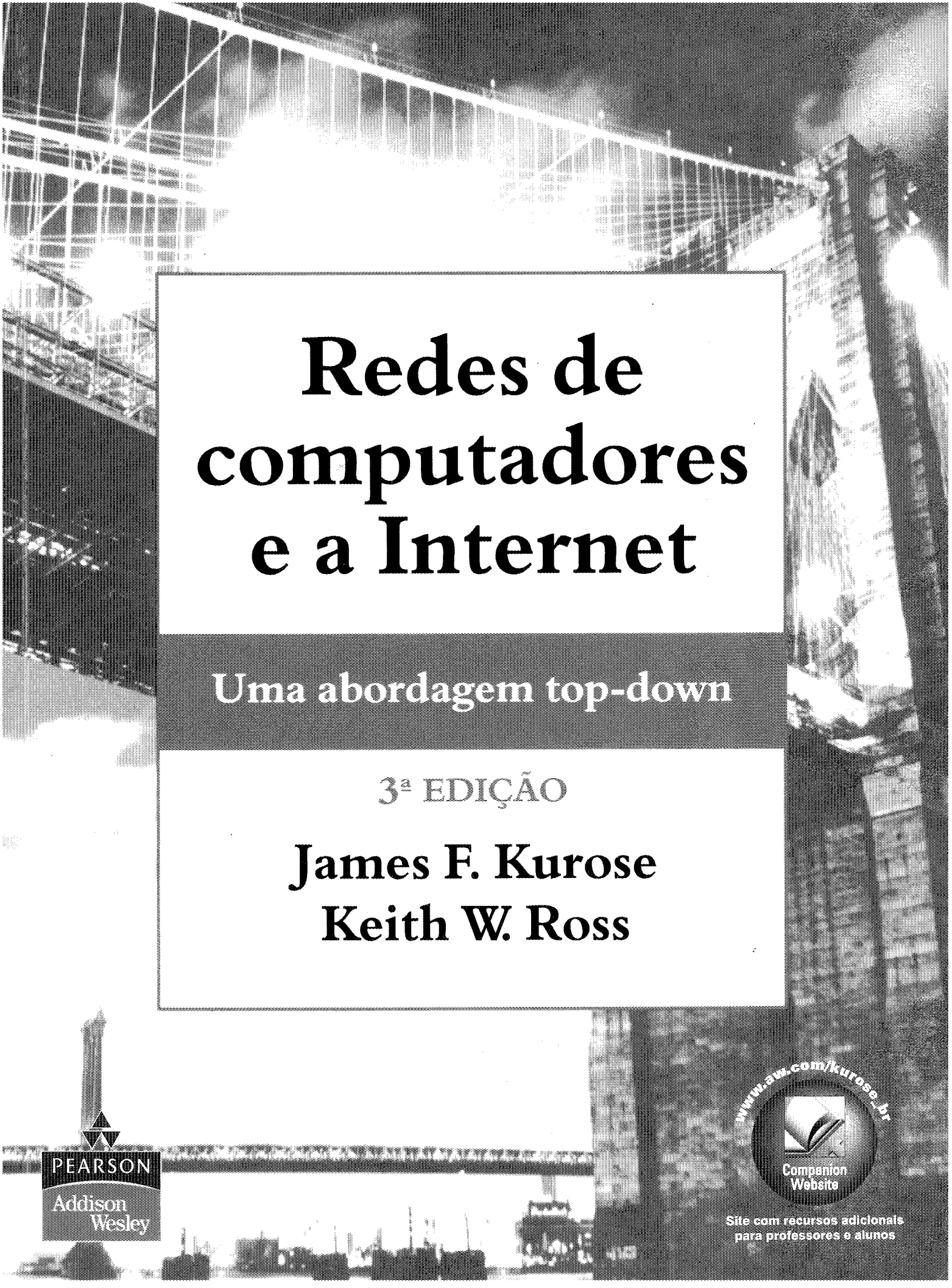


Redes de computadores e a Internet

Uma abordagem top-down

3ª EDIÇÃO

James F. Kurose
Keith W. Ross

PEARSON
Addison Wesley

www.aw.com/kurose_br
Companion Website
Site com recursos adicionais para professores e alunos

PEARSON EDUCATION
Prentice Hall / Addison Wesley / Makron Books
Promoção Universitária
CORTESIA
(Venda Proibida)
universitarios@pearsoned.com
(011) 3613-1212

Redes de computadores e a Internet

Uma abordagem top-down

James F. Kurose
Keith W. Ross

Pearson Education
EMPRESA CIDADÃ

Redes de computadores e a Internet

Uma abordagem top-down

James F. Kurose
Keith W. Ross

Tradução
Arlete Simille Marques
Engenheira Química – UFPR

Revisão Técnica
Wagner Luiz Zucchi
Professor Doutor do Departamento de Sistemas Eletrônicos
da Escola Politécnica da Universidade de São Paulo

PEARSON
Addison Wesley

São Paulo

Brasil Argentina Colômbia Costa Rica Chile Espanha Guatemala México Porto Rico Venezuela

CÓPIA NÃO AUTORIZADA É CRIME
ABDR
ASSOCIAÇÃO BRASILEIRA DE DIREITOS REPROGRÁFICOS
RESPEITE O DIREITO AUTORAL
EDITORA AFILIADA

@ 2006 by Pearson Education do Brasil
Título original: *Computer networking: a top-down approach featuring the Internet, third edition*
@ 2005 by Pearson Education, Inc.
Tradução autorizada a partir da edição original em inglês, publicada pela Pearson Education, Inc., sob o selo Pearson Addison Wesley.
Todos os direitos reservados. Nenhuma parte desta publicação poderá ser reproduzida ou transmitida de qualquer modo ou por qualquer outro meio, eletrônico ou mecânico, incluindo fotocópia, gravação ou qualquer outro tipo de sistema de armazenamento e transmissão de informação, sem prévia autorização, por escrito, da Pearson Education do Brasil.

Gerente editorial: Roger Trimer
Gerente de produção: Heber Lisboa
Editora de desenvolvimento: Patrícia Carla Rodrigues
Editora de texto: Josie Rogero
Preparação: Glória Cunha e João Carlos Campos
Revisão: Hebe Ester Lucas e Maria Aiko Nishijima
Capa: Marcelo Françozo, sobre projeto original de Joyce Cosentino Wells
Imagem de capa: @2004 Photodisc
Projeto gráfico e diagramação: ERJ Composição Editorial e Artes Gráficas Ltda.

Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)

Kurose, James F.
Redes de computadores e a Internet : uma abordagem top-down / James F. Kurose, Keith W. Ross ; tradução Arlete Simille Marques ; revisão técnica Wagner Luiz Zucchi. – 3. ed. – São Paulo : Pearson Addison Wesley, 2006.

Título original: Computer networking third edition a top-down approach featuring the Internet.
Bibliografia.
ISBN 85-88639-18-1

1. Internet 2. Redes de computadores I. Ross, Keith W.. II. Título.

05-6199 CDD-004.67

Índices para catálogo sistemático:
1. Internet : Redes de computadores : Processamento de dados 004.67

2005
Direitos exclusivos para a língua portuguesa cedidos à Pearson Education do Brasil, uma empresa do grupo Pearson Education.
Av. Ermano Marchetti, 1435
CEP: 05038-001 – Lapa – São Paulo – SP
Tel.: (11) 3613-1222 Fax (11) 3611-0444
e-mail: vendas@pearsoned.com

Dedicatória

Para Julie e nossas três preciosidades:
Chris, Charlie e Nina
JFK

Para minha maravilhosa esposa
Véronique (ela é uma ótima cozinheira, entre
outras coisas!) e nossas três filhas: Cécile,
Claire e Katie
KWR

Sobre os autores

Jim Kurose

Jim Kurose é professor e catedrático do Departamento de Ciências da Computação da Universidade de Massachusetts, em Amherst.

Ele foi condecorado inúmeras vezes por suas atividades educacionais, incluindo o Outstanding Teacher Awards da National Technological University (oito vezes), Universidade de Massachusetts e Northeast Association of Graduate Schools. Ele recebeu a IEEE Taylor Booth Education Medal e foi reconhecido por sua liderança no Massachusetts' Commonwealth Information Technology Initiative. Recebeu também o GE Fellowship, o IBM Faculty Development Award e o Lilly Teaching Fellowship.

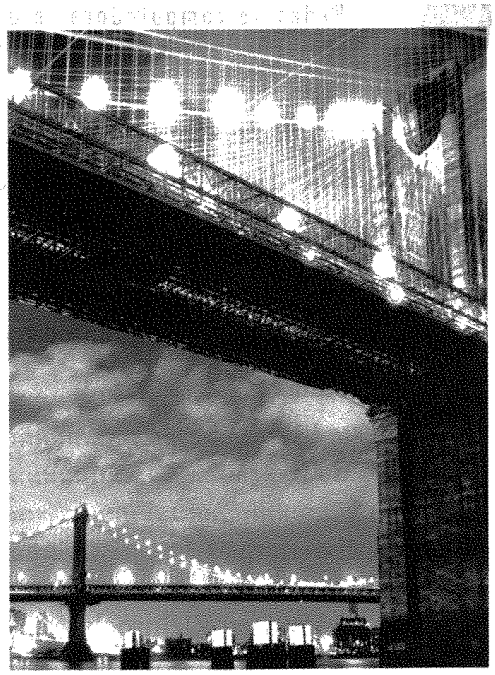
O dr. Kurose foi editor-chefe do IEEE Transactions on Communications e do IEEE/ACM Transactions on Networking. É membro ativo do comitê de programas do IEEE Infocom, da ACM SIGCOMM e da ACM SIGMETRICS há muitos anos e atuou como Technical Program Co-Chair para essas conferências. É membro do IEEE e da ACM. Seus estudos incluem protocolos e arquitetura de rede, medidas de rede, redes de sensores, comunicação multimídia e modelagem e avaliação de desempenho. É doutor em ciência da computação pela Universidade de Colúmbia.

Keith Ross

Keith Ross é o Leonard J. Shustek Professor em ciência da computação na Polytechnic University em Brooklyn. De 1995 a 1998 lecionou no Departamento de Engenharia de Sistemas da Universidade da Pensilvânia. De 1998 a 2003 foi professor do Departamento de Comunicações em Multimídia no Institute Eurecom, na França. Keith Ross é também o principal fundador e presidente da Wimba, que desenvolve tecnologias voice-over-IP para ensino a distância.

Dr. Ross publicou vários artigos e escreveu dois livros. Prestou serviços nos conselhos editoriais de vários jornais, incluindo IEEE/ACM Transactions Networking, e em inúmeros comitês de programas, incluindo ACM SIGCOMM e IEEE Infocom. Supervisionou quinze teses de doutorado. Entre seus interesses na pesquisa e no ensino estão sistemas P2P, redes multimídias, protocolos de rede e modelagem de tráfego. É doutor pela Universidade de Michigan.

