cypress).

brouter

(*Bridging ROUTER*) Dispositivo que opera como puente para unos protocolos y como ruteador para otros (por ejemplo, un *brouter* puede puentear protocolos DECNET y rutear IP).

BSC

(*Binary Synchronous Communication*) Ver BISYNC.

BSD UNIX

(*Berkeley Software Distribution UNIX*) Versión de UNIX distribuida por U.C. Berkeley o uno de los sistemas comerciales derivados de él. BSD UNIX fue el primero en incluir los protocolos TCP/IP.

Chernobylgram

(*Chernobyl datagram*) Un paquete tan malformado que ocasionará que el sistema receptor se "derrita" (es decir que lo pondrá fuera de funcionamiento).

CCIRN

(*Coordinating Committee for Intercontinental Research Networking*) Grupo internacional que ayuda a coordinar la cooperación internacional en la investigación y el desarrollo del enlace de redes.

CCITT

(*Consultative Committee on International Telephony and Telegraphy*) Nombre actual de la International Telecommunications Union.

cell (celda, célula)

Pequeña trama de tamaño fijo utilizada en las redes ATM. Cada celda ATM contiene 48 octetos de datos y 5 octetos de encabezado.

checksum (suma de verificación)

Número entero calculado a partir de una secuencia de octetos que son tratados como enteros en una suma para calcular su valor total. Una suma de verificación se utiliza para detectar errores que aparecen cuando una secuencia de octetos se transmite de una máquina a otra. Por lo general, el software de protocolo calcula una suma de verificación y la anexa al paquete que se está transmitiendo. En la recepción, el software de protocolo verifica el contenido del paquete recalculando la suma de verificación y comparándolo con el dato obtenido de la transmisión. Muchos protocolos TCP/IP utilizan una suma de verificación de 16 bits, calculada por complemento aritmético a uno con todos los campos enteros en el paquete almacenados en el orden de octetos de la red.

CIDR

Abreviatura de *Classless Inter-Domain Routing*.

class of address (clase de dirección)

Categoría de una dirección IP. La clase de dirección determina la localización de la frontera entre el prefijo de red y el sufijo de anfitrión.

Classless InterDomain Routing (CIDR)

Esquema de ruteo y direccionamiento que emplea un grupo de direcciones de clase C contiguas en lugar de direcciones de clase B. CIDR fue adoptado como solución temporal al problema del agotamiento de espacio de direcciones de clase B.

client-server (cliente-servidor)

Modelo de interacción en un sistema distribuido en el que un programa, en una localidad, envía una solicitud a otro programa en otra localidad y espera una respuesta. El programa solicitante se conoce como cliente, el programa que atiende la solicitud como servidor. Es común que se estructure primero el software cliente y después el servidor.

connection (conexión)

Abstracción proporcionada por el software de protocolo. El TCP ofrece una conexión de una aplicación en una computadora a la aplicación en otra.

connectionless service (servicio sin conexión)

Característica del servicio de entrega de paquetes ofrecida por la mayor parte del hardware y por el Protocolo de Internet (IP). Un servicio sin conexión trata a cada paquete o datagrama como un entidad separada que contiene las direcciones de fuente y destino.

core gateway (compuerta núcleo)

Ruteador, de un conjunto de ruteadores, con rutas explícitas hacia todos los destinos en una red de redes. El sistema de núcleo de Internet participa en un solo protocolo de ruteo; todos los ruteadores núcleo intercambian actualizaciones de rutas de manera periódica para asegurarse de que sus tablas de ruteo se mantienen consistentes.

CRC

(*Cyclic Redundancy Code*) Número entero calculado a partir de una secuencia de octetos utilizados para detectar errores que aparecen cuando una secuencia de octetos se transmite de una máquina a otra. Por lo general, el hardware de red de conmutación de paquetes calcula un CRC y lo añade a un paquete cuando se transmite. Durante la recepción, el hardware verifica el contenido del paquete recalculando el CRC y comparándolo con el valor enviado. Aun cuando hace a las computadoras más caras, un CRC detecta más errores que una suma de verificación que se vale de métodos de suma.

CSMA/CD

Característica del hardware de red que al operar permite que varias estaciones compitan por el acceso a un medio de transmisión escuchando para saber si el medio está ocupado, y mecanismo que permite al hardware detectar cuando dos estaciones intentan transmisiones simultáneas. Ethernet utiliza CSMA/CD.

DARPA

(*Defense Advanced Research Projects Agency*) Nombre anterior de ARPA.

datagram (datagrama)

Ver datagrama IP.

DCE

(*Data Communications Equipment*) Término de los estándares de protocolo ITU-TS que se aplica al equipo de conmutación que forma una red de conmutación de paquetes para distinguirlos de las computadoras o terminales que se conectan a la red. Ver también DTE.

DDCMP

(*Digital Data Communication Message Protocol*) Protocolo a nivel de enlace utilizado en la red de columna vertebral NSFNET original.

DDN

(*Defense Data Network*) Parte de Internet asociada con las localidades militares de Estados Unidos.

demultiplex (demultiplexor)

Dispositivo que separa una entrada común en varias salidas. El demultiplexado se presenta en muchos niveles. El hardware demultiplexa señales de una línea de transmisión basada en tiempo o en una frecuencia portadora para permitir varias transmisiones simultáneas a través de un solo cable físico. El software demultiplexa los datagramas entrantes enviando cada uno hacia el módulo de protocolo de alto nivel apropiado o a un programa de aplicación.

DHCP

(*Dynamic Host Configuration Protocol*) Protocolo utilizado por un anfitrión para obtener toda la información de configuración necesaria incluida en una dirección IP.

directed broadcast address (dirección de difusión dirigida)

Dirección IP que especifica a "todos los anfitriones" en una red dada. La copia de una difusión dirigida se rutea hacia la red especificada desde la que es difundida a todas las máquinas de una red.

DNS

(*Domain Name System*) Sistema de base de datos distribuida en línea y utilizado para transformar nombres de máquina en direcciones IP que puedan leer los usuarios. Los servidores DNS, a través de Internet, implantan un espacio de nombres jerárquico que permite a las localidades contar con libertad para asignar nombres de máquinas y direcciones. DNS también soporta transformaciones separadas entre destinos de correo y direcciones IP.

domain (dominio)

Parte de una jerarquía de nombres. Sintácticamente, un nombre de dominio consiste en una secuencia de nombres (etiquetas) separadas por puntos.

dotted decimal notation (notación decimal con puntos)

Representación sintáctica para un entero de 32 bits que consiste en cuatro números de 8 bits escritos en base 10 con puntos que los separan. Muchos programas de aplicación TCP/IP aceptan la notación decimal con puntos en lugar de los nombres de máquinas de destino.

DS3

Clasificación de telefonía relacionada con la velocidad de líneas arrendadas equivalente a 45 Mbps aproximadamente.

DTE

(*Data Terminal Equipment*) Término de los estándares de protocolo ITU-TS aplicado a computadoras y/o a terminales para distinguirlas de una red de conmutación de paquetes a la que están conectadas. Ver también DCE.

DVMRP

(*Distance Vector Multicast Routing Protocol*) Protocolo utilizado para difundir rutas de multidifusión.

E.164

Formato de una dirección especificado por ITU-TS y utilizado con ATM.

EACK

(*Extended ACKnowledgement*) Sinónimo de SACK.

Ethernet meltdown

Hecho que ocasiona una saturación, o casi, en una red Ethernet. Por lo general es resultado de paquetes ilegales o de ruteo equivocado y normalmente dura un intervalo de tiempo corto.

EGP

(*Exterior Gateway Protocol*) Protocolo utilizado por un ruteador, en un sistema autónomo, para anunciar la dirección IP de la red de tal sistema hacia un ruteador en otro sistema autónomo.

EIA

(*Electronics Industry Association*) Organización de estándares para la industria electrónica. Conocida por los estándares RS232C y RS422 que especifican las características eléctricas de las interconexiones entre terminales y computadoras o entre dos computadoras.

encapsulation (encapsulación)

Técnica utilizada por los protocolos estratificados por capas en la cual un protocolo de nivel inferior acepta un mensaje de un protocolo de nivel superior y lo coloca en la sección de datos de su trama de bajo nivel. La encapsulación implica que los datagramas que viajan a través de una red física cuentan con una secuencia de encabezados de los que el primero proviene de la trama de red física, el siguiente del Protocolo Internet (IP), el siguiente del protocolo de transporte, y así sucesivamente.

epoch date (fecha de época)

Fecha elegida como fecha a partir de la cual se mide el tiempo. El TCP/IP utiliza como fecha de época el 1 de enero de 1900, del Tiempo Universal (formalmente, llamado Tiempo del Meridiano de Greenwich). Cuando los programas TCP/IP intercambian fechas u horas del día, las expresan como el número de segundos transcurridos desde la fecha de época.

Ethernet

Popular tecnología de red de área local inventada en el Palo Alto Research Center, de Xerox Corporation. Ethernet es un cable coaxial pasivo; las interconexiones contienen todos los componentes activos. Ethernet es un sistema de entrega con el mejor esfuerzo que utiliza tecnología CSMA/CD. Xerox Corporation, Digital Equipment Corporation, e Intel Corporation desarrollaron y publicaron el estándar para Ethernet de 10 Mbps. Originalmente, Ethernet utilizaba un cable coaxial. En versiones posteriores empezó a utilizar un cable coaxial delgado (*thinnet*) o un cable de par trenzado (10Base-T).

eXternal Data Representation

Ver XDR.

fair queueing (cola de espera justa)

Técnica bien conocida para controlar el congestionamiento en los ruteadores. Se le llama "fair" (justa) pues restringe a todos los anfitriones a fin de que compartan equitativamente el ancho de banda de un ruteador. La técnica de cola de espera justa no es completamente satisfactoria ya que no distingue entre pequeños y grandes anfitriones o entre anfitriones con unas cuantas conexiones activas y otros con varias.

FCCSET

(*Federal Coordinating Council for Science, Engineering, and Technology*) Grupo gubernamental notable por sus reportes sobre la computación de alta velocidad y la investigación de redes de alta velocidad.

FDDI

(*Fiber Distribution Data Interface*) Tecnología de red token ring basada en fibras ópticas. FDDI especifica una razón de transferencia de datos a 100 Mbps utilizando luz con una longitud de onda de 1300 nanómetros, limitando las redes a 200 km de longitud aproximadamente y con repetidores cada 2 km o menos.

FDM

(*Frequency Division Multiplexing*) Método de transferencia de varias señales independientes a través de un solo medio asignando a cada frecuencia de portadora. El hardware que combina las señales se conoce como multiplexor; el que separa las señales se conoce como demultiplexor. Ver también TDM.

file server (servidor de archivos)

Proceso que corre en una computadora para proporcionar acceso a los archivos en esa misma computadora, a solicitud de programas que corren en máquinas remotas. El término se aplica con frecuencia y con cierta ambigüedad a computadoras que corren programas servidores de archivos.

firewall (muro de seguridad)

Configuración de ruteadores y redes colocados entre la organización interna de una red de redes y su conexión con redes de redes externas, con el fin de proporcionar seguridad.

flat namespace (espacio de nombres plano)

Característica de cualquier espacio de nombres en el que los nombres de objetos se seleccionan de un solo conjunto de cadenas (por ejemplo, los nombres en una ciudad). El espacio de nombres plano contrasta con el espacio de nombres jerárquico, en el cual los nombres se subdividen en subsecciones que corresponden a la jerarquía de autoridad que las administra.

flow control (control de flujo)

Control de la razón de transferencia a la que introducen los anfitriones y los ruteadores paquetes en una red o en una red de redes, por lo general para evitar congestionamientos.

fragmentation (fragmentación)

Proceso de dividir un datagrama IP en pequeñas piezas cuando deben viajar a través de una red que no puede manejar el tamaño del datagrama original. Cada fragmento tiene el mismo formato que un datagrama; los campos en el encabezado IP especifican si un datagrama es un fragmento y, de ser así, el desplazamiento del fragmento con respecto al datagrama original. El software IP en el receptor final debe reensamblar los fragmentos para obtener el datagrama original.

frame (trama)

Literalmente, un paquete transmitido a través de una línea serial. El término deriva de los protocolos orientados a carácter que añaden caracteres especiales de comienzo-de-trama y de fin-de-trama cuando transmiten paquetes. Este término se utilizó a lo largo de este libro para nombrar a los objetos que transmiten las redes físicas.

FTP

(*File Transfer Protocol*) Protocolo estándar de alto nivel del TCP/IP que sirve para transferir archivos de una máquina a otra. El FTP utiliza al TCP.

FYI

(*For Your Information*) Subconjunto de RFC que no son estándares técnicos o descripciones de protocolos. Los FYI por lo general cubren información sobre temas relacionados con el TCP/IP o Internet.

gated

(*GATEway Daemon*) Programa que corre en un ruteador que utiliza un IGP para reunir información de ruteo desde dentro de un sistema autónomo, y un EGP para anunciar la información a otros sistemas autónomos.

gateway (compuerta)

Originalmente los investigadores utilizaron la *gateway* (*compuerta*) IP para referirse a las computadoras dedicadas al ruteo de paquetes; los vendedores han adoptado el término *ruteador*. Compuerta significa, ahora, programa de aplicación que interconecta dos servicios (por ejemplo, una compuerta de correo).

gateway requirements (requerimientos de compuerta)

Documento que especifica los requerimientos para un ruteador IP.

GGP

(*Gateway to Gateway Protocol*) Protocolo originalmente utilizado por las compuertas núcleo para intercambiar información de ruteo. En la actualidad el GGP es obsoleto.

gopher

Servicio de información utilizado a través de Internet.

GOSIP

(*Government Open Systems Interconnection Profile*) Documento proporcionado por el gobierno de Estados Unidos que especifica las instituciones que pueden utilizar los protocolos OSI en las redes nuevas desde agosto de 1991. Aun cuando GOSIP se pensó originalmente para eliminar el uso del TCP/IP en las redes de redes del gobierno, se han hecho aclaraciones para especificar que las dependencias gubernamentales pueden continuar usando el TCP/IP.

hardware address (dirección de hardware)

Dirección de bajo nivel utilizada por las redes físicas. Cada tipo de hardware de red tiene su propio esquema de direccionamiento (por ejemplo, en una red Ethernet las direcciones son de 48 bits).

HELLO

Protocolo utilizado en la red de columna vertebral NSFNET. Hello resulta interesante pues elige rutas con un tiempo de retardo mínimo.

HELO

Comando en el intercambio inicial del protocolo SMTP.

hierarchical routing (ruteo jerárquico)

Ruteo que se basa en un esquema de direccionamiento jerárquico. La mayor parte del ruteo del TCP/IP se basa en una jerarquía de dos niveles en la que una dirección IP se divide una parte de red y otra parte en anfitrión. Los ruteadores utilizan sólo la parte de red hasta que los datagramas alcanzan un ruteador que los pueda entregar directamente. Las subredes introducen niveles adicionales de ruteo jerárquico.

hop count (conteo de saltos)

Medición de la distancia entre dos puntos en una red de redes. Un conteo de saltos *n* significa que *n* ruteadores separan a la fuente y al destino.

host (anfitrión)

Cualquier sistema de computadora de usuario final que se conecta a una red. Los anfitriones abarcan desde computadoras personales hasta supercomputadoras. Comparar con ruteador.

host requirements (requerimientos de anfitrión)

Documento extenso que contiene la revisión y actualización de muchos protocolos TCP/IP. Los documentos de requerimientos de anfitrión están publicados en un par de RFC.

hub (concentrador)

Dispositivo electrónico al que se conectan varias computadoras, por lo general mediante un cable de par trenzado. Un concentrador simula una red que interconecta a las computadoras conectadas. La tecnología de concentradores es popular en Ethernet.

IAB

(*Internet Architecture Board*) Pequeño grupo de personas que establece las políticas y directivas para el TCP/IP y la red global de Internet. Ver IETF.

IANA

(*Internet Assigned Number Authority*) Grupo responsable de la asignación de constantes utilizadas en los protocolos TCP/IP. Estas constantes generalmente son números.

ICCB

(*Internet Control and Configuration Board*) Predecesor de la IAB.

ICMP

(*Internet Control Message Protocol*) Parte integral del protocolo de Internet (IP) que resuelve errores y controla los mensajes. Específicamente, los anfitriones y los ruteadores utilizan el ICMP para enviar reportes de problemas relacionados con datagramas que se devuelven a la fuente original que envía el datagrama. El ICMP también incluye una solicitud/réplica de eco utilizada para probar si un destino es accesible y responde.

IEN

(*Internet Engineering Notes*) Serie de notas desarrolladas de manera paralela a los RFC. Aun cuando la serie es obsoleta, algunos IEN contienen las primeras discusiones del TCP/IP e Internet que no se encuentran en los RFC.

IETF

(*Internet Engineering Task Force*) Grupo de personas vinculado de cerca con el IAB, que trabaja en el diseño y la ingeniería del TCP/IP y la red global de Internet. El IETF se divide en áreas, cada una de las cuales cuenta con una administración independiente. Las áreas, a su vez, se dividen en grupos de trabajo.

IESG

(*Internet Engineering Steering Group*) Comité del personal directivo de IETF y los gerentes de área. El IESG coordina las actividades entre los grupos de trabajo del IETF.

IGP

(*Interior Gateway Protocol*) Término genérico aplicado a cualquier protocolo utilizado para difundir accesibilidad de red e información de ruteo dentro de un sistema autónomo. Aun cuando no es el único estándar IGP, el RIP está entre los más populares.

IGMP

(*Internet Group Management Protocol*) Protocolo que utilizan los anfitriones para mantener a los ruteadores locales informados de sus miembros y de sus grupos de multidifusión. Cuando todos los anfitriones abandonan un grupo, los ruteadores no envían los datagramas que lleguen para el grupo.

INOC

(*Internet Network Operations Center*) Originalmente, era un grupo de personas en BBN que monitoreaba y controlaba el sistema de compuertas núcleo de Internet. Ahora, se aplica a cualquier grupo que monitorea una red de redes.

International Organization for Standardization

Ver ISO.

International Telecommunications Union (ITU)

Organización Internacional que establece los estándares para la interconexión del equipo telefónico. Ésta definió el estándar para el protocolo de red X.25. (Nota: en Europa, PTT ofrece servicio telefónico de voz y servicio de red X.25).

Internet (red de redes)

Físicamente, una conexión de redes de conmutación de paquetes interconectadas por ruteadores, junto con los protocolos TCP/IP permiten que la red funcione como una sola red virtual extensa. Cuando se escribe con mayúscula, Internet se refiere específicamente a la red global de Internet.

Internet

Conjunto de redes y ruteadores que abarca 61 países y utiliza los protocolos TCP/IP para formar una sola red virtual cooperativa. Internet conecta más de cuatro millones de computadoras.

Internet address (dirección Internet)

Ver dirección IP.

Internet Protocol

Ver IP.

Internet Society

Organización no lucrativa establecida para promover el uso de Internet. La Sociedad Internet es la organización anfitrión de IAB.

Internet worm (gusano de Internet)

Programa diseñado para viajar a través de Internet y reproducirse a sí mismo indefinidamente. Cuando un estudiante liberó el gusano de Internet, hizo que Internet y muchas computadoras conectadas quedaran fuera de operación por horas.

INTERNIC

(*INTERnet Network Information Center*) Organización que proporciona información sobre servicios de Internet y documentos de protocolos. Además, INTERNIC maneja el registro de las direcciones IP y los nombres de dominio.

Interoperability (interoperabilidad)

Capacidad del software y el hardware en máquinas diversas, de vendedores diferentes para comunicarse con éxito. Este término es el que describe mejor el objetivo del enlace de redes, cuya meta es definir un ambiente de red abstracto independiente del hardware, en el que sea posible construir una computación distribuida, a nivel del transporte de red, sin conocer los detalles de las tecnologías subyacentes.

IP

(*Internet Protocol*) Protocolo estándar que define a los datagramas IP como la unidad de información que pasa a través de una red de redes y proporciona las bases para el servicio de entrega de paquetes sin conexión y con el mejor esfuerzo. El IP incluye el control ICMP y los protocolos de mensaje de error como parte integral. El conjunto de protocolos completo se conoce frecuentemente como TCP/IP pues el TCP y el IP son los dos protocolos más importantes.

IP address (dirección IP)

Dirección de 32 bits asignada a cada anfitrión que participa en una red de redes TCP/IP. Una dirección IP es una abstracción de la dirección de hardware físico. Para hacer el ruteo eficiente, cada dirección IP se divide en parte en red y en parte en anfitrión.

IP datagram (datagrama IP)

Unidad básica de información que pasa a través de una red de redes TCP/IP. Un datagrama IP es a una red de redes lo que un paquete de hardware es a una red física. Contiene las direcciones de fuente y destino junto así como los datos.

IPng

(*Internet Protocol - the Next Generation*) Término aplicado a todas las actividades alrededor de la especificación y la estandarización de la próxima versión del IP. Ver también IPv6.

IPv4

Sinónimo de la versión actual del IP.

IPv6

Nombre oficial de la próxima versión del IP. Ver también IPng.

IRTF

(*Internet Research Task Force*) Grupo de personas que trabaja en problemas de investigación relacionados con el TCP/IP e Internet.

ISDN

(*Integrated Services Digital Network*) Nombre de los servicios de red digital que las compañías telefónicas tienen proyectado proporcionar.

ISO

(*International Organization for Standardization*) Organización internacional que bosqueja, discute, propone y especifica estándares para los protocolos de red. ISO es mejor conocido por su modelo de referencia de siete capas que describe la organización conceptual de los protocolos. Aun cuando se propuso como un conjunto de protocolos para la interconexión de sistemas abiertos, los protocolos OSI no han sido ampliamente aceptados a nivel comercial.

ISOC

Abreviatura de *Internet SOCiety*.

ISODE

(*ISO Development Environment*) Software que proporciona una interfaz ISO de protocolo de nivel de transporte por encima del TCP/IP. ISODE fue diseñado para permitir a los investigadores experimentar con los protocolos OSI de alto nivel sin requerir una red de redes que soporte los niveles inferiores del conjunto OSI.

ITU-TS

Abreviatura de *Telecommunication Section* de la *International Telecommunication Union.*

Karn's Algorithm (algoritmo de Karn)

Algoritmo que permite a los protocolos de transporte distinguir entre buenas y malas muestras de tiempo de viaje redondo para mejorar estimaciones de viaje redondo.

Kbps

(*Kilo Bits Per Second*) Medida de la cantidad de datos transmitidos. Ver también Mbps y baud.

Kramer

Término humorístico aplicado a un paquete de apariencia singular; el nombre se tomó de un programa de televisión.

LAN

(*Local Area Network*) Cualquier tecnología de red física diseñada para cubrir distancias cortas (del orden de unos cuantos cientos de metros). Por lo general las LAN operan a velocidades que van de los diez millones de bits por segundo a varios gigabits por segundo. Algunos ejemplos incluyen las redes Ethernet y las FDDI. Ver MAN y WAN.

level 1 (nivel 1)

Se trata de la comunicación a nivel de interfaz de hardware. El nombre se deriva del modelo de referencia de siete capas ISO. Las especificaciones del nivel 1 se refieren a las conexiones físicas, incluyendo los conectores, la configuración y los voltajes en los cables.

level 2 (nivel 2)

Se trata de la comunicación a nivel de enlace (por ejemplo, los formatos de trama) o a un nivel de enlace de las conexiones derivadas del modelo de referencia de siete capas de ISO. En las redes de área local, el nivel 2 hace referencia al formato de las direcciones y de trama física. Así, una dirección de nivel 2 es una dirección de trama física (por ejemplo, una dirección Ethernet).

level 3 (nivel 3)

Se trata de la comunicación a nivel de transporte derivado del modelo de referencia de siete capas de ISO. Para redes de redes TCP/IP, el nivel 3 se refiere al IP y el formato de datagrama IP. Así, una dirección de nivel 3 es una dirección IP.

LIS

(*Logical IP Subnet*) Grupo de computadoras conectado vía ATM que utiliza ATM como una red local aislada. Una computadora en una LIS no puede enviar un datagrama directamente a otra computadora de una LIS diferente.

little endian

Formato para almacenar o transmitir datos binarios en el que el octeto (bit) más significativo se encuentra primero. Ver *big endian*.

LLC

(*Logical Link Control*) Uno de los campos en un encabezado NSAP.

MAC

(*Media Access Control*) Se trata en general de los protocolos de hardware de bajo nivel utilizados para accesar una red en particular. El término *dirección MAC* se utiliza con frecuencia como sinónimo de *dirección física*.

mail bridge (puente de correo)

Término empleado informalmente como sinónimo de compuerta de correo.

mail exchanger (intercambiador de correo)

Computadora que acepta e-mail; algunas máquinas que intercambian correo lo envían hacia otras computadoras. DNS tiene un tipo de dirección separado para las máquinas que intercambian correo.

mail exploder (distribuidor de correo)

Parte de un sistema de correo electrónico que acepta correo y una lista de direcciones como entrada y envía una copia del mensaje a cada dirección de la lista. La mayor parte de los sistemas de correo electrónico incorpora un distribuidor de correo para permitir que los usuarios definan listas de correo locales.

mail gateway (compuerta de correo)

Máquina que se conecta a dos o más sistemas de correo electrónico (en especial a sistemas de correo diferentes o de dos redes distintas) y transfiere mensajes de correo entre ellas. Las compuertas de correo generalmente capturan un mensaje de correo completo, lo reformatean siguiendo las reglas del sistema de correo de destino y luego envían el mensaje.

MAN

(*Metropolitan Area Network*) Cualquiera de las nuevas tecnologías de red que operan a altas velocidades (por lo general, de cientos de megabits a varios gigabits por segundo) en distancias que abarcan un área metropolitana. Ver LAN y WAN.

Management Information Base

Ver MIB.

martians (marcianos)

Término humorístico que se aplica a paquetes que aparecen de manera inesperada en una red equivocada, frecuentemente debido a tablas de ruteo incorrectas.

maximum segment lifetime (máximo tiempo de vida de un segmento)

Se trata del lapso de tiempo más largo que un datagrama puede permanecer en Internet. Los protocolos utilizan MSL para garantizar un límite de tiempo a la permanencia de paquetes duplicados.

maximum segment size (máximo tamaño de un segmento)

Término utilizado en el TCP. El MMS es la mayor cantidad de datos que puede transmitirse en un segmento. El emisor y el receptor negocian el tamaño máximo de segmento al comienzo de una conexión.

maximum transfer unit

Ver MTU.

MBONE

(*Multicast BackBONE*) Acuerdo de cooperación entre localidades para enviar datagramas de multidifusión a través de Internet mediante el procedimiento de túneles.

Mbps

(*Millions of Bits Per Second o millones de bits por segundo*) Medida de la cantidad de datos transmitidos.

MIB

(*Management Information Base*) Conjunto de variables (bases de datos) que un ruteador mantiene corriendo SNMP. Los administradores pueden obtener o almacenar estas variables. El estándar actual es MIB-II.

MILNET

(*MILitary NETwork, Red Militar*) Originalmente parte de ARPANET, MILNET se separó en 1984.

MIME

(*Multipurpose Internet Mail Extensions*) Estándar utilizado para codificar datos como imágenes, en texto ASCII, para su transmisión a través del correo electrónico.

Mosaic

Programa que proporciona a los usuarios una interfaz gráfica para FTP, gopher y WWW.

mrouted

(*Multicast ROUTE Daemon*) Programa utilizado con un núcleo de multidifusión para establecer ruteo de multidifusión.

MSS

Abreviatura de *Maximum Segment Size* (*Máximo tamaño de segmento*).

MTU

(*Maximum Transfer Unit*) La mayor cantidad de datos que se puede transferir por unidad a través de una red física dada. El MTU lo determina el hardware de red.

multi-homed host (anfitrión múltiple)

Anfitrión que utiliza el TCP/IP y que tiene conexiones con dos o más redes físicas.

multicast (multidifusión)

Técnica que permite que copias de un solo paquete se transfieran a un subconjunto seleccionado de todos los posibles destinos. Algunos tipos de hardware (por ejemplo, Ethernet) soportan la multidifusión y permiten que una interfaz de red pertenezca a uno o más grupos de multidifusión. El IP soporta una capacidad de multidifusión de red de redes.