Sumário



Prefácio xv

Capítulo 1 – Redes de Computadores e a Internet 1

1.1 O que é a Internet? 2
1.1.1 Uma descrição detalhada da rede 2
1.1.2 Uma descrição do serviço 4
1.1.3 O que é um protocolo? 5

1.2 A periferia da Internet 7
1.2.1 Sistemas finais, clientes e servidores 7
1.2.2 Serviço não orientado para conexão e serviço orientado para conexão 8

1.3 O núcleo da rede 11
1.3.1 Comutação de circuitos e comutação de pacotes 11
1.3.2 Redes de comutação de pacotes: redes de datagramas e redes de circuitos virtuais 17

1.4 Redes de acesso e meios físicos 19
1.4.1 Redes de acesso 20
1.4.2 Meios físicos 24

1.5 ISPs e backbones da Internet 26

1.6 Atraso e perda em redes de comutação de pacotes 28
1.6.1 Tipos de atraso 28
1.6.2 Atraso de fila e perda de pacote 31
1.6.3 Atraso e rotas na Internet 33

1.7 Camadas de protocolo e seus modelos de serviço 34
1.7.1 Arquitetura de camadas 34
1.7.2 Camadas, mensagens, segmentos, datagramas e quadros 38

1.8 História das redes de computadores e da Internet 40
1.8.1 Desenvolvimento da comutação de pacotes: 1961-1972 40
1.8.2 Redes proprietárias e trabalho em rede: 1972-1980 41
1.8.3 Proliferação de redes: 1980-1990 43
1.8.4 A explosão da Internet: a década de 1990 43
1.8.5 Desenvolvimentos recentes 45
1.9 Resumo 45
O guia deste livro 46
Exercícios de fixação 47
Problemas 48
Questões dissertativas 52
Ethereal Lab 53
Entrevista • Leonard Kleinrock 55

Capítulo 2 – Camada de Aplicação 57

2.1 Princípios de aplicações de rede 57
2.1.1 Arquiteturas de aplicação de rede 58
2.1.2 Comunicação entre processos 61
2.1.3 Protocolos de camada de aplicação 63
2.1.4 De que serviços uma aplicação necessita? 64
2.1.5 Serviços providos pelos Protocolos de Transporte da Internet 65
2.1.6 Aplicações de rede abordadas neste livro 67
2.2 A Web e o HTTP 68
2.2.1 Descrição geral do HTTP 68
2.2.2 Conexões persistentes e não persistentes 70
2.2.3 Formato da mensagem HTTP 72
2.2.4 Interação usuário-servidor: cookies 76
2.2.5 Conteúdo HTTP 78
2.2.6 Caches Web 78
2.2.7 GET condicional 81
2.3 Transferência de arquivo: FTP 82
2.3.1 Comandos e respostas FTP 83
2.4 Correio eletrônico na Internet 84
2.4.1 SMTP 86
2.4.2 Comparação com o HTTP 89
2.4.3 Formatos de mensagem de correio e MIME 89
2.4.4 Protocolos de acesso ao correio 92
2.5 DNS: o serviço de diretório da Internet 95
2.5.1 Serviços fornecidos pelo DNS 96
2.5.2 Visão geral do modo de funcionamento do DNS 98
2.5.3 Registros e mensagens DNS 102
2.6 Compartilhamento de arquivos P2P 106
2.7 Programação e desenvolvimento de aplicações com TCP 113
2.7.1 Programação de aplicações com TCP 114

2.7.2 Um exemplo de aplicação cliente–servidor em Java 116
2.8 Programação de sockets com UDP 122
2.9 Construindo um servidor Web simples 128
2.9.1 Funções do servidor Web 128
2.10 Resumo 132
Exercícios de fixação 133
Problemas 134
Questões dissertativas 138
Tarefas de programação de sockets 138
Ethereal Labs 140
Entrevista • Tim Berners-Lee 141

Capítulo 3 – Camada de Transporte 143

3.1 Introdução e serviços de camada de transporte 143
3.1.1 Relação entre as camadas de transporte e de rede 145
3.1.2 Visão geral da camada de transporte na Internet 146
3.2 Multiplexação e demultiplexação 147
3.3 Transporte não orientado para conexão: UDP 153
3.3.1 Estrutura do segmento UDP 155
3.3.2 Soma de verificação UDP 156
3.4 Princípios da transferência confiável de dados 157
3.4.1 Construindo um protocolo de transferência confiável de dados 159
3.4.2 Protocolos de transferência confiável de dados com paralelismo 166
3.4.3 Go-Back-N 169
3.4.4 Repetição seletiva 173
3.5 Transporte orientado para conexão: TCP 178
3.5.1 A conexão TCP 178
3.5.2 Estrutura do segmento TCP 181
3.5.3 Estimativa do tempo de viagem de ida e volta e de esgotamento de temporização 184
3.5.4 Transferência confiável de dados 187
3.5.5 Controle de fluxo 193
3.5.6 Gerenciamento da conexão TCP 195
3.6 Princípios de controle de congestionamento 199
3.6.1 As causas e os custos do congestionamento 199
3.6.2 Mecanismos de controle de congestionamento 204
3.6.3 Exemplo de controle de congestionamento assistido pela rede: controle de congestionamento ATM ABR 205
3.7 Controle de congestionamento TCP 206
3.7.1 Justiça 213
3.7.2 Modelagem do atraso TCP 215
3.8 Resumo 223
Exercícios de fixação 224

Problemas 225
Questões dissertativas 231
Tarefas de programação 231
Ethereal Labs: Explorando o TCP 232
Entrevista • Sally Floyd 233

Capítulo 4 – A Camada de Rede 235

4.1 Introdução 236
4.1.1 Repasse e roteamento 236
4.1.2 Modelos de serviço de rede 239
4.2 Redes de circuitos virtuais e de datagramas 241
4.2.1 Redes de circuitos virtuais 242
4.2.2 Redes de datagramas 244
4.2.3 Origens das redes de circuitos virtuais e de datagramas 246
4.3 O que há dentro de um roteador? 247
4.3.1 Portas de entrada 248
4.3.2 Elemento de comutação 250
4.3.3 Portas de saída 252
4.3.4 Onde ocorre formação de fila? 252
4.4 O Protocolo da Internet (IP): repasse e endereçamento na Internet 255
4.4.1 Formato do datagrama 255
4.4.2 Endereçamento IPv4 260
4.4.3 Protocolo de Mensagens de Controle da Internet (ICMP) 269
4.4.4 IPv6 270
4.5 Algoritmos de roteamento 275
4.5.1 O algoritmo de roteamento de estado de enlace (LS) 278
4.5.2 O algoritmo de roteamento de vetor de distâncias (DV) 281
4.5.3 Roteamento hierárquico 287
4.6 Roteamento na Internet 290
4.6.1 Roteamento intra-AS na Internet: RIP 291
4.6.2 Roteamento intra-AS na Internet: OSPF 293
4.6.3 Roteamento externo a sistemas autônomos: BGP 297
4.7 Roteamento broadcast e multicast 301
4.7.1 Algoritmos de roteamento broadcast 302
4.7.2 Multicast 306
4.8 Resumo 313
Exercícios de fixação 314
Problemas 316
Questões dissertativas 322
Tarefas de programação 323
Ethereal Lab 324
Entrevista • Vinton G. Cerf 325

Capítulo 5 – A Camada de Enlace e Redes Locais 327

5.1 Camada de enlace: introdução e serviços 328

5.1.1 Os serviços fornecidos pela camada de enlace 328
5.1.2 Comunicação por adaptadores 331
5.2 Técnicas de detecção e correção de erros 332
5.2.1 Verificações de paridade 333
5.2.2 Métodos de soma de verificação 335
5.2.3 Verificação de redundância cíclica (CRC) 335
5.3 Protocolos de acesso múltiplo 337
5.3.1 Protocolos de divisão de canal 339
5.3.2 Protocolos de acesso aleatório 341
5.3.3 Protocolos de revezamento 346
5.3.4 Redes locais (LANs) 347
5.4 Endereçamento na camada de enlace 348
5.4.1 Endereços MAC 348
5.4.2 ARP (protocolo de resolução de endereços) 350
5.4.3 Protocolo de Configuração Dinâmica de Hospedeiro 353
5.5 Ethernet 356
5.5.2 CSMA/CD: o protocolo de acesso múltiplo da Ethernet 360
5.5.3 Tecnologias Ethernet 362
5.6 Interconexões: hubs e comutadores 364
5.6.1 Hubs 364
5.6.2 Comutadores de camada de enlace 365
5.7 PPP: o protocolo ponto-a-ponto 372
5.7.1 Enquadramento de dados PPP 374
5.7.2 Protocolo de controle de enlace (LCP) e protocolos de controle de rede do PPP (NCP) 376
5.8 Virtualização de enlace: uma rede como camada de enlace 377
5.8.1 Redes ATM (Modo de Transferência Assíncrono) 378
5.8.2 Comutação de Rótulos Multiprotocolo (MPLS) 382
5.9 Resumo 384
Exercícios de fixação 385
Problemas 386
Questões dissertativas 389
Ethereal Lab 390
Entrevista • Simon S. Lam 391

Capítulo 6 – Redes sem Fio e Redes Móveis 393

6.1 Introdução 394
6.2 Características de enlaces e redes sem fio 396
6.2.1 CDMA 398
6.3 Wi-Fi: LANs sem fio 802.11 401
6.3.1 A arquitetura 802.11 402
6.3.2 O protocolo MAC 802.11 404

6.3.3 O quadro IEEE 802.11 408
6.3.4 Mobilidade na mesma sub-rede IP 411
6.3.5 802.15 e Bluetooth 412
6.4 Acesso celular à Internet 413
6.4.1 Uma visão geral da arquitetura celular 413
6.4.2 Padrões e tecnologias celulares: um breve levantamento 416
6.5 Gerenciamento da mobilidade: princípios 418
6.5.1 Endereçamento 421
6.5.2 Roteamento para um nó móvel 422
6.6 IP móvel 426
6.7 Gerenciamento de mobilidade em redes celulares 429
6.7.1 Roteando chamadas para um usuário móvel 430
6.7.2 Transferências em GSM (handoffs) 431
6.8 Sem fio e mobilidade: impacto sobre protocolos de camadas superiores 434
6.9 Resumo 435
Exercícios de fixação 436
Problemas 436
Questões dissertativas 438
Ethereal Lab 438
Entrevista • Charlie Perkins 439

Capítulo 7 – Redes Multimídia 442

7.1 Aplicações de rede multimídia 442
7.1.1 Exemplos de aplicações multimídia 443
7.1.2 Obstáculos para a multimídia na Internet de hoje 445
7.1.3 Como a Internet deveria evoluir para dar melhor suporte à multimídia? 446
7.1.4 Compressão de áudio e vídeo 448
7.2 Áudio e vídeo de fluxo contínuo armazenados 449
7.2.1 Acesso a áudio e vídeo por meio de um servidor Web 450
7.2.2 Envio de multimídia de um servidor de fluxo contínuo a uma aplicação auxiliar 452
7.2.3 RTSP (protocolo de fluxo contínuo em tempo real) 454
7.3 Fazendo o melhor possível com o serviço de melhor esforço: um telefone por Internet como exemplo 457
7.3.1 As limitações de um serviço de melhor esforço 458
7.3.2 Eliminação da variação de atraso no receptor para áudio 459
7.3.3 Recuperação de perda de pacotes 462
7.3.4 Áudio e vídeo de fluxo contínuo armazenados 465
7.4 Protocolos para aplicações interativas em tempo real 465
7.4.1 Protocolo de tempo real (RTP) 465
7.4.2 Protocolo de Controle RTP (RTCP) 469
7.4.3 SIP 471
7.4.4 H.323 476
7.5 Distribuição de multimídia: redes de distribuição de conteúdo 477

7.6 Além do melhor esforço 480
7.6.1 Cenário 1: uma aplicação de áudio de 1 Mbps e uma transferência FTP 481
7.6.2 Cenário 2: uma aplicação de áudio de 1 Mbps e uma transferência FTP de alta prioridade 482
7.6.3 Cenário 3: uma aplicação de áudio malcomportada e uma transferência FTP 483
7.6.4 Cenário 4: duas aplicações de áudio de 1 Mbps por um enlace de 1,5 Mbps sobrecarregado 484
7.7 Mecanismos de escalonamento e regulação 485
7.7.1 Mecanismos de escalonamento 485
7.7.2 Regulação: o leaky bucket 489
7.8 Serviços integrados e serviços diferenciados 491
7.8.1 Intserv 491
7.8.2 Diffserv 493
7.9 RSVP 497
7.9.1 A essência do RSVP 498
7.9.2 Alguns exemplos simples 499
7.10 Resumo 503
Exercícios de fixação 504
Problemas 504
Questões dissertativas 507
Tarefa de programação 508
Entrevista • Henning Schulzrinne 509

Capítulo 8 – Segurança em Redes de Computadores 512

8.1 O que é segurança na rede? 513
8.2 Princípios de criptografia 515
8.2.1 Criptografia de chaves simétricas 517
8.2.2 Criptografia de chave pública 521
8.3 Autenticação 525
8.3.1 Protocolo de autenticação ap1.0 525
8.3.2 Protocolo de autenticação ap2.0 526
8.3.3 Protocolo de autenticação ap3.0 527
8.3.4 Protocolo de autenticação ap3.1 527
8.3.5 Protocolo de autenticação ap4.0 528
8.3.6 Protocolo de autenticação ap5.0 529
8.4 Integridade 530
8.4.1 Geração de assinaturas digitais 531
8.4.2 Resumos de mensagem 532
8.4.3 Algoritmos de função de hash 534
8.5 Distribuição de chaves e certificação 536
8.5.1 A central de distribuição de chaves 536
8.5.2 Certificação de chaves públicas 538
8.6 Controle de acesso — Firewalls 541

8.6.1 Filtragem de pacotes 542
8.6.2 Gateway de aplicação 544

8.7 Ataques e contramedidas 546
8.7.1 Mapeamento 546
8.7.2 Análise de pacotes 547
8.7.3 Falsificação 548
8.7.4 Ataques de negação de serviço e ataques de negação de serviço distribuídos 548
8.7.5 Seqüestro 549

8.8 Segurança em muitas camadas: estudos de casos 550
8.8.1 E-mail seguro 550
8.8.2 Camada de sockets de segurança (SSL) e segurança na camada de transporte (TLS) 555
8.8.3 Segurança na camada de rede: IPsec 558
8.8.4 Segurança em IEEE 802.11 560

8.9 Resumo 565
Exercícios de fixação 565
Problemas 566
Questões dissertativas 568
Entrevista • Steven M. Bellovin 569

Capítulo 9 – Gerenciamento de Rede 571

9.1 O que é gerenciamento de rede? 571

9.2 A infra-estrutura do gerenciamento de rede 575

9.3 A estrutura de gerenciamento padrão da Internet 578
9.3.1 SMI (Estrutura de Informações de Gerenciamento) 580
9.3.2 Base de informações de gerenciamento: MIB 582
9.3.3 Operações do protocolo SNMP e mapeamentos de transporte 584
9.3.4 Segurança e administração 586

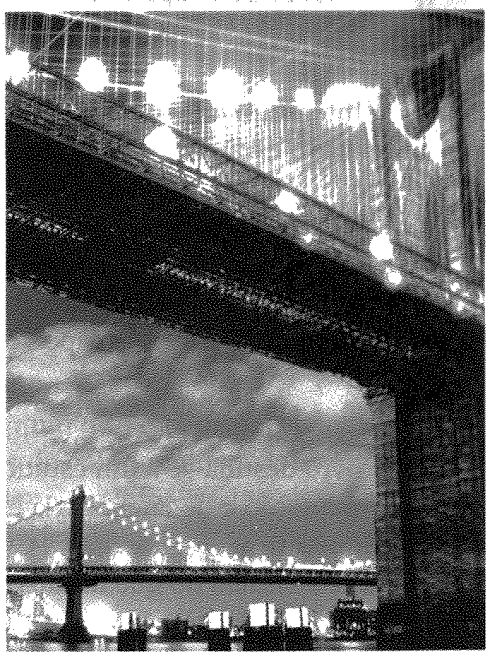
9.4 ASN.1 589

9.5 Conclusão 593
Exercícios de fixação 594
Problemas 594
Questões dissertativas 595
Entrevista • Jeff Case 596

Referências 598

Índice 625

Prefácio



Bem-vindo à terceira edição de *Redes de computadores e a Internet: uma abordagem top-down*. Desde a publicação da primeira edição, nosso livro foi adotado em centenas de universidades e instituições de ensino superior, traduzido para mais de dez idiomas e utilizado por mais de cem mil estudantes e profissionais no mundo inteiro. Muitos desses leitores entraram em contato conosco e ficamos extremamente satisfeitos com sua reação positiva.

Acreditamos que uma importante razão para esse sucesso é que o livro oferece uma abordagem moderna do ensino de redes de computadores. Você poderia perguntar: por que é necessária uma abordagem moderna? Nos últimos anos, testemunhamos duas mudanças revolucionárias no campo das redes — mudanças que não estão refletidas nos livros sobre o assunto publicados entre as décadas de 1980 e 1990. Em primeiro lugar, a Internet dominou o universo das redes de computadores. Atualmente, em qualquer discussão séria sobre redes, é obrigatório ter a Internet em mente. Em segundo lugar, o maior crescimento nessa área ocorreu no âmbito dos serviços e das aplicações, o que pode ser confirmado pelo desenvolvimento da Web, pela utilização dos serviços de e-mail por milhares de usuários ao mesmo tempo, pela recepção de áudio e vídeo, pelo telefone por Internet, pelos serviços de mensagem instantânea, pelas aplicações P2P e pelo comércio eletrônico.

Quais são as novidades desta terceira edição?

Fizemos mudanças nesta terceira edição, mas mantivemos inalterados os aspectos que acreditamos ser (e os professores e estudantes que utilizaram nosso livro confirmaram) os mais importantes deste livro: a abordagem top-down, o foco na Internet, a atenção que demos aos princípios e à prática e sua abordagem e estilo acessíveis do ensino de redes de computadores.

Contudo, fizemos muitas mudanças significativas na terceira edição, incluindo um **novo capítulo sobre redes sem fio e redes móveis**. Hoje estamos testemunhando uma grande mudança no modo como usuários acessam a Internet e seus serviços. Agora eles podem acessar a Internet por meio de dispositivos sem fio no trabalho, em casa e em locais públicos e também quando em trânsito, por meio de um conjunto de dispositivos que inclui laptops, telefones, PDAs e outros. Nosso novo capítulo sobre tecnologia sem fio e mobilidade abrange uma discussão em profundidade do padrão 802.11, uma visão geral do acesso por

celular à Internet e uma discussão abrangente da mobilidade na Internet e em redes celulares. Com a adição desse novo capítulo, o livro contém agora quatro capítulos avançados e especializados: redes sem fio e móveis; redes multimídia; segurança de rede e gerenciamento de redes.

E isso não é tudo. A terceira edição foi atualizada para refletir as rápidas mudanças na área de redes nos últimos anos. Inclui material novo e mais abrangente sobre redes P2P, BGP, MPLS, segurança de rede, roteamento broadcast e endereçamento e repasse na Internet. Também reestruturamos o Capítulo 4, oferecendo uma exposição mais clara dos papéis do repasse e do roteamento e de como eles interagem na camada de rede.

Público-alvo

Este livro destina-se a um estudo inicial de redes de computadores. Pode ser usado em cursos de ciência da computação e de engenharia elétrica. Em termos de linguagem de programação, requer que os estudantes conheçam as linguagens C, C^{++} e Java (mas apenas em alguns lugares). Embora este livro seja mais minucioso e analítico do que muitos outros de introdução às redes de computadores, raramente utiliza conceitos matemáticos que não sejam ensinados no ensino médio. Fizemos um esforço deliberado para evitar o uso de quaisquer conceitos avançados de cálculo, probabilidade ou processos estocásticos. Em conseqüência, o livro é apropriado para cursos de graduação e para o primeiro ano dos cursos de pós-graduação. É também muito útil para os profissionais do setor de telecomunicações.

O que há de singular neste livro?

O assunto rede de computadores é bastante vasto e complexo e envolve muitos conceitos, protocolos e tecnologias que se entrelaçam inextricavelmente. Para dar conta desse escopo e complexidade, muitos livros sobre redes são, em geral, organizados de acordo com as 'camadas' de uma arquitetura de rede. Com a organização em camadas, os estudantes podem vislumbrar a complexidade das redes de computadores — eles aprendem os conceitos e os protocolos distintos de uma parte da arquitetura e, ao mesmo tempo, visualizam o grande quadro da interconexão entre as camadas. Do ponto de vista pedagógico, nossa experiência pessoal confirma que essa abordagem em camadas é, de fato, muito boa. Entretanto, achamos que a abordagem tradicional, a bottom-up — da camada física para a camada de aplicação —, não é a melhor abordagem para um curso moderno de redes de computadores.

Uma abordagem top-down

Na primeira edição, propusemos uma inovação adotando uma visão top-down — isto é, começando na camada de aplicação e descendo até a camada física. A abordagem top-down oferece diversos benefícios importantes. Em primeiro lugar, dá ênfase à camada de aplicação, que tem sido a área de "grande crescimento" das redes de computadores. De fato, muitas das recentes revoluções nesse ramo — incluindo a Web, o compartilhamento de arquivos P2P e o fluxo contínuo de mídia — tiveram lugar nessa camada. Essa abordagem de ênfase inicial à camada de aplicação é diferente das seguidas por muitos outros livros, que têm apenas pouco (ou nenhum) material sobre aplicações de redes, seus requisitos, paradigmas da camada de aplicação (por exemplo, cliente-servidor) e interfaces de programação de aplicação.