Nagle algorithm (algoritmo de Nagle)

Heurística autocronometrada que agrupa los datos que salen para mejorar el desempeño y evitar el síndrome de las ventanas tontas.

NAK

(*Negative Acknowledgement*) Respuesta de un recipiente de datos al emisor de los mismos para informarle que la transmisión no se realizó con éxito (por ejemplo, si los datos se alteraron por errores de transmisión). Por lo general, un NAK activa la retransmisión de datos perdidos.

NAP

(*Network Access Provider*) Compañía que proporciona conectividad con Internet.

name resolution (resolución de nombres)

Proceso de transformación de un nombre en su dirección correspondiente. El sistema de nombres de dominio proporciona un mecanismo para nombrar a las computadoras en las que los programas utilizan un servidor de nombres remoto para transformar el nombre de una máquina en una dirección IP.

NetBIOS

(*Network Basic Input Output System*) NetBIOS es la interfaz estándar para las redes en computadoras personales de IBM y compatibles. El TCP/IP incluye directivas que describen cómo transformar las operaciones NetBIOS en operaciones TCP/IP equivalentes.

network byte order (orden de octeto de red)

Estándar del TCP/IP para la transmisión de enteros que especifica que el bit más significativo aparece primero (big endian). Las máquinas emisoras son requeridas para hacer la traducción desde la representación de enteros local hacia el orden de octeto de red, y las máquinas receptoras son requeridas para hacer la traducción del orden de octeto de red a la representación de la máquina local.

network management (administración de red)

Ver MIB y SNMP.

NFS

(*Network File System*) Protocolo desarrollado por SUN Microsystems Incorporated que utiliza el IP a fin de permitir que un conjunto de computadoras coopere para accesar los sistemas de archivos de otras, como si éstas fueran locales.

NIC

(*Network Information Center*) Antecesor de INTERNIC.

NIST

(*National Institute of Standards and Technology*) Formalmente, National Bureau of Standards. NIST es una de las organizaciones de estándares dentro de Estados Unidos que establecen estándares para los protocolos de red.

NOC

(*Network Operations Center*) Originalmente, la organización en BBN que monitoreaba y controlaba varias redes que formaban parte de la red global de Internet. Ahora es utilizada por cualquier organización que administre una red.

NSAP

(*Network Service Access Point*) Formato de dirección que puede codificarse en 20 octetos. El ATM Forum recomienda utilizar direcciones NSAP.

NSF

(*National Science Foundation*) Dependencia gubernamental de Estados Unidos que inició algunas de las investigaciones y desarrollos de Internet.

NSFNET

(*National Science Foundation NETwork*) Se utiliza para describir la red de columna vertebral en Estados Unidos, que es administrada por la NSF.

OC3

Razón de transferencia de bits de aproximadamente 155 millones de bits por segundo que emplea conexiones de fibra óptica.

OSI

(*Open Systems Interconnection*) Se trata de los protocolos, específicamente estándares de ISO, para la interconexión de sistemas de computadoras cooperativos.

OSPF

(*Open Shortest Path First*) Protocolo de ruteo diseñado por la IETF.

packet (paquete)

Se trata, en términos generales, de cualquier bloque pequeño de datos enviado a través de una red de conmutación de paquetes.

PDN

(*Public Data Network*) Servicio de red ofrecido por una conocida compañía de comunicaciones. Por lo común, la PDN utiliza protocolos X.25.

PEM

(*Privacy Enchanced Mail*) Protocolo para cifrado de e-mail que se emplea para evitar que usuarios externos lean mensajes que viajan a través de una red de redes.

PING

(*Packet InterNet Groper*) Nombre de un programa utilizado con las redes de redes TCP/IP que se usa para probar la accesibilidad de un destino, enviando una solicitud de eco ICMP y esperando una respuesta. El término es utilizado ahora como verbo, por ejemplo, "haz ping al anfitrión A para ver si está activo".

port (puerto)

Ver puerto de protocolo.

positive acknowledgement (reconocimiento positivo)

Ver ACK.

ppp

(*Point to Point Protocol*) Protocolo para enmarcar al IP cuando se envía a través de una línea serial. Ver también SLIP.

promiscuous ARP

Ver proxy ARP.

protocol (protocolo)

Descripción formal de formatos de mensajes y reglas que dos o más máquinas deben seguir para intercambiar mensajes. Los protocolos pueden describir detalles de bajo nivel de las interfaces de máquina a máquina (por ejemplo, el orden en el que los bits de un octeto se envían a través de un cable) o del intercambio entre programas de aplicación (por ejemplo, la forma en que un programa transfiere un archivo a través de una red de redes). La mayor parte de los protocolos incluye descripciones intuitivas de las interacciones esperadas así como especificaciones más formales, utilizando modelos de máquinas de estado finito.

protocol port (puerto de protocolo)

Abstracción que los protocolos de transporte del TCP/IP utilizan para distinguir entre varios destinos sin una computadora anfitrión dada. Los protocolos TCP/IP identifican puertos mediante el uso de enteros positivos pequeños. Usualmente el sistema operativo permite a un programa de aplicación especificar qué puerto desea utilizar. Algunos puertos se reservan para servicios estándar (por ejemplo, el correo electrónico).

proxy ARP (asignación de poder ARP)

Técnica mediante la cual una máquina, usualmente un ruteador, responde una solicitud ARP proyectada para otro proporcionando su propia dirección física. Al pretender ser otra máqui-

na, el ruteador acepta la responsabilidad del envío de paquetes hacia adelante. El propósito de la asignación de poder ARP (proxy ARP) es permitir que una localidad utilice una sola dirección de red IP con múltiples redes físicas.

pseudo header (pseudo encabezado)

Información de direcciones IP de fuente y destino enviadas en el encabezado IP, pero incluidas en un TCP o en suma de verificación UDP.

public key encryption (cifrado de clave pública)

Técnica de cifrado que genera claves de cifrado por pares. Un elemento del par se mantiene en secreto y el otro se publica.

PUP

(*Parc Universal Packet*) En el sistema de red de redes desarrollado por Xerox Corporation, un PUP es la unidad fundamental de transferencia, como lo es un datagrama IP en una red de redes TCP/IP. La palabra se derivó del nombre del laboratorio en el que se desarrollo la red de redes Xerox, el Palo Alto Research Center (PARC).

push (empujar)

Operación que realiza una aplicación en una conexión TCP para forzar a que un dato se envíe inmediatamente. Un bit en el encabezado de segmento marca el dato que se está empujando.

RARP

(*Reverse Address Resolution Protocol*) Protocolo TCP/IP que una máquina sin disco utiliza al arrancar para encontrar su dirección IP. La máquina difunde una solicitud que contiene su dirección de hardware físico y un servidor responde enviando a la máquina su dirección IP. RARP toma su formato de nombre y mensaje de otro protocolo de resolución de dirección IP, ARP.

RDP

(*Reliable Datagram Protocol*) Protocolo que proporciona un servicio de datagramas confiable por encima del servicio de datagramas no confiable que proporciona el IP. RDP no está entre los protocolos TCP/IP implantados más ampliamente.

reassembly (reensamblado)

Proceso de reunir todos los fragmentos de un datagrama IP y utilizarlos para crear una copia del datagrama original. El destino final realiza el reensamblado.

redirect (redireccionamiento)

Mensaje ICMP enviado de un ruteador a un anfitrión en una red local para instruir al anfitrión a que cambie de ruta,

repeater (repetidor)

Dispositivo de hardware que extiende las LAN. Un repetidor copia señales eléctricas de una red física a otra. No son muy populares.

reverse path forwarding (envío por trayectoria inversa)

Técnica utilizada para propagar paquetes de difusión. El IP utiliza el envío por trayectorias inversas para propagar difusión de subred.

RFC

(*Request For Comments*) Nombre de una serie de notas que contienen estudios, mediciones, ideas, técnicas y observaciones, así como estándares de protocolo TCP/IP aceptados. Los RFC están disponibles en línea.

RIP

(*Routing Information Protocol*) Protocolo utilizado para difundir información de ruteo dentro de un sistema autónomo. El RIP deriva de un protocolo del mismo nombre desarrollado originalmente por Xerox.

RJE

(*Remote Job Entry*) Servicio que permite el control de una tarea determinada desde una localidad remota.

rlogin

(Remote LOGIN) Protocolo de acceso remoto desarrollado para UNIX por Berkeley. rlogin ofrece esencialmente el mismo servicio que TELNET.

ROADS

(*Running Out of ADdress Space*) Se refiere al inminente agotamiento del espacio de direcciones de la clase B.

route (ruta)

En general, una ruta es la trayectoria que el tráfico de red toma de su fuente a su destino. En una red de redes TCP/IP, cada datagrama IP es ruteado de manera independiente; las rutas pueden cambiar dinámicamente.

routed

(*Route Daemon*) Programa concebido para UNIX que implementa el protocolo RIP. Se pronuncia "route-d".

router (ruteador)

Computadora dedicada, de propósito especial, que se conecta a dos o más redes y envía paquetes de una red a otra. En particular, un ruteador IP envía datagramas IP entre las redes a las que está conectado. Un ruteador utiliza las direcciones de destino en un datagrama para decidir el próximo salto al que enviará el datagrama. Los investigadores originalmente utilizaban el término *compuerta IP*.

RPC

(*Remote Procedure Call*) Tecnología en la que un programa invoca servicios a través de una red haciendo modificaciones en los procedimientos de llamada. El protocolo NFS utiliza un tipo específico de RPC.

RS232

Estándar de EIA que especifica las características eléctricas de las interconexiones de baja velocidad entre terminales y computadoras o entre dos computadoras. Aun cuando el estándar más utilizado es RS232C, la mayoría de la gente se refiere a él como RS232.

RTO

(*Round Trip time-Out*) Retardo utilizado antes de una retransmisión. El TCP calcula RTO como una función del tiempo y la variación del viaje redondo actual.

RTT

(*Round Trip Time*) Medición del retardo entre dos anfitriones. El tiempo de viaje redondo consiste en el tiempo total que toma a un solo paquete o datagrama dejar una máquina, alcanzar otra y regresar. En la mayor parte de las redes de conmutación de paquetes, los retardos varían en función del tráfico y el congestionamiento de la red. Así, la medición del tiempo de viaje redondo es un promedio, que puede tener una desviación estándar alta.

SACK

(Selective ACKnowledgement) Mecanismo de acuse de recibo utilizado con los protocolos de ventanas deslizantes que permite al receptor hacer el reconocimiento de los paquetes recibidos fuera de orden, pero dentro de la ventana deslizante actual. También se le conoce como acuse de recibo extendido. Compárelo con el esquema de acuse de recibo acumulativo utilizado por el TCP.

segment (segmento)

Unidad de transferencia enviada del TCP en una máquina al TCP en otra. Cada segmento contiene parte de un flujo de octetos, que son enviados entre las máquinas, así como campos adicionales que identifican la posición actual en el flujo y una suma de verificación que asegura la validez de los datos recibidos.

selective acknowledgement (reconocimiento selectivo)

Ver SACK.

self-identifying frame (trama autoidentificable)

Cualquier paquete o trama de red que incluya un campo para identificar el tipo de datos que se están transportando. Ethernet utiliza tramas que se autoidentifican, pero ATM no.

SGMP

(*Simple Gateway Monitoring Protocol*) Antecesor de SNMP.

signaling (señalización)

Término de telefonía que hace referencia al proceso por medio del cual ciertos protocolos establecen un circuito.

silly window syndrome (síndrome de las ventanas tontas)

Situación que se puede originar en el TCP, en la cual el receptor anuncia repetidamente una ventana pequeña y el emisor envía continuamente un segmento pequeño para llenarla. La transmisión resultante de pequeños segmentos hace ineficiente el uso del ancho de banda.

SIP

(*Simple IP*) Propuesta original que formó la base para IPng.

SIPP

(*SIP Plus*) Extensión de SIP que se ha propuesto como IPng. IETF espera que SIPP sea apoyada. Ver IPv6.

sliding window (ventana deslizante)

Característica de los protocolos que permite a un emisor transmitir más de un paquete de datos antes de recibir un acuse de recibo. Luego de recibir el acuse de recibo para el primer paquete enviado, el emisor "desliza" la ventana del paquete y envía otro. El número de paquetes u octetos que salen, se conoce como *tamaño de la ventana*; al incrementarse el tamaño de la ventana, aumenta el desempeño.

SLIP

(*Serial Line IP*) Protocolo de proceso de tramas utilizado para envíos IP a través de una línea serial. SLIP es popular cuando hace envíos IP en una línea telefónica de marcación. Ver PPP.

slow-start (arranque lento)

Esquema para evitar congestionamientos en el TCP, en donde el TCP incrementa el tamaño de sus ventanas conforme llegan los ACK. El término no es muy preciso debido a que el arranque lento puede lograr un alto desempeño utilizando un incremento exponencial.

SMDS

(*Switched Multimegabit Data Service*) Servicio de paquetes sin conexión, desarrollado por una compañía telefónica local.

SMTP

(*Simple Mail Transfer Protocol*) Protocolo estándar del TCP/IP para transferir mensajes de correo electrónico de una máquina a otra. SMTP especifica cómo interactúan dos sistemas de correo y el formato de los mensajes de control que intercambian para transferir el correo.

SNA

(*System Network Architecture*) Nombre aplicado a una arquitectura y a una clase de productos de red ofrecidos por IBM. SNA no interopera con el TCP/IP.

SNAP

(*SubNetwork Attachment Point*) Pequeño encabezado añadido a los datos cuando son enviados a través de una red que no tiene tramas que se autoidentifiquen. El encabezado SNAP especifica el tipo de datos.

SNMP

(*Simple Network Monitoring Plotocol*) Protocolo estándar utilizado para monitorear anfitriones, ruteadores y las redes a las que están conectados. La segunda versión del protocolo se conoce como SNMPv2. Ver también MIB.

SOA

(*Start Of Authority*) Clave de acceso utilizada con DNS para denotar el comienzo de los registros para los que un servidor particular es la autoridad. Otros registros en el servidor son reportados como respuestas no autorizadas.

socket

Abstracción proporcionada por el sistema operativo UNIX que permite a un programa de aplicación accesar los protocolos TCP/IP.

source quench (apagado de fuente)

Técnica de control de congestionamientos en la que una máquina que enfrenta un congestionamiento envía un mensaje hacia la fuente que envía los paquetes solicitando que deje de transmitir. En una red de redes TCP/IP, los ruteadores utilizan el apagado de fuente ICMP para detener o reducir la transmisión de datagramas IP.

source route (ruta de fuente)

Ruta que se determina por la fuente. En el IP, una ruta de fuente consiste en una lista de ruteadores que un datagrama debe visitar; el ruteador se especifica como una opción IP. La ruta de fuente se utiliza casi siempre para depuración.