Em segundo lugar, nossa experiência como professores (e a de muitos outros que utilizaram este livro) confirma que ensinar aplicações de rede logo no início do curso é uma poderosa ferramenta motivadora. Os estudantes ficam mais entusiasmados ao aprender como funcionam as aplicações de rede — aplicações como o e-mail e a Web, que a maioria deles usa diariamente. Entendendo as aplicações, o estudante pode entender os serviços de rede necessários ao suporte de tais aplicações. Pode também, por sua vez, examinar as várias maneiras como esses serviços são fornecidos e implementados nas camadas mais baixas. Assim, a discussão das aplicações logo no início fornece a motivação necessária para os demais assuntos do livro.

Em terceiro lugar, essa abordagem top-down habilita o professor a apresentar o desenvolvimento de aplicações de rede no estágio inicial. Os estudantes não apenas vêem como funcionam aplicações e protocolos populares, como também aprendem que é fácil criar suas próprias aplicações e protocolos de aplica-

ção de rede. Com a abordagem top-down, eles entram imediatamente em contato com as noções de interfaces de programação de aplicações (*application programming interfaces* — API) e com os modelos de serviços e protocolos — conceitos importantes que reaparecem em todas as camadas subseqüentes. Ao apresentar exemplos de programação de sockets em Java, destacamos as idéias centrais sem confundir os estudantes com códigos complexos. Estudantes de engenharia elétrica e ciência da computação provavelmente não terão dificuldades para entender o código Java.

Um foco na Internet

Este livro, como indica seu título, apresenta a Internet e utiliza a arquitetura e os protocolos da Internet como veículo primordial para estudar conceitos fundamentais de redes de computadores. É claro que também incluímos conceitos e protocolos de outras arquiteturas de rede. Mas os holofotes estão claramente dirigidos à Internet, fato refletido na organização do livro, que gira em torno da arquitetura de cinco camadas da Internet: aplicação, transporte, rede, enlace e física.

Outro benefício de colocar a Internet sob os holofotes é que a maioria dos estudantes de ciência da computação e de engenharia elétrica está ávida por conhecer a Internet e seus protocolos. Eles sabem que a Internet é uma tecnologia revolucionária e inovadora e podem constatar que ela está provocando uma profunda transformação em nosso mundo. Dada a enorme relevância da Internet, os estudantes estão naturalmente curiosos em saber o que há por trás dela. Assim, fica fácil para um professor manter seus alunos interessados nos princípios básicos usando a Internet como guia.

Abordagem dos princípios

Duas das características exclusivas deste livro — sua abordagem top-down e seu foco na Internet — aparecem no título e subtítulo. Se pudéssemos, teríamos acrescentado uma *terceira* palavra ao subtítulo — *princípios*. O campo das redes agora está suficientemente maduro para que uma quantidade de assuntos de importância fundamental possa ser identificada. Por exemplo, na camada de transporte, dentre os assuntos importantes estão a comunicação confiável por uma camada de rede não confiável, o estabelecimento/encerramento de conexões e mútua apresentação, o controle de congestionamento e de fluxo e a multiplexação. Na camada de rede, dois assuntos fundamentalmente importantes são: como determinar 'bons' caminhos entre dois roteadores e como interconectar um grande número de redes heterogêneas. Na camada de enlace, um problema fundamental é como compartilhar um canal de acesso múltiplo. Na segurança de rede, técnicas para prover sigilo, autenticação e integridade de mensagens são baseadas em fundamentos da criptografia. Este livro identifica as questões fundamentais de redes e apresenta abordagens para enfrentar essas questões. Aprendendo esses princípios, o estudante adquire conhecimento de longa validade — muito tempo após os padrões e protocolos de rede de hoje tornarem-se obsoletos, os princípios que ele incorpora continuarão importantes e relevantes. Acreditamos que o uso da Internet para apresentar o assunto aos estudantes e a ênfase dada à abordagem das questões e das soluções permitirão que os alunos entendam rapidamente qualquer tecnologia de rede.

Materiais de apoio

O livro inclui um site Web de apoio, `www.aw.com/kurose_br`, com recursos didáticos adicionais para professores e estudantes.

Para os estudantes

- *Exercícios on-line autocorrigíveis.* Teste seus conhecimentos com exercícios de múltipla escolha e, se quiser, envie o resultado para seu professor.
- *Tarefas de programação.* As tarefas de programação incluem a construção de um servidor Web multithread, de um cliente de e-mail com interface GUI, programação dos lados cliente e servidor de

um protocolo confiável de transporte de dados, programação de um algoritmo de roteamento distribuído e outros mais.

- *Ethereal Labs*. O site fornece detalhes práticos, em inglês, dos Ethereal Labs que constam no livro.

Para os professores

Fornecemos um pacote de suplementos para auxiliar o ensino do conteúdo deste livro. Esse material, restrito a professores, está disponível no Companion Website do livro (www.aw.com/kurose_br).

- *Slides em PowerPoint*. O site do livro apresenta slides em PowerPoint para todos os nove capítulos. Os slides dão detalhes completos sobre cada capítulo e neles usamos gráficos e animações — e não apenas marcadores — para tornar sua aula visualmente interessante e atraente. Fornecemos os slides originais aos professores para que eles possam personalizá-los do modo que melhor atenda a suas necessidades de ensino. Alguns desses slides foram fornecidos por professores que utilizaram nosso livro em seus cursos.
- *Solução dos exercícios*. O site do livro oferece um manual de soluções, em inglês, para os exercícios apresentados ao final de cada capítulo.

O acesso dos estudantes é livre. Para ter acesso ao conteúdo exclusivo para professores, entre em contato com seu representante Pearson ou envie um e-mail para universitarios@pearsoned.com.

Características pedagógicas

Damos aulas de redes de computadores há quase 20 anos. Adicionamos a este livro uma experiência agregada de mais de 35 anos de ensino para mais de três mil estudantes. Durante esse período, também participamos ativamente na área de pesquisas sobre redes de computadores. (De fato, Jim e Keith se conheceram quando faziam mestrado, freqüentando um curso sobre redes de computadores ministrado por Mischa Schwartz em 1979, na Universidade de Colúmbia.) Achamos que isso nos dá uma boa perspectiva do que foi a rede e de qual será, provavelmente, seu futuro. Não obstante, resistimos às tentações de dar ao material deste livro um viés que favorecesse nossos projetos de pesquisa prediletos. Se você estiver interessado em nossas pesquisas, consulte nosso site Web pessoal. Este livro é sobre redes de computadores modernas — é sobre protocolos e tecnologias contemporâneas, bem como sobre os princípios subjacentes a esses protocolos e tecnologias. Também achamos que aprender (e ensinar!) redes pode ser divertido. Esperamos que um certo senso de humor e a utilização de analogias e exemplos do mundo real que aparecem neste livro tornem este material ainda mais divertido.

Adendos e princípios na prática

O campo das redes de computadores tem uma história rica e fascinante. Fizemos, neste livro, um esforço especial para contar a história das redes de computadores, o qual se materializou em uma seção histórica especial no Capítulo 1 e em 12 adendos históricos espalhados pelos demais capítulos. Nesses artigos históricos, apresentamos a invenção da comutação de pacotes, a evolução da Internet, o nascimento de importantes empresas gigantes de redes, como a Cisco e a 3Com, e muitos outros eventos relevantes. Os estudantes certamente se sentirão estimulados por esses acontecimentos históricos. Em cada capítulo, incluímos um adendo especial que acentua um princípio importante de rede de computadores. Esses adendos auxiliarão o estudante a compreender alguns dos conceitos fundamentais que estão sendo aplicados às redes modernas.

Entrevistas

Inserimos uma outra característica original que deverá inspirar e motivar os estudantes — entrevistas com inovadores famosos no campo de redes. Apresentamos entrevistas com Leonard Kleinrock, Tim Berners-Lee, Sally Floyd, Vint Cerf, Simon Lam, Charlie Perkins, Henning Schulzrinne, Steven Bellovin e Jeff Case.

Dependência entre os capítulos

O primeiro capítulo deste livro apresenta um apanhado geral sobre redes de computadores. Com a introdução de muitos conceitos e terminologias fundamentais, ele monta o cenário para o restante do livro. Todos os outros capítulos dependem diretamente desse primeiro. Recomendamos que os professores, após o terem completado, percorram em seqüência os capítulos 2 ao 5, ensinando, assim, de acordo com a filosofia top-down. Cada um dos cinco primeiros capítulos utiliza material dos capítulos precedentes.

Após ter completado os cinco primeiros capítulos, o professor terá bastante flexibilidade. Não há interdependência entre os quatro últimos capítulos, de modo que eles podem ser ensinados em qualquer ordem. Contudo, cada um dos quatro depende de material dos cinco primeiros. Muitos professores ensinam os primeiros cinco capítulos e então um dos quatro últimos para arrematar.

Uma nota final: gostaríamos que vocês entrassem em contato conosco

Incentivamos professores e estudantes a criar novas aplicações Java que ilustrem os conceitos e protocolos apresentados neste livro. Se você tiver uma aplicação que julgue apropriada para este livro, apresente-a, por gentileza, aos autores (em inglês). Se a aplicação (incluindo notação e terminologia) for adequada, teremos o prazer de incluí-la no site Web do livro em inglês (www.aw.com/kurose-ross), com a devida referência a seus autores. Do mesmo modo, também solicitamos aos professores que nos enviem novos problemas (e soluções) que possam complementar as atuais seções de exercícios de fixação. Esses problemas podem vir a ser incorporados à parte desse site Web dedicada apenas aos professores.

Gostaríamos também que estudantes e professores nos enviassem por e-mail (em inglês) quaisquer comentários que porventura queiram fazer sobre nosso livro. Foi maravilhoso saber a opinião de tantos professores de todo o mundo sobre as duas primeiras edições. Sintam-se à vontade também para nos enviar URLs interessantes ou discordar de qualquer de nossas afirmações e nos informar o que funciona e o que não funciona. Digam-nos ainda o que poderia (ou o que não deveria) ser incluído na próxima edição. Enviem seus e-mails para `kurose@cs.umass.edu` e `ross@poly.edu`.

Até a data de publicação deste livro, todos os URLs mencionados no texto estavam funcionando. Devido ao caráter dinâmico da Internet, porém, alguns deles podem mudar ou deixar de existir. Para tentar encontrar os sites e informações mencionados, use uma ferramenta de busca como `www.google.com` ou `www.yahoo.com`.