SPF

(*Shortest Path First*) Clase de protocolos de actualización de ruteo que utilizan el algoritmo de Dijkstra para calcular las rutas más cortas.

STD

(*STanDard*) Designación que se utiliza para especificar que un RFC en particular describe un protocolo estándar.

subnet addressing (direccionamiento de subred)

Extensión de un esquema de direccionamiento IP que permite a una localidad utilizar una sola dirección IP para varias redes físicas. Al exterior de la localidad el uso de direccionamiento de subred continúa el ruteo como es usual, dividiendo la dirección de destino en una parte de red y en una parte local. Los anfitriones y los ruteadores dentro de una localidad utilizan el direccionamiento de subred para interpretar la parte local de la dirección, dividiéndola, una parte como red física y otra como anfitrión.

SubNetwork Attachment Point

Ver SNAP.

supernet addressing

Otro nombre para el ruteo Classless Inter-Domain.

SWS

Ver *Silly Window Syndrome.*

SYN

(*SYNchronizing segment*) Primer segmento enviado por el protocolo TCP, se utiliza para sincronizar los dos extremos de una conexión en la preparación de una conexión abierta.

T3

Nombre en telefonía para un protocolo utilizado en líneas de velocidad DS3. El término se emplea con frecuencia (erróneamente) como un sinónimo de DS3.

TCP

(*Transmission Control Protocol*) Protocolo de nivel de transporte TCP/IP estándar que proporciona el servicio de flujo confiable full duplex y del cual dependen muchas aplicaciones. El TCP/IP permite que el proceso en una máquina envíe un flujo de datos hacia el proceso de otra. El TCP está orientado a la conexión en el sentido de que, antes de transmitir datos, los participantes deben establecer la conexión. Todos los datos viajan en segmentos TCP, en donde cada viaje se realiza a través de Internet en un datagrama IP. El conjunto de protocolos completo se conoce frecuentemente como TCP/IP debido a que el TCP y el IP son los dos protocolos más importantes.

TCP/IP Internet Protocol Suite

Nombre oficial de los protocolos TCP/IP.

TDM

(*Time Division Multiplexing*) Técnica utilizada para multiplexar varias señales en un solo canal de transmisión de hardware, lo que permite que cada señal utilice el canal por un corto tiempo, antes de dar el paso a la próxima. Ver también FDM.

TDMA

(*Time Division Multiple Access*) Método de acceso de red en el que el tiempo se divide en ranuras y cada nodo de la red es asignado a una de las ranuras. Dado que todos los nodos utilizan TDMA, deben sincronizarse con exactitud (incluso si la red introduce retardos de propagación entre ellos), la tecnología TDMA es difícil de diseñar y el equipo es costoso.

TELNET

Protocolo estándar del TCP/IP para servicio de terminal remota. TELNET permite al usuario en una localidad interactuar con un sistema de tiempo compartido remoto como si el teclado y el monitor del usuario estuvieran conectados a la máquina remota.

TFTP

(*Trivial File Transfer Protocol*) Protocolo estándar TCP/IP para transferencia de archivos con capacidad mínima y sobrecarga mínima. El TFTP depende sólo del servicio de entrega de datagramas sin conexión y no confiable (UDP), de este modo, puede utilizarse en máquinas como las estaciones de trabajo que conservan el software en ROM y lo utilizan para arrancar.

thicknet

Nombre del cable coaxial grueso utilizado originalmente en Ethernet. Véase thinnet y 10Base-T.

thinnet

Se trata del cable coaxial delgado y flexible empleado en Ethernet. Ver thicknet y 10Base-T.

TLI

(*Transport Layer Interface*) Alternativa para la interfaz socket definida para el System V de UNIX.

TLV encoding

Cualquier formato de representación que codifica a cada característica con un campo de tipo seguido por un campo de longitud y un valor. Las opciones IP frecuentemente utilizan codificación TLV.

tn3270

Versión de TELNET para usarse con terminales IBM 3270.

token ring

Cuando se utiliza en sentido genérico, se refiere a un tipo de tecnología de red que controla el acceso de medios pasando un paquete distintivo, llamado token (ficha), de máquina en máquina. Una máquina puede transmitir un paquete sólo cuando tiene la ficha (token). Cuando se utiliza con un sentido específico, se refiere al hardware de red token ring producido por IBM.

TOS

(*Type Of Service*) Cada encabezado de datagrama IP incluye un campo que permite al emisor especificar el tipo de servicio deseado. En la práctica, pocos ruteadores utilizan TOS cuando eligen una ruta.

TP-4

Protocolo diseñado por ISO, similar al TCP.

traceroute

Programa que registra la trayectoria hacia un destino. Traceroute envía una secuencia de datagramas con el Time-To-Live puesto a 1, 2, etc., y utiliza mensajes ICMP TIME EXCEEDED que retornan para determinar rutas a lo largo de la trayectoria.

trailers (remolques)

Método no convencional para encapsular datagramas IP para transmisión mediante el cual la información del encabezado se coloca al final del paquete. Los trailers se han utilizado con Ethernet para ayudar a alinear datos en los límites de página. AAL5 de ATM utiliza trailers.

transceiver (transceptor)

Dispositivo que conecta una interfaz de anfitrión a una red de área local (por ejemplo, Ethernet). Los transceptores de Ethernet contienen electrónica analógica que aplica señales a los cables y percibe colisiones.

TRPB

(*Truncated Reverse Path Broadcast*) Técnica utilizada para propagar datagramas de multidifusión.

TTL

(Time To Live) Técnica utilizada en los sistemas de entrega con el mejor esfuerzo para evitar que los paquetes permanezcan en un ciclo por tiempo indefinido. Por ejemplo, a cada datagrama IP se le asigna un tiempo límite entero cuando se crea. Cada ruteador decrementa el campo de tiempo límite cuando el datagrama llega, y un ruteador descarta cualquier datagrama si el contador de tiempo límite llega a cero.

tunneling (encapsulado)

Técnica mediante la cual un paquete es encapsulado en un protocolo de alto nivel y pasa a través del sistema de transporte. El MBONE aplica la técnica de túnel a cada datagrama de multidifusión IP en lugar de aplicarla a un datagrama convencional IP.

twisted pair Ethernet (par trenzado Ethernet)

Esquema de cableado de Ethernet que utiliza pares de cables trenzados desde cada computadora hacia un concentrador. Ver thicknet y thinnet.

type of service routing (tipo de ruteo de servicio)

Esquema de ruteo en el que la elección de una trayectoria depende de las características de la tecnología de red subyacente, así como de la trayectoria más corta hacia el destino. En principio, el Protocolo Internet (IP) se adapta al tipo de servicio de ruteo debido a que los datagramas contienen un campo de solicitud de tipo de servicio. En la práctica pocos ruteadores cumplen con el tipo de servicio solicitado.

UART

(*Universal Asynchronous Receiver and Transmitter*) Dispositivo electrónico que consiste en un solo chip que puede enviar o recibir caracteres en líneas de comunicación serial asíncronas que utilizan RS232. Los UART son flexibles pues tienen un control sobre las líneas que permite a los diseñadores seleccionar parámetros como velocidad de transmisión, paridad, número de bits de parada y control de módem. Los UART se ocupan en terminales, módems y en tarjetas de entrada/salida de computadoras que conectan a la computadora con las terminales.

UCBCAST

Ver difusión Berkeley.

UDP

(*User Datagram Protocol*) Protocolo estándar TCP/IP que permite a un programa de aplicación en una máquina enviar un datagrama hacia el programa de aplicación en otra máquina. El UDP utiliza el Protocolo de Internet (IP) para entregar datagramas. Conceptualmente la diferencia importante entre los datagramas UDP y los IP es que el UDP incluye un número de puerto de protocolo, lo que permite al emisor distinguir entre varios programas de aplicación en una máquina remota dada. En la práctica el UDP también incluye una suma de verificación opcional en el datagrama que se está enviando.

unicast (unidifusión)

Método mediante el cual un paquete se envía a un solo destino. La mayor parte de los datagramas IP se manda vía unidifusión. Ver multicast (multidifusión).

universal time (tiempo universal)

Se trata del estándar internacional que inicialmente se conoció como Hora del Meridiano de Greenwich. También, se le conoce como tiempo coordinado universal.

urgent data (dato urgente)

Método utilizado en el TCP para enviar datos fuera de banda. Un receptor procesa los datos urgentes en cuanto los recibe.

URL

(*Uniform Resource Locator*) Cadena que proporciona la localización de una parte de la información. La cadena comienza con el tipo de protocolo (por ejemplo, el FTP) seguido por la identificación de información específica (por ejemplo, el nombre de dominio de un servidor y el nombre de la trayectoria hacia un archivo en el servidor).

UUCP

(*Unix to Unix Copy Program*) Programa de aplicación desarrollado a mediados de los setenta para la versión 7 de UNIX que permite a un sistema de tiempo compartido de UNIX copiar archivos hacia, o desde otro sistema de tiempo compartido de UNIX en un solo enlace (normalmente, una línea de marcación). Dado que el UUCP es la base para la transferencia de correo electrónico en UNIX, el término se utiliza de manera general para hacer referencia a la transferencia de correo en UNIX.

vBNS

Red de columna vertebral de Internet de 155 Mbps, planeada para desarrollarse en Estados Unidos durante 1995.

VPI/VCI

(*Virtual Path Identifier y Virtual Circuit Identifier*) Los dos campos de un identificador de conexión ATM; cada conexión que abre un anfitrión tiene un par VPI/VCI único.

vector distance (vector-distancia)

Clase de protocolo de actualización de ruteo que utiliza un algoritmo distribuido de la trayectoria más corta, en la que cada ruteador participante envía a sus vecinos una lista de redes que puede alcanzar y la distancia hasta cada red. Compararlo con SPF.

very high speed Backbone Network Service

Ver vBNS.

virtual circuit (circuito virtual)

Abstracción básica proporcionada por un protocolo orientado a la conexión como el TCP. Una vez que un circuito virtual se ha creado, se establece un efecto hasta que se desactiva explícitamente.

WAN

(*Wide Area Network*) Cualquier tecnología de red que abarca distancias geográficas extensas. También llamadas redes de gran alcance, las WAN actualmente operan a bajas velocidades y tienen retardos significativamente mayores que las redes que operan sobre distancias cortas. Ver LAN y MAN.

well-known port (puerto bien conocido)

Cualquier conjunto de números de puerto de protocolo preasignados para usos específicos por los protocolos de nivel de transporte (es decir, TCP y UDP). Cada servidor está en la lista de un puerto bien conocido, de este modo, sus clientes pueden localizarlo.

window (ventana)

Ver sliding window (ventana deslizante).

Winsock

Interfaz de un programa de aplicación que permite a un programa, que utiliza socket, llamar a Windows de Microsoft para correr.

working group (grupo de trabajo)

El término se aplica a un comité de IETF. Cada grupo de trabajo es responsable de un protocolo particular o del diseño de algún aspecto.

World Wide Web

Servicio de información a gran escala que permite a un usuario buscar información. WWW ofrece un sistema de hipermedios que puede almacenar información como texto, gráficos, audio, etcétera.

WWW

Ver *World Wide Web.*

X

Ver X-Window System.

X.25

Protocolo estándar de ITU-TS para el servicio de red a nivel de transporte. Es posible formar túneles a través de X.25. X.25 es más popular en Europa.

X25NET

X.25 *NETwork* Servicio ofrecido por CSNET que transfiere tráfico IP entre una localidad suscrita e Internet por medio del X.25.

X.400

Protocolo de ITU-TS para el correo electrónico.

XDR

(*eXternal Data Representation*) Estándar para una representación de datos independiente de la máquina. Para utilizar XDR, un emisor traduce desde la representación de una máquina local a la representación externa estándar y un receptor traduce de la representación externa a la representación de la máquina local.

X-Window System

Sistema de software desarrollado en el MIT para mostrar y administrar las salidas en presentaciones de mapas de bits. Cada ventana consiste en una región rectangular de la pantalla que contiene texto o gráficos de un programa remoto. Un programa especial, llamado administrador de ventana, permite al usuario crear, mover, sobreponer y suprimir ventanas.

zone of authority (zona de autoridad)

Término utilizado en el sistema de nombres de dominio que hace referencia a un grupo de nombres para los que un servidor de nombres dado es una autoridad. Cada zona de autoridad debe proporcionarse con dos servidores de nombres que no tengan puntos comunes de posibles fallas.

Bibliografía

ABRAMSON, N. [1970], The ALOHA System – Another Alternative for Computer Communications. *Proceedings of the Fall Joint Computer Conference.*

ABRAMSON, N. y F. KUO (EDS.) [1973], Computer Communication *Netwoorks*, Prentice Hall, Englewood Cliffs, New Jersey.

ANDREWS, D. W., y G. D. SHULTZ [1982], A Token-Ring Architecture for Local Area Networks: An Update, *Proceedings of Fall 82 COMPCON*, IEEE.

BALL, J. E., E. J. BURKE, I. GERTNER, K. A. LANTZ, y R. F. RASHID [1979], Perspectives on Message-Based Distributed Computing, *IEEE Computing Networking Symposium*, 46-51.

BBN [1981], A History of the ARPANET: The First Decade, *Technical Report* Bolt, Beranek, y Newman, Inc.

BBN [Diciembre1981], Specification for the Interconnection of a Host y an IMP (revised), *Technical Report 1822*, Bolt, Beranek, y Newman, Inc.

BIAGIONI E., E. COOPER, y R. SANSOM [marzo 1993], Designing a Practical ATM LAN, *IEEE Network*, 32-39.

BERTSEKAS D. y R. GALLAGER [1987], *Data Networks*, Prentice-Hall, Englewood Cliffs, New Jersey.

BIRRELL, A., y B. NELSON [febrero 1984], Implementing Remote Procedure Calls, *ACM Transactions on Computer Systems*, 2(1), 39-59.