Agradecimentos

Desde o início deste projeto, em 1996, muitas pessoas nos deram inestimável auxílio e influenciaram nossas idéias sobre como melhor organizar e ministrar um curso sobre redes. Nosso MUITO OBRIGADO a todos os que nos ajudaram, sobretudo às centenas de estudantes, acadêmicos e profissionais que nos enviaram sugestões e comentários sobre as edições anteriores e sugestões para futuras edições deste livro. Nossos agradecimentos especiais para:

Al Aho (Universidade de Colúmbia)
Pratima Akkunoor (Arizona State University)
Paul Amer (Universidade de Delaware)
Shamiul Azom (Arizona State University)
Paul Barford (Universidade de Wisconsin)
Bobby Bhattacharjee (Universidade de Maryland)
Steven Bellovin (AT&T Research)
Pravin Bhagwat (Wibhu)
Supratik Bhattacharyya (Sprint)
Shahid Bokhari (University of Engineering & Technology, Lahore)
Ernst Biersack (Eurécom Institute)
Daniel Brushteyn (ex-aluno da Universidade da Pensilvânia)
Ken Calvert (Universidade de Kentucky)
Evandro Cantu (Universidade Federal de Santa Catarina)
Jeff Case (SNMP Research International)
Vinton Cerf (MCI WordCom)
Byung Kyu Choi (Michigan Technological University)
John Daigle (Universidade do Mississippi)
Edmundo A. de Souza e Silva (Universidade Federal do Rio de Janeiro)
Philippe Decuetos (Eurécom Institute)
Christophe Diot (Sprint)
Michalis Faloutsos (University of California at Riverside)

Wu-chi Feng (Oregon Graduate Institute)
Charles M. Fleckenstein (Sprint)
David Flessas (Sprint)
Sally Floyd (ICIR, University of California at Berkeley)
Paul Francis (Cornell)
Lixin Gao (Universidade de Massachusetts)
J.J. Garcia-Luna-Aceves (University of California at Santa Cruz)
Mario Gerla (University of California at Los Angeles)
David Goodman (Polytechnic University)
Tim Griffin (AT&T Research)
Max Hailperin (Gustavus Adolphus College)
Bruce Harvey (Florida A&M University, Florida State University)
Carl Hauser (Washington State University)
Phillipp Hoschka (INRIA/W3C)
Albert Huang (ex-aluno da Universidade da Pensilvânia)
Esther A. Hughes (Virginia Commonwealth University)
Jobin James (University of California at Riverside)
Sugih Jamin (Universidade de Michigan)
Shivkumar Kalyanaraman (Rensselaer Polytechnic Institute)
Jussi Kangasharju (Universidade de Darmstadt)
Sneha Kasera (Universidade de Utah)
Hyojin Kim (ex-aluno da Universidade da Pensilvânia)
Leonard Kleinrock (University of California at Los Angeles)
David Kotz (Dartmouth College)
Beshan Kulapala (Arizona State University)
Miguel A. Labrador (University of South Florida)
Steve Lai (Ohio State University)
Tim-Berners Lee (World Wide Web Consortium)
Brian Levine (Universidade de Massachusetts)
William Liang (ex-aluno da Universidade da Pensilvânia)
Willis Marti (Texas A&M University)
Deep Medhi (University of Missouri, Kansas City)
Bob Metcalfe (International Data Group)
Sue Moon (KAIST)
Erich Nahum (IBM Research)
Christos Papadopoulos (University of Southern Californ
Craig Partridge (BBN Technologies)
Radia Perlman (Sun Microsystems)
Jitendra Padhye (Microsoft Research)
Kevin Phillips (Sprint)
George Polyzos (University of California at San Diego)
Sriram Rajagopalan (Arizona State University)
Ken Reek (Rochester Institute of Technology)
Martin Reisslein (Arizona State University)
Jennifer Rexford (AT&T Research)
Sumit Roy (Universidade de Washington)
Avi Rubin (Johns Hopkins University)
Dan Rubenstein (Universidade de Colúmbia)
Despina Saparilla (Lucent Bell Labs)
Henning Schulzrinne (Universidade de Colúmbia)
Mischa Schwartz (Universidade de Colúmbia)
Harish Sethu (Drexel University)
K. Sam Shanmugan (Universidade de Kansas)
Prashant Shenoy (Universidade de Massachusetts)
Clay Shields (Universidade de Georgetown)
Subin Shrestra (Universidade da Pensilvânia)
Peter Steenkiste (Carnegie Mellon University)
Tatsuya Suda (University of California at Irvine)
Kin Sun Tam (State University of New York at Albany)
Don Towsley (Universidade de Massachusetts)
David Turner (California State University, San Bernardin
David Wetherall (Universidade de Washington)
Ira Winston (Universidade da Pensilvânia)
Raj Yavatkar (Intel)
Yechiam Yemini (Universidade de Colúmbia)
Ellen Zegura (Georgia Institute of Technology)
Hui Zhang (Carnegie Mellon University)
Lixia Zhang (University of Califórnia at Los Angeles)
ZhiLi Zhang (Univesidade de Minnesota)
Shuchun Zhang (ex-aluno da Universidade da Pensilvâni
Phil Zimmermann (consultor independente)

Queremos agradecer também a toda a equipe da Addison Wesley, que fez um trabalho realmente nota vel (e que teve de suportar dois autores muito complicados e quase sempre atrasados). Nossos agradecimer tos a Marilyn Lloyd, Susan Hartman Sullivan, Patty Mahtani e Beth Paquin. Agradecemos aos artistas gráf cos Janet Theurer e Patrice Rossi Calkin, pelo trabalho que executaram nas figuras da segunda e da terceir edições deste livro, e a Nancy Kotary e Daniel Rausch pelo maravilhoso trabalho de produção desta edição Finalmente, um agradecimento muito especial a Susan, nossa editora na Addison Wesley. Este livro nã seria o que é (e talvez nem tivesse existido) sem sua administração cordial, constante incentivo, paciênci quase infinita, bom humor e perseverança.

Os editores da edição brasileira dedicam este livro ao prof. Nery Machado Filho (*in memoriam*).

Nota do Revisor Técnico

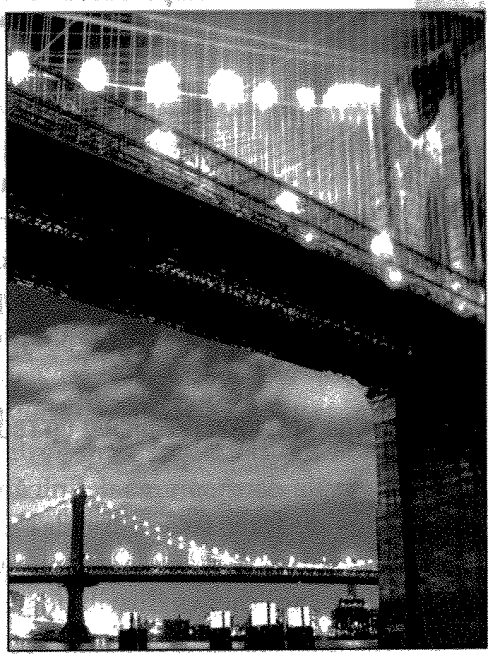
Mais uma vez foi com imensa satisfação que pude apreciar e rever a tradução do material apresentad em *Redes de computadores e a Internet: uma abordagem top-down*.

De minha parte, sou grato pelos comentários recebidos sobre a primeira edição e conto com a colabo ração de alunos, professores e profissionais de redes para a melhoria do trabalho.

WLZ

Capítulo 1

Redes de Computadores e a Internet



De browsers Web de telefones celulares a cafés que oferecem acesso sem fio à Internet, de redes domésticas com acesso de banda larga a infra-estruturas tradicionais de TI em ambientes de trabalho com PCs interligados em rede, carros em rede, redes de sensores ambientais, Internet interplanetária — quando achamos que as redes de computadores já estão praticamente presentes em toda parte, novas aplicações começam a ser desenvolvidas para ampliar ainda mais o alcance das redes existentes hoje! Este livro fornece uma introdução moderna ao campo dinâmico das redes de computadores, apresentando os princípios e o entendimento prático de que você precisará para utilizar não apenas as redes de hoje, mas também as de amanhã.

O primeiro capítulo apresenta um panorama de redes de computadores e da Internet. Nosso objetivo é pintar um quadro amplo que nos permita ver a floresta por entre as árvores. Cobriremos um terreno bastante extenso neste capítulo de introdução e discutiremos várias peças de uma rede de computadores, sem perder de vista o quadro geral. Este capítulo lança as fundações para o restante do livro.

O panorama geral de redes de computadores que apresentaremos neste capítulo será estruturado como segue. Após apresentarmos brevemente a terminologia e os conceitos fundamentais, examinaremos primeiramente os componentes básicos de hardware e software que compõem uma rede. Partiremos da periferia da rede e examinaremos os sistemas finais e aplicações de rede executados nela. Consideraremos os serviços de transporte fornecidos a essas aplicações. Em seguida exploraremos o cerne de uma rede de computadores examinando os enlaces e comutadores que transportam dados, bem como as redes de acesso e meios físicos que conectam sistemas finais ao núcleo da rede. Aprenderemos que a Internet é uma rede de redes e observaremos como essas redes se conectam umas com as outras.

Após concluirmos essa revisão sobre a periferia e o núcleo de uma rede de computadores, adotaremos uma visão mais ampla e mais abstrata na segunda metade deste capítulo. Examinaremos as causas do atraso de transferência de dados e das perdas em uma rede de computadores e forneceremos modelos quantitativos simples para o atraso fim-a-fim, modelos que levam em conta atrasos de transmissão, propagação e fila. Depois apresentaremos alguns princípios fundamentais de arquitetura em redes de computadores, a saber: protocolos em camadas e modelos de serviço. Finalmente, encerraremos este capítulo com um breve histórico da computação em rede.

1.1 O que é a Internet?

Neste livro, usamos a Internet pública, uma rede de computadores específica, como o veículo principal para discutir as redes de computadores e seus protocolos. Mas o que é a Internet? Gostaríamos de poder defini-la em uma única frase que você pudesse repetir para seus familiares e amigos, mas infelizmente a Internet é muito complexa e está sempre mudando, tanto no que se refere a seus componentes de hardware e software quanto aos serviços que oferece.

1.1.1 Uma descrição detalhada da rede

Em vez de darmos uma definição em uma única frase, tentaremos uma abordagem mais descritiva. Há dois modos de fazer isso: um deles é descrever a Internet em detalhes, isto é, os componentes básicos de hardware e software que a formam; outro é descrever a Internet como uma infra-estrutura de rede que provê serviços para aplicações distribuídas. Começaremos com a descrição detalhada, usando a Figura 1.1 para ilustrar nossa discussão.