BOGGS, D., J. SHOCH, E. TAFT, y R. METCALFE [abril 1980], Pup: An Internetwork Architecture, *IEEE y Transactions on Communications.*

BORMAN, D., [abril 1989], Implementing TCP/IP on a Cray Computer, *Computer Communication Review*, 19(2), 11-15.

BROWN, M., K. KOLLING, y E. TAFT [noviembre1985], The Alpine File System, *ACM Transactions on Computer Systems*, 3(4), 261-293.

BROWNBRIDGE D., L. MARSHALL., y B. RANDELL [diciembre 1982], The Newcastle Connections or UNIXes of the World Unite!, *Software – Practice and Experience*, 12(12), 1147-1162.

CASNER, S., y S. DEERING [julio 1992], First IETF Internet Audiocast, *Computer Communications Review*, 22(3), 92-97.

CERF, V., y E. CAIN [octubre 1983], The DOD Internet Architecture Model, *Computer Networks.*

CERF, V., y R. KAHN [mayo 1974], A Protocol for Packet Networks Interconnection, *IEEE Transactions of Communications*, Com-22(5).

CERF, V. [octubre 1989], A History of the ARPANET, *ConneXions, The Interoperability Report*, 480 San Antonio Rd, Suite 100, Mountain View, California.

CHERITON, D. R. [1983], Local Networking and Internetworking in the V-System, *Proceedings of the Eighth Data Communications Symposium.*

CHERITON, D. R. [abril 1984], The V Kernel: A Software Base for Distributed Systems, *IEEE Software*, 1(2), 19-42.

CHERITON, D. [agosto 1986], VMTP: A Transport Protocol for the Next Generation of Communication Systems, *Proceedings of ACM SIGCOMM' 86*, 406-415.

CHERITON, D., y T. MANN [mayo 1984], Uniform Access to Distributed Name Interpretation in the V-System, *Proceedings IEE Fourth International Conference on Distributed Computing Systems*, 290-297.

CHESSON, G. [junio 1987], Protocol Engine Design, *Proceedings of the 1987 Summer USENIX Conference*, Phoenix, AZ.

CLARK, D. [diciembre 1985], The structure of Systems Using Upcalls, *Proceedings of the Tenth ACM Symposium on Operating Systems Principles*, 171-180.

CLARK, D., M. LAMBERT, y L. ZHANG [agosto 1987], NETBLT: A High Throughput Transport Protocol, *Proceedings of ACM SIGCOMM* '87.

CLARK, D., V. JACOBSON, J. ROMKEY, y H. SALWEN [junio 1989]. An Analysis of TCP Processing Overhead, *IEEE Communications*, 23-29.

COHEN, D., [1981], On Holy Wars and a Plea for Peace, *IEEE Computer*, 48-54.

COMER, D. E. y J. T. KORB [1983], CSNET Protocol Software: The Ip-to-X25 Interface, *Computer Communications Review*, 13(2).

COMER, D. E. [1984], *Operating System Design – The XINU Approach*, Prentice-Hall, Englewood Cliffs, New Jersey,

COMER, D. E. [1987], *Operating System Design Vol. II. – Internetworking With XINU*, Prentice-Hall, Englewood Cliffs, New Jersey.

COMER, D. E. y D. L. STEVENS [1994] *Internetworking With TCP/IP Volume II – Design, Implementation, and Internals*, 2nd edition, Prentice-Hall, Englewood Cliffs, New Jersey.

COMER, D. E. y D. L. STEVENS [1993] *Internetworking With TCP/IP Volume III – Client-Server Programming And Applications, BSD socket version*, Prentice-Hall, Englewood Cliffs, New Jersey.

COMER, D. E. y D. L. STEVENS [1994] *Internetworking With TCP/IP Volume III – Client-Serve Programming And Applications, AT&T TLI version*, Prentice-Hall, Englewood Cliffs, New Jersey.

COMER, D. E., T. NARTEN, y R. YAVATKAR [abril 1987], The Cypress Network: A Low-Cost Internet Connection Technology, *Technical Report TR-653*, Purdue University, West Lafayette, IN.

COMER, D. E., T. NARTEN, y R. YAVATKAR [1987], The Cypress Coaxial Packet Switch, *Computer Networks and ISDN Systems*, vol., 14:2-5, 383-388.

COTTON, I, [1979], Technologies for Local Area Computer Networks, *Proceedings of the Local Area Communications Network Symposium.*

CROWLEY, T., H. FORSDICK, M. LANDAU, y V. TRAVERS [junio 1987], The Diamond Multimedia Editor, *Proceedings of the 1987 Summer USENIX Conference*, Phoenix, AZ.

DALAL Y. K. y R. S. PRINTIS [1981], 48-Bit Absolute Internet and Ethernet Host Numbers, *Proceedings of the Seventh Data Communications Symposium.*

DEERING S.E., y D. R. CHERITON [mayo 1990]. Multicast Routing in Datagram Internetworks and Extended LANs. *ACM Transactions on Computer Systems*, 8(2), 85-110.

DEERING, S., D. ESTRIN, D. FARINACCI, V. JACOBSON, C-G LIU y L. WEI [agosto 1994], An Architecture for Wide-Area Multicasting Routing, *Proceedings of ACM SIGCOMM '94*, 126-135.

DEENING, P. J., [septiembre-octubre 1989], *The Science of Computing: Worldnet*, In American Scientist, 432-434.

DENNING, P. J., [noviembre-diciembre 1989], *The Science of Computing: The ARPANET After Twenty Years*, in American Scientist, 530-534.

DE PRYCKER, M. [1993] *Asynchronous Transfer Mode Solution for Broadband ISDN*, 2da edición, Ellis Horwood, UK.

DIGITAL, EQUIPMENT CORPORATION, INTEL CORPORATION, y XEROX CORPORATION [septiembre 1980], *The Ethernet: A Local Area Network Data Link Layer and Physical Layer Specification.*

DION, J. [octubre 1980]. The Cambridge File Server, *Operating Systems Review*, 14(4), 26-35.

DRIVER, H., H. HOPEWELL, y J. IAQUINTO [septiembre 1979], How the Gateway Regulates Information Control, *Data Communications.*

EDGE, S. W. [1979], Comparison of the Hop-by-Hop and Endpoint Approaches to Network Interconnection, in *Flow Control in Computer Networks*, J-L. GRANGE and M. GIEN (EDS.), North-Holland. Amsterdam, 359-373.

EDGE, S. [1983], An Adaptive Timeout Algorithm for Retransmission Across a Packet Switching Network, *Proceedings of ACM SIGCOMM '83.*

ENSLOW, P. [enero 1978], What is a 'Distributed' Data Processing System? *Computer*, 13-21.

ERIKSSON, H. [agosto 1994] MBONE: The Multicast Backbone, *Communications of the ACM*, 37(8), 54-60.

FALK, G. [1983]. The Structure and Function of Network Protocols, in *Computer Communications, Volume I: Principles*. CHOU, W. (ED.), Prentice-Hall, Englewood Cliffs, New Jersey.

FARMER, W, D., y E. E., NEWHALL [1969], An Experimental Distributed Switching System to Handle Bursty Computer Traffic, *Proceedings of the ACM Symposium on Probabilistic Optimization of Data Communication Systems*, 1-33.

FCCSET [noviembre 1987], A Research and Development Strategy for High Performance Computing, *Report from the Executive Office of the President and Office of Science and Technology Policy.*

FEDOR, M. [junio 1988], GATED: A Multi-Routing Protocol Daemon for UNIX, *Proceedings of the 1988 Summer USENIX Conference*, San Francisco, California.

FEINLER, J., O. J. JACOBSEN, y M. STAHL [diciembre 1985], *DDN Protocol Handbook Volume Two, DARPA Internet Protocols*, DDN Network Information Center, SRI International, 333 Ravenswood Avenue, Room EJ291, Menlo Park, California.

FLOYD, S. y V. JACOBSON [agosto 1993], Random Early Detection Gateways for Congestion Avoidance, *IEEE/ACM Transactions on Networking*, I(4).

FRANK, H., y W. CHOU [1971], Routing in Computer Networks, *Networks*, 1(1), 99-112.

FRANK, H., y J. FRISCH [1971], *Communication, Transmission, and Transportation Networks*. Addison-Wesley, Reading, Massachusetts.

FRANTA, W. R., y I. CHLAMTAC [1981], *Local Networks*, Lexington Books, Lexington, Massachusetts.

FRICC [mayo 1989], *Program Plan for the National Research and Education Network*. Federal Research Internet Coordinating Committee, US Department of Energy, Office of Scientific Computing report ER-7.

FRIDRICH, M., y W. OLDER [diciembre 1981], The Felix File Server, *Proceedings of the Eigth Symposium on Operating Systems Principles*, 37-46.

FULTZ, G. L. y L. KLEINROCK [junio 14-16, 1971], Adaptative Routing Techniques for Store-and-Forward Computer Communication Networks, presented at *IEEE International Conference on Communications*, Montreal, Canada.

GERLA, M., y L. KLEINROCK [abril 1980], Flow Control: A Comparative Survey, *IEEE Transactions on Communications.*

GOSIP [abril 1989], U.S. Government Open Systems Interconnection Profile (GOSIP) version 2.0, GOSIP Advanced Requirements Group, National Institute of Standards and Technology (NIST).

GRANGE, J-L-. y M. GIEN (EDS.) [1979], *Flow Control in Computer Networks*, North-Holland, Amsterdam.

GREEN, P. E. (ED.) [1982], *Computer Network Architectures and Protocols*, Plenum Press, New York.

HINDEN, R., J. HAVERTY, y A. SHELTZER [septiembre 1983]. The DARPA Internet: Interconnecting Heterogeneous Computer Networks with Gateways, *Computer.*

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION [junio 1986a]. Information processing systems — Open Systems Interconnection — *Transport Service Definition*, International Standard number 8072, ISO, Switzerland.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION [julio 1986b], Information processing systems — Open Systems Interconnection — *Connection Oriented Transport Protocol Specification*, International Standard number 8073, ISO, Switzerland

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION [mayo 1987a], Information processing systems — Open Systems Interconnection — *Specification of Basic Specification of Abstract Syntax Notation One (ASN.1).* International Standard number 8824, ISO, Switzerland.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION [mayo 1987b], Information processing systems — Open Systems Interconnection — *Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1),* International Standard number 8825, ISO, Switzerland.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION [mayo 1988a], Information processing systems — Open Systems Interconnection — *Management Information Service Definition. Part 2: Common Management Information Service.* Draft International Standard number 9595-2, ISO, Switzerland.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION [mayo 1988a], Information processing systems — Open Systems Interconnection — *Management Information Protocol Definition, Part 2: Common Management Information Protocol,* Draft International Standard number 9596-2.

JACOBSEN, O. J. (PUBLISHER) [1987-], ConneXions, The Interoperability Report, *Interop Company*, a division of Softbank Exposition and Conference Company, Foster City, California.

JACOBSON, V. [agosto 1988], Congestion Avoidance and Control, *Proceedings ACM SIGCOMM'88.*

JAIN, R. [enero 1985], On Caching Out-of-Order Packets in Window Flow Controlled Networks. *Technical Report*, DEC-TR-342, Digital Equipment-Corporation.

JAIN, R. [marzo 1986], Divergence of Timeout Algorithms for Packet Retransmissions. *Proceedings Fifth Annual International Phoenix Conference on Computers and Communications*, Scottsdale, AZ.

JAIN, R. [octubre 1986], A Timeout-Based Congestion Control Scheme for Window Flow-Controlled Networks, *IEEE Journal on Selected Areas in Communications*, Vol. SAC-4, no. 7.

JAIN, R. [mayo 1992], Myths About Congestion Management in High-speed Networks, *Internetworking: Research and Experience*, 3(3), 101-113.

JENNINGS, D. M., L. H. LANDWEBER, y I. H. FUCHS [febrero 28, 1986], Computer Networking for Scientists and Engineers. *Science* vol 231, 941-950

JUBIN, J. y J. TORNOW [enero 1987]]. The DARPA Packet Radio Network Protocols, *IEEE Proceedings.*

KAHN, R. [noviembre 1972], Resource-Sharing Computer Communications Networks, *Proceedings of the IEEE*, 60(11), 1397-1407.

KARN, P., H. PRICE, y R. DIERSING [mayo 1985]. Packet Radio in the Amateur Service, *IEEE Journal on Selected Areas in Communications.*

KARN, P., y C. PARTRIDGE [agosto 1987], Improving Round-Trip Time Estimates in Reliable Transport Protocols. *Proceedings of ACM SIGCOMM'87.*

KENT, C., y J. MOGUL [agosto 1987], Fragmentation Considered Harmful, *Proceedings of ACM SIGCOMM'87.*

KLINE, C. [agosto 1987], Supercomputer on the Internet: A Case Study, *Proceedings of ACM SIGCOMM'87.*

KOCHAN, S. G., y P.H. WOODS [1989], *UNIX Networking*, Hayden Books, Indianapolis, IN.

LABARRE, L. (ED.) [diciembre 1989], OSI Internet Management: Management Information Base, *Internet Draft <IEFT.DRAFTS>DRAFT-IETF-SNMP-MIB2-01.TXT*, DDN Network Information Center, SRI International, Ravenswood, CA.

LAMPSON, B. W., M. PAUL, y H. J. SIEGERT (EDS.) [1981], *Distributed Systems - Architecture and Implementation (An Advanced Course)*, Springer-Verlag, Berlin.

LANZILLO, A. L., y C. PARTRIDGE [enero 1989], Implementation of Dial-up IP for UNIX Systems, *Proceedings 1989 Winter USENIX Technical Conference*, San Diego C.A.

LAQUEY, T. L., [julio 1989], *User's Directory of Computer Networks*, Digital Press, Bedford, MA.

LAZAR, A. [noviembre 1983], Optimal Flow Control of a Class of Queuing Networks in Equilibrium. *IEEE Transactions on Automatic Control*, Vol. AC-28:11.

LEFFLER, S., M. MCKUSICK, M. KARELS, y J. QUARTERMAN [1989], *The Design and Implementation of the 4.3BSD UNIX Operating System*, Addison-Wesley, Reading, Massachusetts.

LYNCH, D. C., (FOUNDER) [1987-], The NETWORLD+INTEROP Conference, *Interop Company*, a division of Softbank Exposition and Conference Company, Foster City, California.

MCNAMARA, J. [1982], *Technical Aspects of Data Communications*, Digital Press, Digital Equipment Corporation, Bedford, Massachusetts.

MCQUILLAN, J. M., I. RICHER, y E. ROSEN [mayo 1980], The New Routing Algorithm for the ARPANET, *IEEE Transactions on Communications*, (COM-28), 711-719.

MERIT [noviembre 1987], Management and Operation of the NSFNET Backbone Network: A Proposal Funded by the National Science Foundation and the State of Michigan, *MERIT Incorporated*, Ann Arbor, Michigan.

METCALFE, R. M., y D. R. BOGGS [julio 1976], Ethernet: Distributed Packet Switching for Local Computer Networks, *Communications of the ACM*, 19(7), 395-404.

MILLER, C. K., y D. M. THOMPSON [marzo 1982], Marking a Case for Token Passing in Local Networks, *Data Communications.*

MILLS, D., y H-W. BRAUN [agosto 1987], The NSFENT Backbone Network, *Proceedings of ACM SIGCOMM'87.*

MITCHELL, J., y J. DION [abril 1982], A Comparison of Two Network-Based File Servers, *Communications of the ACM*, 25(4), 233-245.

MORRIS, R. [1979], Fixing Timeout Intervals for Lost Packet Detection in Computer Communications Networks, *Proceedings AFIPS National Computer Conference*, AFIPS Press, Montvale, New Jersey.

NAGLE, J. [abril 1987], On Packet Switches With Infinite Storage, *IEEE Transactions on Communications*, Vol. COM-35:4.

NARTEN, T. [sept. 1989] Internet Routing, *Proceedings ACM SIGCOMM'89.*

NEEDHAM, R. M. [1979], Systems Aspects of the Cambridge Ring, *Proceedings of the ACM Seventh Symposium on Operating Systems Principles*, 82-85.

NELSON, J. [septiembre 1983], 802; A Progress Report, *Datamation*.

OPEN, D., y Y. DALAL [octubre 1981], The Clearinghouse: A Decentralized Agent for Locating Named Objects, Office Products Division, XEROX Corporation.

PARTRIDGE, C. [junio 1986], Mail Routing Using Domain Names: An Informal Tour, *Proceedings of the 1986 Summer USENIX Conference*, Atlanta, GA.

PARTRIDGE, C. [junio 1987], Implementing the Reliable Data Protocol (RDP), *Proceedings of the 1987 Summer USENIX Conference*, Phoenix, Arizona

PARTRIDGE, C. [1994], *Gigabit Networking*, Addison-Wesley, Reading, Massachusetts.

PETERSON, L. [1985], *Defining and Naming the Fundamental Objects in a Distributed Message System, Ph.D. Dissertation, Purdue University, West Lafayette, Indiana.*

PIERCE, J. R. [1972], Networks for Block Switching of Data, *Bell System Technical Journal*, 51.

POSTEL, J. B. [abril 1980], Internetwork Protocol Approaches, *IEEE Transactions on Communications*, COM-28. 604-611-

POSTEL, J. B., C. A. SUNSHINE, y D. CHEN [1981], The ARPA Internet Protocol, *Computer Networks*.

QUARTERMAN, J. S. [1990], *The Matrix: Computer Networks and Conferencing Systems Worldwide*, Digital Press, Digital Equipment Corporation, Maynard, MA.

QUARTERMAN, J. S. y J. C. HOSKINS [octubre 1986], Notable Computer Networks, *Communications of the ACM*, 29(10).

RAMAKRISHNAN, K. y R. JAIN [mayo 1990], A Binary Feedback Scheme For Congestion Avoidance In Computer Networks, *ACM Transactions on Computer Systems*, 8(2), 158-181.

REYNOLDS, J., J. POSTEL, A. R. KATZ, G. G. FINN, y A. L. DESCHON [octubre 1985], The DARPA Experimental Multimedia Mail System, *IEEE Computer*.

RITCHIE, D. M., y K. THOMPSON [julio 1974], The UNIX Time-Sharing System, *Communications of the ACM*, 17(7), 365-375; revisado y reimpreso en *Bell System Technical Journal*, 57(6), [julio-agosto 1978], 1905-1929.

ROSE, M. (ED.) [octubre 1989], Management Information Base for Networks Managemenet of TCP/IP-based Internets, Internet Draft *<IETF.DRAFTS>DRAFTS-IETF-OIM-MIB2-00.TXT*, DDN Network Information Center, SRI International, Ravenswood, CA.

ROSENTHAL, R. (ED.) [noviembre 1982], *The Selection of Local Area Computer Networks*, National Bureau of Standards Special Publication 500-96.

SALTZER, J. [1978], Naming and Binding of Objects, *Operating Systems, An Advanced Course*, Springer-Verlag, 99-208.

SALTZER, J. [abril 1982], Naming and Binding of Network Destinations, *International Symposium on Local Computer Networks*, IFIP/T.C.6, 311-317.

SALTZER, J., D. REED, y D. CLARK [noviembre 1984], End-to-End Arguments in System Design, *ACM Transactions on Computer Systems*, 2(4), 277-288.

SCHWARTZ, M., y T. STERN [abril 1980], *IEEE Transactions on Communications*, COM-28(4), 539-552.

SHOCH, J. F. [1978], Internetwork Naming, Addressing, and Routing, *Proceedings of COMPCON*.

SHOCH, J. F., Y. DALAL, y D. REDELL [agosto 1982], Evolution of the Ethernet Local Computer Network, *Computer*.

SNA [1975], *IBM System Network Architecture — General Information*, IBM System Development Division, Publications Center, Department E01, P.O. Box 12195, Research Triangle Park, North Carolina, 27709.

SOLOMON, M., L. LANDWEBER, y D. NEUHEGEN [1982], The CSNET Name Server, *Computer Networks* (6), 161-172.

STALLINGS, W. [1984], *Local Networks: An Introduction*, Macmillan Publishing Company, New York.

STALLINGS, W. [1985], *Data and Computer Communications*, Macmillan Publishing Company, New York.

SWINEHART, D., G. MCDANIEL, y D. R. BOGGS [diciembre 1979], WFS: A Simple Shared File System for a Distributed Environment, *Proceedings of the Seventh Symposium on Operating System Principles*, 9-17.

TANENBAUM, A. [1981], *Computer Networks: Toward Distributed Processing Systems*, Prentice-Hall, Englewood Cliffs, New Jersey.

TICHY, W., y Z. RUAN [junio 1984], Towards a Distributed File System, *Proceedings of Summer 84 USENIX Conference*, Salt Lake City, Utah, 87-97

TOMLINSON, R. S. [1975], Selecting Sequence Numbers, *Proceedings ACM SIGOPS/SIGCOMM Interprocess Communication Workshop*, 11-23, 1975.

WARD, A. A. [1980], TRIX: A Network-Oriented Operating System, *Proceedings of COMPCON*, 344-349.

WATSON, R. [1981], Timer-Based Mechanisms in Reliable Transport Protocol Connection Management, *Computer Networks*, North-Holland Publishing Company.

WEINBERGER, P. J. [1985], The UNIX Eighth Eidition Network File System, *Proceedings 1985 ACM Computer Science Conference*, 299-301.

WELCH, B., y J. OSTERHAUT [mayo 1986], Prefix Tables: A Simple Mechanism for Locating Files in a Distributed System, *Proceedings IEEE Sixth International Conference on Distributed Computing Systems*, 1845-189.

WILKES, M. V., y D. J. WHEELER [mayo 1979], The Cambridge Digital Communication Ring, *Proceedings Local Area Computer Network Symposium.*

XEROX [1981], Internet Transport Protocols, *Report XSIS 028112*, Xerox Corporation, Office Products Division, Network Systems Administration Office, 3333 Coyote Hill Road, Palo Alto, California.

ZHANG, L. [agosto 1986], Why TCP Timers Don't Work Well, *Proceedings of ACM SIGCOMM'86.*

Índice

Constantes y aspectos numéricos

10Base-T 25, 566
1822 39
220 447
221 449
250 447
576 99, 566
802.3 20, 566
822 444, 452, 521, 566
9180 314

A

AA 566
AAL1 312
AAL5 313
abierto activo 202, 566
abierto pasivo 202
abortar 207
abstracción de conexiones 201
accept, llamada de sistema 349
accesabilidad 264
acceso a internet 484
acceso en línea 424
acceso transparente 424
ACK 196, 566
activo 273
actualización de horizonte dividido 275
actualización de ruteo 245, 257, 261
actualizaciones desencadenadas 276
acuse de recibo 196, 209, 566
  acumulativo 210
  ambiguo 212
  retrasado 227
acuse de recibo acumulativo 210
acuse de recibo de retraso 227
acuse de recibo positivo 195, 589
acuse de recibo selectivo 590
adaptador 21, 26
adaptador de anfitrión 21
adquisición de vecino 257
Agencia de comunicación de defensa 6
agente 457, 567
agente de manejo 457
agente de retransmisión 376
agrupamiento 227
algoritmo 118
  camino más corto 248
  ruteo 118
algoritmo de adaptación de retransmisión 211
algoritmo de camino más corto de Dijkstra 248
Algoritmo de Karn 212, 573
algoritmo del camino más corto 248
algoritmo Nagle 228, 584
alias
  correo 441
alimentación de líneas 412
ambigüedad de acuses de recibo 212
anfitrión 38, 577
anfitrión baluarte 490, 569
anfitrión multi-horned 63, 584
anfitriones confiables 410
ANS 44, 567
ANSI 567
ANSNET 44, 567
apagado de origen 132, 591
apagado gracioso 219
área 281
área de memoria temporal de correo 440

ARP 77, 567
encapsulación 81
hack 145
implantación 79
protocolo 75
ARP inversa 89
ARP promiscuo 144, 586
ARP sustituto 144, 586
ARPA 2, 37, 567
ARPANET 37, 567
ARQ 567
Arquitecto de Internet 9
arrendamiento 375
arrendamiento de dirección 375
arrendamiento DHCP 375
asignación atómica 467
asignación universal 189
ASN.1 461, 475, 568
ATM 36, 305, 568
NNI 306
UNI 307
ATMARP 318, 568
AUI 21, 568
autenticación 281, 483
autenticación débil 483
autocurativo 33
Autoridad de asignación de números de Internet 524
Autoridad de asignación de números de Internet 69
autorización 482
autosincronizante 228
aviso de ruta 244
avisos de ventanas 204
axioma de vínculo más débil 486

B

balance de cargas 281
banda amplia 569
banda base 569
Base de Información de Manejo 459, 582
base64 449
baudio 569
BBN 38
Bellman 242
Bellman-Ford 242
BGP 266, 569
big endian 71, 569
bilbioteca de sockets 365
BISYNC 569
bit de fin-de-paquete 313
bit de fragmento 101
bloque 487
BNC 570
BOOTP 138, 367, 368, 570
bosquejo de Internet 12
bps 570
brouter (puente brouter) 570
BSC 570
bucle 67
bucles de ruteo 241
buffer 182
bus 27
búsqueda inversa 403
búsqueda por apuntador 402, 403
Butterfly 248

C

cable coaxial 20
cable transceptor 21
cableado 26
cache de nombres 397
caída de paquetes 131
campo de datos 30
campo de longitud de encabezado 95
campo de tipo 30
Capa de adaptación de ATM 310, 566
capa de red de redes 169
capa de transporte 168
capacidad 28
carácter de regreso de carro 412
CCIRN 571
CCITT 39, 45, 166, 571
CDDI 33
célula 36, 310, 571
centro 25, 577
Centro de Información de la red Internet 69

centro de información de red 11
cerrado 219
Chernobylgrama 571
ciclo de ruteo 135
CIDR 156, 571
cierre de conexiones 219
cifrado 483
cifrado público de claves 483, 587
circuito virtual 46, 194, 596
circuito virtual conmutado 308
circuito virtual permanente 308
cliente 327, 328, 368
  ejemplo 359
cliente-servidor 327, 571
  ver volumen III
CLNS 500
codificación TLV 510, 593
cola de correo 440
colapso de congestionamiento 216
colisión 27
columna vertebral de multidifusión 302
comando PORT (FTP) 431
comienzo lento 216, 217, 590
comienzo suave 217
comodín 349
compresión cero 511
comunidad 468
conector BNC 24
conexión 5, 194, 217, 308, 572
  cierre 219
  reiniciación 220
conexión de control 427
conexión de transferencia de datos 427
conexión remota 4, 410
confederación autónoma 267
confianza 482
confianza mutua 482
configuración automática 374
configuración dinámica 374
configuración manual 374
congestionamiento 132, 215
conmutación de circuitos 18
conmutación de paquetes 18
conmutación IP 112
conmutador de paquetes 19
contenido 450
conteo de saltos 135, 245, 273, 577
contexto específico 470
control de acceso 483, 484
control de congestionamiento 205
control de flujo 130, 205, 575
control de fragmentación 100
control de vínculos lógicos 315
convergencia 313
convergencia lenta 274
copiado de archivo completo 425
correo con privacía mejorada 495
correo electrónico 4, 439
  destino 442
  lista 441
  transferencia a la memoria intermedia 436
correspondencia más larga 157
CRC 30, 572
CSMA 27
CSMA/CD 30, 572
CSNET 45
cuenta al infinito 274