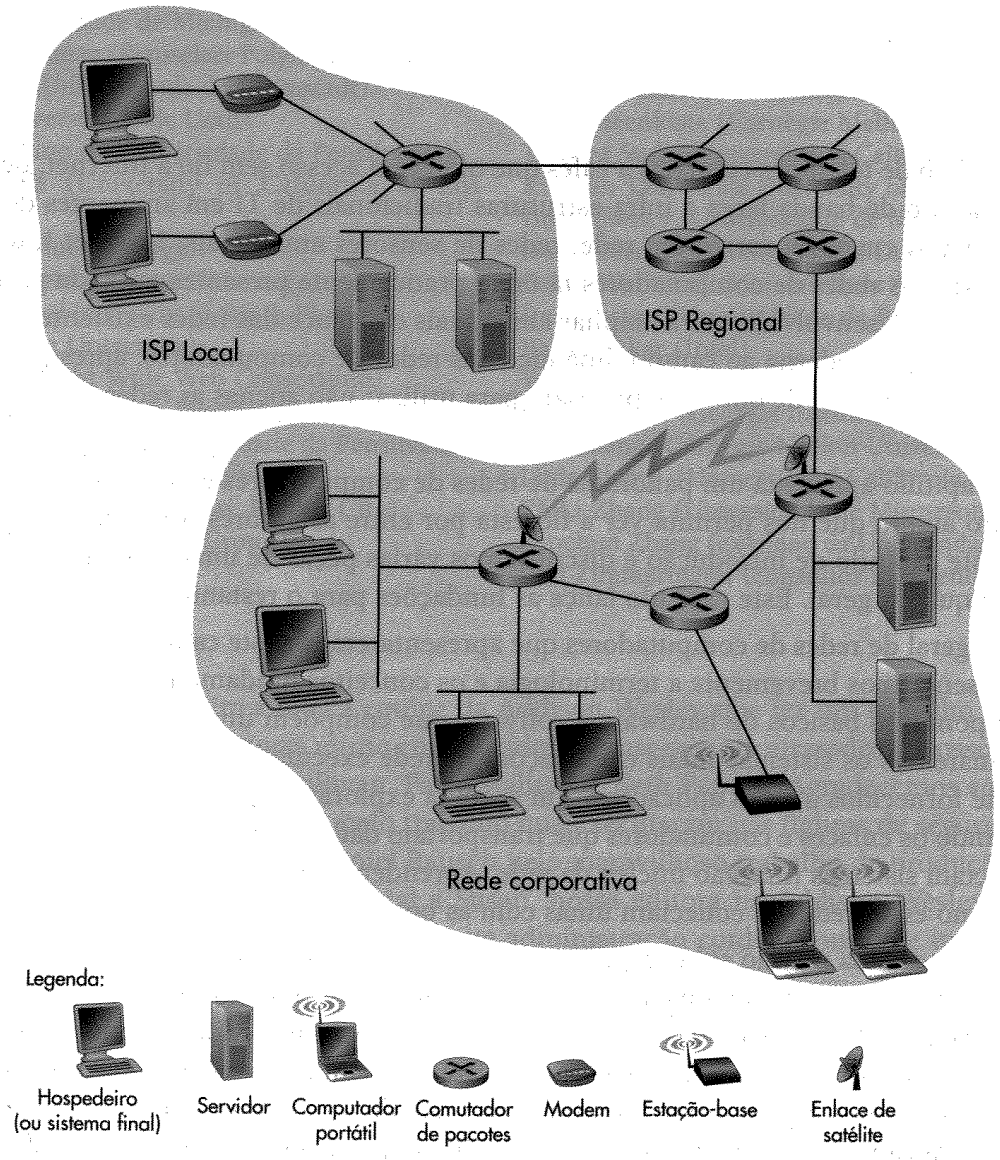


Figura 1.1 Alguns componentes da Internet

A Internet pública é uma rede de computadores mundial, isto é, uma rede que interconecta milhões de equipamentos de computação em todo o mundo. Não faz muito tempo, esses equipamentos eram primordialmente PCs tradicionais de mesa, estações de trabalho com sistema Unix e os chamados servidores que armazenam e transmitem informações, como páginas Web e mensagens de e-mail. Porém, sistemas finais que não são componentes tradicionais da Internet, como agendas digitais (PDAs), TVs, computadores portáteis, telefones celulares, automóveis, equipamentos de sensoriamento ambiental, telas de fotos, sistemas domésticos elétricos e de segurança, câmeras Web e até mesmo torradeiras estão sendo cada vez mais conectados à Internet [BBC, 2001]. Realmente, o termo *rede de computadores* está começando a soar um tanto desatualizado, dados os muitos equipamentos não tradicionais que estão sendo ligados à Internet. No jargão da Internet, todos esses equipamentos são denominados **hospedeiros** ou **sistemas finais**. Em janeiro de 2003, havia mais de 233 milhões de sistemas finais usando a Internet, e esse número continua a crescer rapidamente [ISC, 2004].

Sistemas finais são conectados entre si por **enlaces** (links) **de comunicação**. Na Seção 1.4, veremos que há muitos tipos de enlaces de comunicação, que são constituídos de diferentes tipos de meios físicos, entre eles cabos coaxiais, fios de cobre, fibras óticas e ondas de rádio. Enlaces diferentes podem transmitir dados em taxas diferentes, sendo a **taxa de transmissão** de um enlace medida em bits por segundo.

Em geral, sistemas finais não são interligados diretamente por um único enlace de comunicação. Em vez disso, são interconectados indiretamente por equipamentos intermediários de comutação conhecidos como **comutadores de pacotes**. Um comutador de pacotes encaminha a informação que está chegando em um de seus enlaces de comunicação de entrada para um de seus enlaces de comunicação de saída. No jargão das redes de computadores, o bloco de informação é denominado **pacote.** Há comutadores de pacotes de todos os tipos e formas, mas os dois mais proeminentes na Internet de hoje são **roteadores** e **comutadores de camada de enlace** (*switches*). Esses dois tipos de comutadores encaminham pacotes a seus destinos finais. Examinaremos detalhadamente roteadores no Capítulo 4 e comutadores de camada de enlace no Capítulo 5.

A seqüência de enlaces de comunicação e comutadores de pacotes que um pacote percorre desde o sistema final remetente até o sistema final receptor é conhecida como **rota** ou **caminho** através da rede. Em vez de prover um caminho *dedicado* entre sistemas finais comunicantes, a Internet usa uma técnica conhecida como **comutação de pacotes**, que permite que vários sistemas finais comunicantes compartilhem ao mesmo tempo um caminho ou partes dele. As primeiras redes de comutação de pacotes, criadas na década de 1970, são os ancestrais mais antigos da Internet de hoje. A exata quantidade de tráfego na Internet de hoje é um assunto que gera certa controvérsia [Odylsko, 2003], mas, de acordo com estimativas conservadoras, a taxa mensal de tráfego em redes de longa distância dos Estados Unidos é de aproximadamente 100 mil terabytes por mês, com a quantidade de tráfego praticamente dobrando a cada ano.

Sistemas finais acessam a Internet por meio de **Provedores de Serviços de Internet** (Internet Service Providers — ISPs), entre eles ISPs residenciais como AOL ou sua empresa local de telefonia ou cabo; ISPs corporativos, ISPs de universidades e ISPs como o T-Mobile, que provê acesso sem fio em aeroportos, hotéis, cafés e outros locais públicos. Cada ISP é uma rede de comutadores de pacotes e enlaces de comunicação. ISPs provêem aos sistemas finais uma variedade de tipos de acesso à rede, incluindo acesso por modem discado de 56 kbps, acesso residencial de banda larga como modem de cabo coaxial ou DSL (linha digital de assinante), acesso por LAN de alta velocidade e acesso sem fio. ISPs também provêem acesso a provedores de conteúdo, conectando sites Web diretamente à Internet. Para permitir comunicação entre usuários da Internet e possibilitar a usuários acesso mundial ao conteúdo da Internet, esses ISPs de nível mais baixo são interconectados por meio de ISPs de nível mais alto, nacionais e internacionais, como AT&T e Sprint. Um ISP de nível mais alto consiste em roteadores de alta velocidade interconectados com enlaces de fibra ótica de alta velocidade. Cada rede ISP, seja de nível mais alto ou mais baixo, é gerenciada de forma independente, executa o protocolo IP (ver adiante) e obedece a certas convenções de nomeação e endereço. Examinaremos ISPs e sua interconexão mais detalhadamente na Seção 1.5.

Os sistemas finais, os comutadores de pacotes e outras peças da Internet executam **protocolos** que controlam o envio e o recebimento de informações dentro da Internet. O **TCP** (Transmission Control Protocol — Protocolo de Controle de Transmissão) e o **IP** (Internet Protocol — Protocolo da Internet) são dois dos protocolos mais importantes da Internet. O protocolo IP especifica o formato dos pacotes que são enviados e recebidos entre roteadores e sistemas finais. Os principais protocolos da Internet são conhecidos coletivamente como **TCP/IP**. Começaremos a examinar protocolos neste capítulo de introdução. Mas isso é só um começo — grande parte deste livro trata de protocolos de redes de computadores!

Dada a importância de protocolos para a Internet, é adequado que todos concordem sobre o que cada um dos protocolos faz. É aqui que os padrões entram em ação. **Padrões da Internet** são desenvolvidos pela IETF (Internet Engineering Task Force — Força de Trabalho de Engenharia de Internet) [IETF, 2004]. Os documentos padronizados da IETF são denominados **RFCs** (*request for comments* — pedido de comentários). Os RFCs começaram como solicitações gerais de comentários (por isso o nome) para resolver problemas de arquitetura que a precursora da Internet enfrentava. Os RFCs tendem a ser bastante técnicos e detalhados. Definem protocolos como TCP, IP, HTTP (para a Web) e SMTP (Protocolo Simples de Transferência de Correio — Simple Mail Transfer Protocol) (para e-mail padronizado). A IETF também padronizou quais protocolos devem ser executados por um hospedeiro da Internet [RFC 112; RFC 1123] e por um roteador da Internet [RFC 1812]. Existem mais de 3.500 RFCs. Outros organismos também especificam padrões para componentes de rede, mais notavelmente para enlaces de rede. O IEEE 802 LAN/MAN Standards Committee [IEEE, 802, 2004], por exemplo, especifica os padrões da Ethernet e Wi-Fi sem fio.

A Internet pública (isto é, a rede global de redes discutida anteriormente) é a rede a que normalmente nos referimos como *a* Internet. Também há muitas redes privadas, tais como redes corporativas e governamentais, cujos hospedeiros não podem trocar mensagens com hospedeiros que estão fora da rede privada (a menos que as mensagens passem por dispositivos denominados firewalls, que restringem o fluxo de mensagens para dentro e para fora da rede). Essas redes privadas são freqüentemente denominadas **intranets**, pois usam o mesmo tipo de hospedeiros, roteadores, enlaces e protocolos da Internet pública.

1.1.2 Uma descrição do serviço

A discussão anterior identificou muitos dos componentes que formam a Internet. Agora deixaremos de lado a descrição dos mecanismos e adotaremos uma visão mais abstrata, orientada para serviço:

- A Internet permite que **aplicações distribuídas** que executam em seus sistemas finais troquem dados entre si. Entre essas aplicações estão a navegação na Web, mensagem instantânea, áudio e vídeo em tempo real, telefonia pela Internet, jogos distribuídos, compartilhamento de arquivos peer-to-peer (P2P), login remoto, correio eletrônico e mais, muito mais. Vale enfatizar que a Web não é uma rede isolada, mas apenas uma das muitas aplicações distribuídas que utilizam os serviços de comunicação providos pela Internet.
- A Internet provê dois serviços a suas aplicações distribuídas: um **serviço confiável orientado para conexão** e um **serviço não confiável não orientado para conexão**. De maneira geral, o serviço confiável orientado para conexão garante que os dados transmitidos de uma origem a um destino sejam finalmente entregues ao destinatário em ordem e completos. O serviço não confiável não orientado para conexão não oferece nenhuma garantia quanto à entrega final. Comumente, uma aplicação distribuída usa um ou outro desses dois serviços (mas não ambos).
- Atualmente, a Internet não provê um serviço que ofereça garantias quanto ao *tempo que gastará* para levar os dados da origem ao destino. E, a menos que você aumente a taxa de transmissão do dispositivo de acesso a seu provedor de serviços de Internet, hoje não é possível pagar mais para obter um serviço melhor (por exemplo, atrasos limitados) — uma situação que alguns (particularmente os norte-americanos) acham estranha. Examinaremos as pesquisas mais avançadas da Internet que visam à mudança dessa situação no Capítulo 7.

Esta segunda descrição da Internet — isto é, em termos dos serviços que ela provê a aplicações distribuídas — é muito importante. Cada vez mais, os avanços na tecnologia dos componentes da Internet estão sendo guiados pelas necessidades de novas aplicações. Portanto, é importante ter sempre em mente que a Internet é uma infra-estrutura na qual novas aplicações estão constantemente sendo inventadas e disponibilizadas.

Acabamos de apresentar duas descrições da Internet: uma delas diz respeito a seus componentes de hardware e software, e a outra, aos serviços que ela oferece a aplicações distribuídas. Mas talvez você ainda esteja confuso sobre o que é a Internet. O que é comutação de pacotes, TCP/IP e serviço orientado para conexão? O que são roteadores? Que tipos de enlaces de comunicação estão presentes na Internet? O que é uma aplicação distribuída? Como uma torradeira ou um sensor de variações meteorológicas pode ser ligado à Internet? Se você está um pouco assustado com tudo isso agora, não se preocupe — a finalidade deste livro é lhe apresentar os mecanismos da Internet e também os princípios que determinam como e por que ela funciona. Explicaremos esses termos e questões importantes nas seções e nos capítulos subseqüentes.

1.1.3 O que é um protocolo?

Agora que já entendemos um pouco o que é a Internet, vamos considerar uma outra palavra fundamental usada em redes de computadores: *protocolo*. O que *é* um protocolo? O que um protocolo faz? Como reconhecer um protocolo?

Uma analogia humana

Provavelmente é mais fácil entender a idéia de um protocolo de rede de computadores considerando primeiramente algumas analogias humanas, já que executamos protocolos o tempo todo. Considere o que você faz quando quer perguntar as horas a alguém. Um diálogo comum é ilustrado na Figura 1.2.

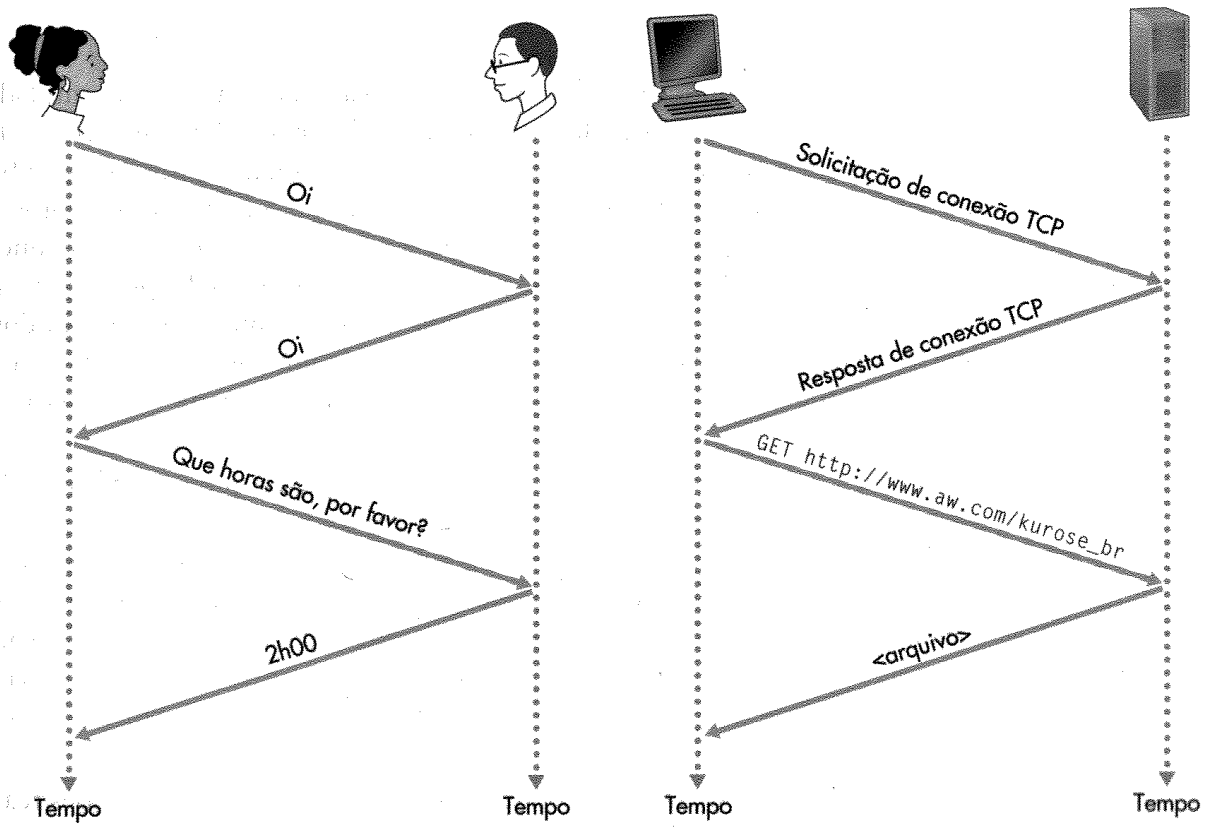


Figura 1.2 Um protocolo humano e um protocolo de rede de computadores

O protocolo humano (ou as boas maneiras, ao menos) ordena que, ao iniciarmos uma comunicação com outra pessoa, primeiramente a cumprimentemos (o primeiro "oi" da Figura 1.2). A resposta comum para um "oi" é um outro "oi". Implicitamente, tomamos a resposta cordial "oi" como uma indicação de que podemos prosseguir e perguntar as horas. Uma resposta diferente ao "oi" inicial (tal como "Não me perturbe!", "I don't speak Portuguese!" ou alguma resposta impublicável) poderia indicar falta de vontade ou incapacidade de comunicação. Nesse caso, o protocolo humano seria não perguntar que horas são. Às vezes, não recebemos nenhuma resposta para uma pergunta, caso em que normalmente desistimos de perguntar as horas à pessoa. Note que, no nosso protocolo humano, *há mensagens específicas que enviamos e ações específicas que realizamos em reação às respostas recebidas ou a outros eventos (como nenhuma resposta após certo tempo)*. É claro que mensagens transmitidas e recebidas e ações realizadas quando essas mensagens são enviadas ou recebidas ou quando ocorrem outros eventos desempenham um papel central em um protocolo humano. Se as pessoas executarem protocolos diferentes (por exemplo, se uma pessoa tem boas maneiras, mas a outra não; se uma delas entende o conceito de horas, mas a outra não), os protocolos não interagem e nenhum trabalho útil pode ser realizado. O mesmo é válido para redes — é preciso que duas (ou mais) entidades comunicantes executem o mesmo protocolo para que uma tarefa seja realizada.

Vamos considerar uma segunda analogia humana. Suponha que você esteja assistindo a uma aula (sobre redes de computadores, por exemplo). O professor está falando monotonamente sobre protocolos e você está confuso. Ele pára e pergunta: "Alguma dúvida?" (uma mensagem que é transmitida a todos os alunos e recebida por todos os que não estão dormindo). Você levanta a mão (transmitindo uma mensagem implícita ao professor). O professor percebe e, com um sorriso, diz "Sim..." (uma mensagem transmitida, incentivando-o a fazer sua pergunta — professores *adoram* perguntas) e você então faz sua pergunta (isto é, transmite sua mensagem ao professor). O professor ouve (recebe sua mensagem) e responde (transmite uma resposta a você). Mais uma vez, percebemos que a transmissão e a recepção de mensagens e um conjunto de ações convencionais, realizadas quando as mensagens são enviadas e recebidas, são o coração desse protocolo de pergunta e resposta.

Protocolos de rede

Um protocolo de rede é semelhante a um protocolo humano; a única diferença é que as entidades que trocam mensagens e realizam ações são componentes de hardware ou software de algum equipamento (por exemplo, computador, roteador ou outro equipamento habilitado para rede). Todas as atividades na Internet que envolvem duas ou mais entidades remotas comunicantes são governadas por um protocolo. Por exemplo, protocolos implementados em hardware nas placas de interface de rede de dois computadores conectados fisicamente controlam o fluxo de bits no 'cabo' entre as duas placas de interface de rede; protocolos de controle de congestionamento em sistemas finais controlam a taxa com que os pacotes são transmitidos entre a origem e o destino; protocolos em roteadores determinam o caminho de um pacote da fonte ao destino. Protocolos estão em execução por toda a Internet e, conseqüentemente, grande parte deste livro trata de protocolos de rede de computadores.

Como exemplo de um protocolo de rede de computadores com o qual você provavelmente está familiarizado, considere o que acontece quando fazemos uma requisição a um servidor Web, isto é, quando digitamos o URL de uma página Web no browser. Isso é mostrado no lado direito da Figura 1.2. Primeiramente, o computador enviará uma mensagem de requisição de conexão ao servidor Web e aguardará uma resposta. O servidor receberá essa mensagem de requisição de conexão e retornará uma mensagem de resposta de conexão. Sabendo que agora está tudo certo para requisitar o documento da Web, o computador envia então o nome da página Web que quer buscar naquele servidor com uma mensagem GET. Por fim, o servidor retorna a página da Web (arquivo) para o computador.

Dados o exemplo humano e o exemplo de rede anteriores, as trocas de mensagens e as ações realizadas quando essas mensagens são enviadas e recebidas são os elementos fundamentais para a definição de um protocolo:

Um protocolo define o formato e a ordem das mensagens trocadas entre duas ou mais entidades comunicantes, bem como as ações realizadas na transmissão e/ou no recebimento de uma mensagem ou outro evento.