D

DARPA 572
datagrama 6, 94, 95, 572
  control de fragmentación 100
  MTU 97
  tamaño 97
  tiempo de vida útil 101
  tipo de servicio 95
  UDP 183
datagrama de Internet 94
datagrama de usuario 181, 183, 367
datagrama IP 99, 574
datos urgentes 207, 416, 421, 595
DCA 37
DCE 572
DDCMP 572
DDN 37, 573
definición de dirección 566
definición de direcciones 76
definición de nombres 395, 584
definición recursiva de nombre 395

demultiplexor 177, 573
dentro 485
derivación manual 490
descripción de archivos 338
destino de correo 441
destino no alcanzable 130
detener temporalmente 275
DHCP 367, 368, 374, 573
difusión 27, 154, 512, 570
difusión Berkeley 569
difusión de subred 154
difusión truncada de caminos reversivos 300
dirección 6, 53, 61, 386
  ARPANET 39
  correo 441, 444
  definición 75, 76
  difusión 29
  Ethernet 28
  física 28
  hardware 20, 28
  internet 63, 75, 85
  IP 62
  multidifusión 29
  red 20
  superred 155
  tipo 62
  tipo D 293
  unidifusión 29
  X.121 46
  X.25 46
dirección ARPANET 40
dirección de buzón de correo 441
dirección de difusión 64, 291
dirección de difusión de red local 64
dirección de hardware 20, 28, 76, 577
dirección de multidifusión 29
dirección de publidifusión limitada 64
dirección de red de redes 76
  notación decimal con puntos 67
dirección de subred 146
dirección de unidifusión 292
dirección dirigida de difusión 64, 573
dirección física 28, 76
dirección Internet 579
dirección IP 62, 93, 580
  notación decimal con puntos 67
dirección muy conocida 293
dirección NSAP 320
dirección tipo A 62
dirección tipo B 62
dirección tipo C 62
dirección tipo D 293
direccionamiento
  correo 441
direccionamiento de camino reversible 155, 583
direccionamiento de correo 441
direccionamiento de subred 591
direccionamiento de superred 155, 591
direccionamiento IP 112
direccionamiento jerárquico 147
disminución multiplicativa 216
disponibilidad 482
disponibilidad de datos 480
Distribución de Software Berkeley BSD 7
distribuidor 441
distribuidor de correo 441, 582
DNS 8, 389, 475, 573
DO (TELNET) 418
DOD 2
DOE 2
dominio 573
dominio padre 396
DON'T (TELNET) 418
DS3 44, 573
DTE 573
DVMRP 297, 574

E

e-mail
  ver correo electrónico
E.164 319, 574
EACK 574
eco
  GGP solicitud/respuesta 247
  ICMP solicitud/respuesta 129

servidor UDP 328
UDP solicitud respuesta 328
EGP 177, 256, 574
accesabilidad de vecino 259
actualización de ruteo 259, 261
adquisición de vecino 258
encabezado de mensaje 259
protocolo 251
punto 259
restricción de provedores 261
solicitud de sondeo 260
vecino 259
EGP2 266
EGP3 266
eliminación del temporizador 213
eliminador de módem 471
encabezado base 502, 568
encabezado de extensión 502
Encabezado de extensión de fragmento 507
encabezado TCP 206
encapsulación 96, 574
datagrama IP 97
ICMP 127
IP 97
RARP 87
enlace de redes 1
entrada estándar 419
entrega con el mejor esfuerzo 27, 93, 293, 569
entrega directa 113
entrega indirecta 113
entrega no confiable de paquetes 92, 93
error estándar 419
escape 410
escribir 227
espacio para nombre plano 386, 575
establecimiento de conexión 217
estado de la máquina
ver tiempo de ruptura
estado de vínculo 247
estándar abierto 500
estándar de bosquejo 524
estándar Internet 524
estándar propuesto 524
estándares de protocolo 8
estandarización 12
estratificación en capas 161, 168
ISO 165
TCP/IP 168
estructura de autoidentificación 30, 38, 314, 584
Estructura de Información de Manejo 460
ether 20
Ethernet 20, 574
adaptador de anfitrión 21
AUI 21
campo de datos 30
campo de tipo 30
centro 25
colisión 27
CRC 30
difusión 27
dirección 28
interfaz de anfitrión 21
marco 29
preámbulo 30
repetidor 30
tipo 97
transceptor 21
Ethernet de cable delgado 23, 24
Ethernet de par trenzado 25, 594
Ethernet grueso 24
etiqueta 389
expansión del alias de correo 441
Extensiones de correo multipropósitos de Internet (MIME) 449

F

familia de protocolos 162
FCCSET 15, 575
FDDI 33, 575
marco 35
símbolo 36
FDM 575
fecha de periodo 330, 574
filtro 487
filtro de paquetes 475, 487
fin-a-fin 168, 170
fingerd 332

FLOW LABEL 504
flujo 194, 504
flujo de datos 203
Ford Fulkerson 242
formato del datagrama 94
fragmentación 97, 506, 576
frontera de dirección 120
FTP 67, 426, 475, 576
FTP anónimo 430, 567
fuera 485
fuera de banda 207, 416
full duplex 195
Fundación Nacional de Ciencias 40
fusión Ethernet 574
Fuzzball 41
FYI 576

G

galleta mágica 372
Gbps 36
GGP 244, 576
GIF 450
gif 450
gopher 13, 473, 576
GOSIP 576
grupo 512
Grupo de control de ingeniería de Internet 11
Grupo de control de Investigación Internet 11
Grupo de Investigación Internet 11
grupo de multidifusión 29
grupo de protocolos 161
Grupo de protocolos Internet TCP/IP 592
grupo de todos los anfitriones 294
grupo de trabajo 10, 596
Grupo de trabajo de ingeniería de Internet 10
Grupo de trabajo de Internet 8
Grupo de trabajo de tarea de investigación Internet 11
grupos transitorios de multidifusión 294
gusano 37, 332
gusano de Internet 37, 332, 579

H

half duplex 195
HDLC 39, 169
HELO 447, 577
HELLO 269, 577
hello (OSPF) 283
hexadecimal con puntos 292
HHS 2
historia 6
histórico 524
HOP LIMIT 503
hora universal 107, 330, 595

I

IAB 8, 577
IANA 69, 524, 578
ICCB 7, 578
ICMP 125, 126, 177, 578
  apagado del origen 132
  código 128
  destino inalcanzable 130
  encapsulación 128
  formato de mensaje 127
  máscara de dirección 139
  máscara de subred 138
  mensaje redireccionado 133
  problema de parámetros 136
  protocolo 125
  redireccionar 133
  solicitud/respuesta de eco 129
  solicitud/respuesta de información 138
  suma de verificación 128
  tiempo excedido 135
  timestamp (marca o sello de hora) 137
  tipo 128
  tipos de mensaje 129
identificador de camino virtual 309
identificador de circuito virtual 309
identificador de objetos 461
Identificador organizacionalmente único 315
IEEE 28

IEN 12, 578
IESG 11, 578
IETF 10, 578
gerente de área 10
grupo de trabajo 10
IGMP 291, 296, 578
IGP 271, 578
IMP 38
implantación
ver Volumen II
InATMARP 321
independencia de tecnologías 5
infinidad 376
infinidad pequeña 274
iniciación 367, 432
INOC 578
Instituto Nacional para Estándares y Tecnología 462
integrado 424
integridad 482
intercambiador (e-mail) 445
intercambiador de correo 392, 445, 582
interconexión de sistema abierto 2
interconexión universal 6, 53
interfaz 21
interfaz de anfitrión 21
Interfaz de Capa de Transporte 365
Interfaz de datos distribuidos por cobre 33
interfaz de red 168, 169
interfaz de unidad de unión 21
Interfaz red a red 306
Interfaz Usuario a Red 307
interior
ruteador o pasarela 269
Internet 578
Internet ARPA/NSF 2
Internet global 2
Internet TCP/IP 2
INTERNIC 11, 69, 386, 521, 579
internos
ver Volumen II
interoperabilidad 4, 579
interpretar como comando 415
interrumpir 207
intervalo de sondeo (EGP) 258
intervalo hello (EGP) 258
IP 580
campo de protocolo 503
código de opción 103
datos 102
dirección 62
dirección de destino 103
dirección de origen 102
encapsulación 96, 97
longitud del encabezado 95
marcación 46
precedencia 95
propósito 93
reensamblaje 98
ruta de origen 105
ruta de registro 104
ruteador 54
suma de verificación 102
tiempo de vida útil 101
timestamp (marca o sello de hora) 106
tipo de servicio 95
versión 95
IP de Línea Serial 174
IP de marcación 46
IP6 500
ipAddrTable 464
ipInReceives 465
IPng 500, 580
ver IPv6
IPv4 501, 580
IPv6 501, 580
destino 503
difusión 512
encabezado fin-a-fin 509
encabezado salto-a-salto 509
fragmentación 506
grupo 512
límite de saltos 503
longitud de carga útil 503
MTU de camino 506
multidifusión 512, 513
prefijo de subred 516
prefijo de suscriptor 516
prefijo proveedor 516
ruta de origen 503

unidifusión 512
versión 503
IRSG 11
IRTF 10, 580
ISDN 580
ISO 165, 580
ISOC 580
ISODE 581
ITU-TS 45, 166, 581

J

jpeg 450
Junta de Actividades de Internet 8
Junta de Arquitectura de Internet 8

K

Kbps 581
kerberos 495
Kramer 581

L

LAN 19, 581
LAPA 166
LAPB 39, 166
límite de ventana de congestionamiento 216
LIS 316, 582
lista de correo 441
lista de sufijos de dominio 402
little endian 71, 582
Llamada de procedimiento remoto 434, 488
llamada de sistema 338
llamada de sistema bind 341
llamada de sistema connect 342
llamada de sistema exec 341
llamada de sistema fork 341
llamada de sistema getdomainname 351
llamada de sistema gethostname 351
llamada de sistema getpeername 347
llamada de sistema getsockname 347
llamada de sistema listen 348
llamada de sistema read 345
llamada de sistema readv 345
llamada de sistema recvfrom 346
llamada de sistema recvmsg 346
llamada de sistema send 344
llamada de sistema sendto 345
llamada de sistema setdomainname 351
llamada de sistema sethostname 351
llamada de sistema socket 340
llamada de sistema write 343
llamada de sistema writev 343
LLC 315, 582

M

MAC 582
MAN 582
manejo de red 458, 585
manejo de red de redes 455
máquina de estado finito 221
marca de datos 416
marcianos 583
marco 29, 35, 166, 575
auto-identificante 30
máscara de dirección 138, 566
máscara de subred 138, 149, 151
MBONE 583
Mbps 583
mecanismo de reporte de errores 126
medición de distancias 244
Memoria de sólo lectura 85
mensaje de control 125
mensaje de tiempo excedido 135
métrica para conteo de saltos 273
MIB 459, 583
MIB-II 459
MILNET 7, 38, 583
MIME 583
Modalidad de transferencia asíncrona 36, 305
modelo de referencia 165
modelo ISO 165
monitoreo 493

monitoreo activo 494
monitoreo pasivo 494
Mosaic 477, 583
MOTIS 167
mrouted 300, 583
MSS 207, 584
MTP 447
MTU 97, 584
datagrama 97
MTU de recorrido 506
MTU de red 97
muestra de viaje redondo 211
muestra del tiempo de viaje redondo 211
multidifusión 65, 291, 292, 512, 513, 584
todos los anfitriones 513
todos los nodos 513
todos los ruteadores 513
multidifusión de todos los anfitriones 513
multidifusión de todos los nodos 513
multidifusión de todos los ruteadores 513
multidifusión Ethernet 292
multidifusión IP 293
multimodalidad 306
multiplexado 177
muro contra incendios (firewall) 479, 575
muro contra incendios de red de redes
firwall 484

N

NAK 584
NAP 515, 584
NASA 2
NBS 462
necesidad de fragmentación 131
NetBIOS 584
netstat 67, 335
NEXT HEADER 506
NFS 434, 585
NIC 585
NIST 585
nivel 1 581
nivel 2 581
nivel 3 581
NNI 306
no autoritario 397
no autorreferenciable 615
no confiable 93
no fragmentar 101, 369
NOC 38, 585
Nodo de conmutación de paquetes 38
nombre 53, 61, 385
abreviatu 401
definición recursiva 396
definidor 393
dominio 385, 389
nombre absoluto 461
nombre de alto nivel 386
nombre de bajo nivel 386
nombre de dominio 385, 389
búsqueda por apuntador 403
definición recursiva 396
servidor 393
zona 405
nombre global 461
notación cuadrangular con puntos 67
notación de sintaxis abstracta 1 167, 461
notación decimal con puntos 67, 573
notación hexadecimal con carácter de dos puntos 511
Notas de ingeniería de Internet 12
NSAP 585
NSF 2, 40, 578
NSFNET 7, 40, 585
nslookup 406
núcleo de multidifusión 300
número de saltos 245, 273
número de sistema autónomo 255
Números asignados 568
NVT 413

O

OC3 585
octeto 30
opción de grabación de ruta 104
opción de ruta de origen 105
opción overload 381
opción timestamp 106

opciones 102
opciones del datagrama 103
opciones IP 103
open-read-write-close 338
orden de octetos 71
orden de octetos de red 584
orden de octetos estándar 71
orden de octetos estándar de red 71
Organización Internacional para la Estandarización 579
orientado a la conexión 18, 37
oscilación de dos pasos 279
OSI 585
OSPF 269, 281, 585
OUI 315

P

PAD 167
paquete 18, 585
paquete de control 455
paquete sin acuse de recibo 198
paradigma de búsqueda-almacenaje 466
partición de espacio de nombre 388
pasando por túneles 46, 488, 594
pasarela 54, 111, 244, 256, 576
  correo 443
  designada 282
  VAN 46
pasarela de correo 443, 582
pasarela de grupo 566
pasarela de red de redes 54
pasarela IP 244, 256
pasarela VAN 46
pasivo 273
PDN 45, 586
PDU 468
PEM 495, 586
perímetro de seguridad 485
PF_INET 340, 342
piggybacking 195
PING 129, 140, 586
poison reverse (contra veneno o antídoto) 276
porteado 280, 576
PPP 174, 586
preámbulo 30
prefijo de subred 516
prefijo de suscriptor 516
prefijo proveedor 516
prevención de SWS 225
prevención de SWS de receptor 226
previsión de congestionamiento 216
Primero el Camino más Corto 247
principio de la estratificación en capas 172
privacía 480, 482, 483
problema de definición de dirección 76
problema de parámetros 136
problema de saltos adicionales 253
problema de traslapamiento de segmentos 230
procedimiento dn_comp 356
procedimiento dn_expand 356
procedimiento endhostent 357
procedimiento endnetent 358
procedimiento endprotoent 358
procedimiento endservent 359
procedimiento gethostbyaddr 357
procedimiento gethostbyname 357
procedimiento gethostent 357
procedimiento getnetbyaddr 357
procedimiento getnetbyname 357
procedimiento getnetent 358
procedimiento getprotobyname 358
procedimiento getprotobynumber 358
procedimiento getprotoent 358
procedimiento getservbyname 359
procedimiento getservbyport 359
procedimiento getservent 359
procedimiento htonl 353
procedimiento htons 353
procedimiento inet_addr 354
procedimiento inet_inaof 355
procedimiento inet_makeaddr 354
procedimiento inet_netof 354
procedimiento inet_network 354
procedimiento inet_ntoa 354
procedimiento ntohl 353
procedimiento ntohs 354
procedimiento res_init 355

procedimiento res_mkquery 355
procedimiento sethostent 357
procedimiento setnetent 358
procedimiento setprotoent 358
procedimiento setservent 359
Procesador de mensajes de interfaz 38
procesamiento de correo 441
proceso 181, 328
proceso de nivel de usuario 181
proceso de usuario 328
programa de aplicación 181
proNET 47
propósito de IP 93
protocolo 3, 586
  aplicación 168
  ARP 75
  BOOTP 367
  datagrama 181
  DHCP 367
  EGP 251, 256
  enlace de datos 168
  estándares 12
  estratificación en capas 161, 168
  flujo 193
  GGP 244
  HELLO 269, 278
  ICMP 125
  IGMP 291
  IGP 271
  Internet 91
  IP 91
  IPng 500
  IPv4 501
  IPv6 501
  manejo de red 458
  MTP 447
  OSPF 269, 281
  puerto 182
  RARP 85, 86
  red de redes 169
  RIP 269, 272
  SMTP 447
  SNMP 458
  ST 501
  TCP 193, 199, 200
  TELNET 408
  UDP 181, 182
Protocolo BOOTP 367
Protocolo BOOTstrap 368
Protocolo de Control de Transmisiones 193, 199
Protocolo de Datagramas de Usuario 182
Protocolo de definición de dirección 77
Protocolo de definición de direcciones reversible 86
Protocolo de enrutamiento de multidifusión de vector de distancia 299
Protocolo de Información de Ruteo 272
Protocolo de Internet para el manejo de grupos 296
Protocolo de mensaje de control de Internet 126
Protocolo de pasarela a pasarela 244
Protocolo de pasarela exterior 256
Protocolo de puerta de frontera 569
Protocolo de puerta interior 271
Protocolo de transferencia de archivos triviales 431
Protocolo DHCP 367
Protocolo dinámico para la configuración de anfitriones 368, 374
Protocolo Internet 91, 93, 580
  versión 499
Protocolo para transferencia de archivos (FTP) 426
Protocolo punto-a-punto 174
Protocolo RIP 269
Protocolo simple de manejo de red 458
Protocolo SPF abierto 281
Protocolo ST 500
Protocolo TCP 193
Proveedor de acceso a red 515
Proveedor de servicio de red 156
próxima generación 500
próxima generación IP 500
pseudo encabezado 184, 209, 515, 587
pseudo terminal 412
PSN 38
puente 111, 570
  correo 443

puente de adaptación 32
puente de aprendizaje 32
puente de correo 443, 582
puerto 131, 328, 586
ARPANET 38
puerto ARPANET 38
puerto de destino 182
puerto de eco 328
puerto de origen 182
puerto de protocolo 200, 223, 328, 586
puerto muy conocido 189, 223, 596
puerto no alcanzable 188
pulsación 21
punto de acceso a servicio de red 319
punto de conexión de subred 315, 591
punto final 201
punto terminal de conexión 202
PUP 587
purga justa 575
push (empujar) 195, 223, 575
PVC 308

R

radio de paquetes 47
RARP 85, 86, 136, 587
rcp 7
RDP 587
realización de conexión 493
realización de difusión 291
recomendado 524
red 18
capacidad 28
dirección 20, 61
red de área amplia 19
red de área local 19
red de columna vertebral 40, 568
red de fragmento 492
red de gran alcance 19
red de nivel medio 40
red de redes 52, 53, 91, 577
características 53
dirección 62, 75, 85
mensaje de control 125
ruteador 54
tabla de ruteo 115
red escondida 253
red inalámbrica 48
red militar 7
redes de columna vertebral del mismo nivel 240
Redes Públicas de Datos 45
Redes y servicios avanzados 45, 567
redirigir 133, 587
redirigir mensaje 133
reensamblado 98, 100, 313, 506, 587
registros de recursos 401
regla de subred 151
regla k-fuera-de-n 248, 259
reiniciación 220
relé de correo 443
repetidor 30, 587
Representación de datos eXternal 434, 575
requerido 524
requerimientos de puerta 576
requerimientos del anfitrión 577
res_send 356
resolución iterativa de nombres 396
respuesta timestamp 137
retirada exponencial 27
retiro 27
retransmisión 195, 210, 211
retransmisión de red regional 39
correo 443
retransmitir 196, 209
retraso 19
RFC 11, 519, 588
RFNM 39
ring 33
RIP 272, 588
RJE 588
rlogin 418, 588
ROADS 156, 588
ROM 85
rotación de contador 33
RPC 435, 475, 588
RS232 589
rsh 419

RTO 589
RTT 589
ruta 55, 61, 133, 588
ruta asignada por omisión 235, 277
ruta de origen 105, 131, 508, 589
ruta de rastreo 140, 593
ruta de subred 146
rutas de publicidad 244
ruteado 272, 300, 588
ruteador 54, 111, 114, 133, 588
  designado 282
  exterior 256
  fragmento 237
  interior 256
  sin ruteo 237
ruteador asignado por omisión 117
ruteador de fragmento 237
ruteador de grupo 237
ruteador de red de redes 54
ruteador designado 282
ruteador exterior 256
ruteador IP 111
ruteador noncore 237
ruteador sin ruteo 237
ruteador transparente 143
ruteador vecino 256
ruteadores de multidifusión 293
ruteo 93, 111
ruteo de origen impreciso 106
ruteo de red de redes 112
ruteo de subred 151
ruteo de tipo de servicio 281, 594
ruteo estricto de origen 106
ruteo IP 112
ruteo jerárquico 147, 577
ruteo por jerarquías 147
Ruteo sin tipo en Inter-dominio 156, 571

S

SACK 589
salida estándar 419
saltos de ruteador 245
SAR (ATM) 313
secuencia de escape 415
segmentación 313
segmento 203, 205, 589
seguridad 117, 479, 480
seguridad de información 480
seguridad de red 480
seguridad de red de redes 480
sentido del transporte 27
señalización 308, 589
servicio
  entrega no confiable de paquetes 93
  servicio de flujo confiable 5
  servicio sin conexión 5
  transporte de flujo confiable 193
Servicio de datos de multimegabits conmutados 314
servicio de eco 328
servicio de fecha 330
servicio de flujo confiable 92
servicio de información sobre la hora 330
Servicio de muy alta velocidad de red de columna vertebral 45, 596
servicio sin conexión 93, 572
servicio transparente 410
servicio universal de comunicaciones 61
servicios de red 4
servicios de red de redes 3
servidor 86, 327, 370
  archivo 328
  ejemplo 361
  hora del día 328
  primario 88
  RARP 87
servidor de archivos 328, 424, 575
servidor de nombres 8
servidor hora-del-día 328
servidor primario 88
servidor RARP 87
SGMP 464, 589
significado cósmico 420
siguiente salto 115, 119, 152
símbolo 35
simétrico 418
Simple IP 500
Simple IP Plus 500
sin conexión 18, 93

síndrome de las ventanas tontas 226, 590
SIP 500, 590
SIPP 500, 590
sistema autónomo 254, 568
Sistema de archivos de red 433
sistema de nomenclatura de dominios 389
Sistema de Nomenclatura de Dominios 8, 385
sistema telefónico 388
Sistema X-Window 596
SLIP 174, 590
SMDS 314, 590
SMTP 447, 590
SNA 590
SNAP 315, 590
SNMP 458, 475, 591
SNMPv2 458
SOA 591
Sociedad Internet 11, 580
sockaddr 342
sockaddr_in 342
socket 7, 340, 591
socket conectado 342
socket no conectado 342
soliciditud de información 136
solicitud de comentarios 16, 519
solicitud de escritura 432
solicitud de lectura 432
solicitud timestamp 137
Sorcerer's Apprentice Bug 433
SPF 256, 591
spoofing 145
SPREAD 248
STD 591
Subred IP lógica 316
subtipo 450
subtipo alternativo (MIME) 450
subtipo de resumen (MIME) 451
subtipo mezclado (MIME) 450
subtipo paralelo (MIME) 451
suma de verificación 102, 128, 571
supernetting (trabajo con superredes) 155
suscriptor 515
SVC 308
SWS 592
SYN 592
SYNCH 416

T

T3 44, 592
tabla de anfitriones 73
tabla de ruteo 115
tamaño
datagrama 97
tamaño de ventana 198, 590
tamaño máximo de segmento 207, 583
tarea 181
TCP 177, 193, 200, 592
puerto de protocolo 223
TCP/IP 2
TDM 592
TDMA 592
tecnología basada en IP 93
TELNET 67, 410, 428, 475, 592
temporizador de reensamblado 100
terminación de tiempo 209, 210
terminación de tiempo y retransmisión 370
terminal virtual de red 411, 413, 428
TFTP 431, 592
thicknet 24, 593
thinnet 23, 24, 593
tiempo de ruptura 334
tiempo de vida 101, 135, 173, 293, 301, 397, 503
tiempo de vida útil máximo de segmento 221, 583
tipo de codificación 449
tipo de contenido 450
tipo de dirección 62, 571
tipo de dominio 392
tipo de nombre 392
Tipo de Servicio 93, 503
tipo de tráfico 504
tipo multiparte (MIME) 450
TLI 366, 593
tn3270 420, 593
token 33
token ring 33, 47, 593
token ring IBM 47

TOS 95, 593
TP-4 593
trabajo con subredes 146
traducción dirección-a-nombre 385
traducción nombre-a-dirección 385
trailers 593
transceptor 21, 594
transferencia confiable 195
transferencia de archivos 4, 421
transferir a memoria intermedia 440
transformación métrica 266
TRPB 300, 594
TTL 101, 301, 594
tubería 340
túnel 300, 301

U

UART 594
UCBCAST 594
UDP 177, 182, 595
  encapsulación 185
  formato de mensaje 183
  protocolo 181
  pseudo encabezado 184
  puerto 182
  servidor de eco 328
UNI 307
unidad de datos de protocolo 469
unidad máxima de transferencia 97, 583
unidifusión 29, 512, 595
Unión Internacional de
  Telecomunicaciones 579
UNIX Berkeley 7
UNIX BSD 7, 570
URL 595
UUCP 446

V

vBNS 45, 595
VCI 309
vecino 244
vecino exterior 256
vecino interior 256
vector-distancia 242, 596
ventana 198, 596
  congestionamiento 215
ventana de congestionamiento 216
ventana deslizable 197, 589
VPI 309
VPI/VCI 309, 595

W

WAN 19, 596
WILL (TELNET) 418
Winsock 337, 365, 596
Wold Wide Web 13, 473, 596
WON'T (TELNET) 418
WWW 596

X

X 596
X.25 39, 45, 597
X.400 167, 597
X25NET 45, 597
XDR 434, 475, 597
XNS 109

Z

zona de autoridad 405, 590

La presente obra fue impresa bajo demanda por vez primera en los talleres de
Publidisa Mexicana SA de CV en el mes de noviembre de 2006.

Publidisa Mexicana SA de CV
Calzada Chabacano N° 69, Planta Alta
Colonia Asturias Deleg. Cuauhtémoc
06850 México DF
www.publidisa.com

Tercera Edición

Redes globales de información con Internet y TCP/IP

Principios básicos, protocolos y arquitectura

Douglas E. Comer

Más de 200,000 ejemplares vendidos

"Es el texto clásico para una introducción a TCP/IP."
— Jon Postel, editor RFC y fundador de la Deputy Internet Architect

"Aun cuando otros lo han intentado, no se ha escrito nada mejor ni se ha hecho una exposición mejor organizada del núcleo de TCP/IP."
— Joel Snyder, *Network Computing*

"Este es un libro excelente como introducción al conjunto de protocolos TCP/IP y sus fundamentos. También es un buen libro de referencia para cualquiera que trabaje con TCP/IP."
— George V. Neville-Neil, *USENIX; login:*

El libro sobre TCP/IP mejor vendido y ampliamente aceptado, *Redes globales de información con Internet y TCP/IP*, es todavía la referencia para cualquiera que desee aprender sobre o trabajar con el conjunto de protocolos de TCP/IP. Esta obra de Douglas Comer proporciona la introducción conceptual más actualizada para los protocolos de TCP/IP y los últimos desarrollos en la tecnología de Internet.

Reconocido por su claridad y accesibilidad, este excelente texto cubre las tecnologías de las redes de área amplia (WAN), en las redes de columna vertebral de Internet, así como las redes de área local (LAN) en tecnologías como Ethernet y FDDI. El texto explica la asignación de direcciones (ARP), la entrega de datagramas sin conexión IP, la detección de errores, la multidifusión y el ruteo.

EN ESTA NUEVA EDICIÓN:

- Discute cómo utilizar TCP/IP en una red ATM.
- Cubre la información y el desarrollo más reciente de IPng (la próxima generación).
- Describe el CIDR (Classless Inter-Domain Routing) y las **super-redes**.
- Discute la seguridad en los ambientes de TCP/IP y el diseño de muros de seguridad.
- Clasifica cientos de RFCs y los protocolos que describen.

Además, esta obra:

- Compara el modelo de referencia de siete capas de ISO con el modelo de referencia de cinco capas de TCP/IP.
- Explica conceptos de TCP como: confiabilidad, acuse de recibo, control de flujo y ventanas deslizantes.
- Detalla las adaptaciones en la retransmisión, incluyendo el arranque lento y el procedimiento para evitar las ventanas tontas.
- Describe la interfaz socket que utilizan las aplicaciones para accesar a los protocolos TCP/IP.
- Presenta arquitecturas de ruteo para redes de redes extensas y pequeñas.
- Discute los temas sobre puentes y ruteadores.
- Examina los servicios de aplicación:
  Sistema de Nombres de Dominio (DNS).
  Correo Electrónico (SMTP, MIME).
  Acceso y transferencia de archivos (FTP, TFTP, NFS).
  Acceso remoto (TELNET, rlogin).
  Administración de red (SNMP, MIB, ANS.1).

ISBN-968-880-541-6
90000
9 789688 805411

PEARSON Educación ®

Visítenos en:
www.pearsoneducacion.net