A Internet e as redes de computadores em geral fazem uso intenso de protocolos. Diferentes tipos de protocolos são usados para realizar diferentes tarefas de comunicação. À medida que for avançando na leitura deste livro, você perceberá que alguns protocolos são simples e diretos, enquanto outros são complexos e intelectualmente profundos. Dominar a área de redes de computadores equivale a entender o que são, por que existem e como funcionam os protocolos de rede.

1.2 A periferia da Internet

Nas seções anteriores, apresentamos uma descrição da Internet e de protocolos de rede fazendo analogias com nossos atos. Agora passaremos a tratar com um pouco mais de profundidade os componentes de uma rede de computadores (e da Internet, em particular). Nesta seção, começamos pela periferia de uma rede e examinamos os componentes com os quais estamos mais familiarizados — a saber, os computadores que usamos diariamente. Na seção seguinte, passaremos da periferia para o núcleo da rede e estudaremos comutação e roteamento em redes de computadores. Então, na Seção 1.4, discutiremos as ligações físicas que transmitem os sinais enviados entre computadores e comutadores.

1.2.1 Sistemas finais, clientes e servidores

No jargão de redes de computadores, os computadores conectados à Internet são usualmente denominados **sistemas finais**, porque estão na periferia da Internet, como ilustrado na Figura 1.3.

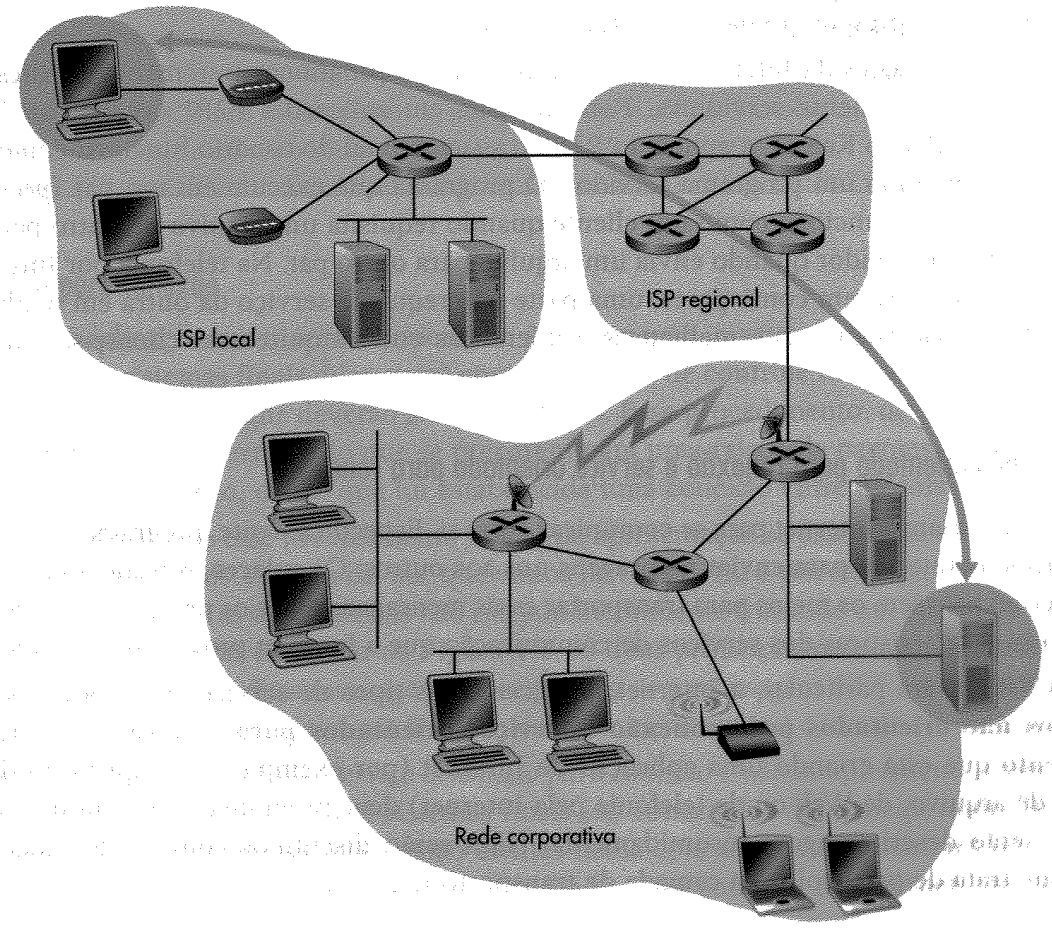


Figura 1.3 Interação entre sistemas finais

Entre os sistemas finais da Internet estão computadores de mesa (por exemplo, PCs de mesa, Macs, estações de trabalho UNIX), servidores (por exemplo, servidores Web e de e-mail) e computadores móveis (por exemplo, computadores portáteis, PDAs e telefones com conexões sem fio à Internet). Além do mais, um número cada vez maior de equipamentos alternativos como terminais clientes e eletroeletrônicos [Thinplanet, 2002], Web TVs e transceptores [Nesbitt], câmeras digitais, eletrodomésticos, equipamentos de piso de fábrica e sensores ambientais estão sendo ligados à Internet como sistemas finais (ver barra lateral).

Sistemas finais também são denominados *hospedeiros* (*hosts*) porque hospedam (isto é, executam) programas de aplicação, tais como um programa de browser da Web, um programa de servidor da Web, um programa leitor de e-mail ou um servidor de e-mail. Neste livro, utilizaremos os termos hospedeiros e sistemas finais como sinônimos. Às vezes, sistemas finais são ainda subdivididos em duas categorias: **clientes** e **servidores**. Informalmente, clientes costumam ser PCs de mesa ou móveis, PDAs e assim por diante, ao passo que servidores tendem a ser máquinas mais poderosas que armazenam e distribuem páginas Web, vídeo em tempo real, retransmissão de e-mails e assim por diante.

No contexto de software de rede há ainda uma outra definição de cliente e de servidor, à qual nos referiremos em todo o livro. Um **programa cliente** é um programa que funciona em um sistema final, que solicita e recebe um serviço de um **programa servidor**, que funciona em um outro sistema final. Esse modelo cliente-servidor, que estudaremos detalhadamente no Capítulo 2 é, sem dúvida, a estrutura mais predominante para aplicações da Internet. A Web, o e-mail, a transferência de arquivo, o login remoto (por exemplo, Telnet), os grupos de discussão e muitas outras aplicações populares adotam o modelo cliente–servidor. Uma vez que um programa cliente normalmente roda em um computador e o programa servidor, em outro, aplicações cliente–servidor de Internet são, por definição, **aplicações distribuídas**. O programa cliente e o programa servidor interagem enviando mensagens um para o outro pela Internet. Nesse nível de abstração, os roteadores, enlaces e outros componentes da Internet funcionam como uma caixa-preta que transfere mensagens entre os componentes distribuídos, comunicantes, de uma aplicação da Internet. Esse é o nível de abstração representado na Figura 1.3.

Nem todas as aplicações da Internet de hoje consistem em programas puramente clientes que interagem com programas puramente servidores. Por exemplo, em aplicações P2P* de compartilhamento de arquivos populares, como o KaZaA, a aplicação P2P no sistema final do usuário funciona como um programa cliente e também como um programa servidor. O programa que executa em um par (*peer*) (isto é, na máquina de um usuário) funciona como um cliente quando requisita um arquivo de outro par, e o programa funciona como um servidor quando envia um arquivo para outro par. Na telefonia por Internet, as duas partes comunicantes interagem como pares; uma parte não requisita serviço da outra em nenhum sentido. Examinaremos detalhadamente as semelhanças e diferenças entre arquiteturas cliente-servidor e P2P no Capítulo 2.

1.2.2 Serviço não orientado para conexão e serviço orientado para conexão

Sistemas finais usam a Internet para se comunicar entre si. Especificamente, programas de sistemas finais usam os serviços da Internet para enviar mensagens uns aos outros. Os enlaces, roteadores e outros componentes da Internet provêem os meios para transportar essas mensagens entre os programas de sistemas finais. Mas quais são as características dos serviços de comunicação que a Internet provê a seus sistemas finais?

Redes TCP/IP e, em particular, a Internet, oferecem dois tipos de serviços às aplicações de sistemas finais: **serviços não orientados para conexão** e **serviços orientados para conexão**. Um profissional de desenvolvimento que está criando uma aplicação de Internet (por exemplo, uma aplicação de e-mail, de transferência de arquivo, de Web ou de telefonia pela Internet) deve programá-la para usar um desses dois serviços, que serão descritos agora resumidamente. (Eles serão discutidos com muito mais detalhes no Capítulo 3, que trata de protocolos da camada de transporte.)

* Lê-se 'peer-to-peer'. (N. do R.T.)

História

Um Conjunto Impressionante de Sistemas Finais da Internet

Não faz muito tempo, os sistemas finais conectados à Internet eram primordialmente computadores tradicionais, como máquinas de mesa e servidores de grande capacidade. Desde o final da década de 1990 até hoje, um amplo leque de equipamentos e dispositivos interessantes, cada vez mais diversos, vem sendo conectado à Internet. A característica comum desses equipamentos é que eles precisam enviar e receber dados digitais de, e para, outros equipamentos. Tendo em vista a onipresença da Internet, seus protocolos bem definidos (padronizados) e a disponibilidade comercial de hardware capacitado para ela, é natural usar sua tecnologia para interconectar esses equipamentos.

Alguns deles parecem ter sido criados exclusivamente para diversão. Um computador de mesa utilizando IP com recurso de moldura de foto [Ceiva, 2004] baixa fotos digitais de um servidor remoto e as apresenta em um dispositivo que parece uma moldura tradicional de fotografia; uma torradeira da Internet baixa informações meteorológicas de um servidor e grava uma imagem da previsão do tempo do dia em questão (por exemplo, nublado, com sol) na sua torrada matinal [BBC, 2001]. Outros dispositivos fornecem informações úteis – câmeras Web apresentam as condições meteorológicas e de tráfego ou fazem monitoramento para fornecer uma localização de interesse; máquinas de lavar roupas conectadas à Internet podem ser monitoradas remotamente por meio de um browser e geram e-mail quando terminam de lavar uma carga de roupa. Telefones celulares que utilizam IP permitem fácil navegação pela Web e transmissão de e-mails e mensagens. Uma nova classe de sistemas de sensoriamento em rede promete revolucionar o modo como observamos e interagimos com nosso ambiente. Redes de sensores incorporadas ao ambiente físico permitem a monitoração de edifícios, pontes e outras estruturas construídas pelo ser humano [Elgamal, 2001]; de atividade sísmica [CISN, 2004]; de hábitats de vida selvagem [Mainwaring, 2001]; de estuários de rios [Baptista, 2003]; de funções biomédicas [Schwiebert, 2001]; de ventos e acidentes meteorológicos na camada atmosférica mais baixa [CASA, 2004] – e disponibilizam essas informações para usuários remotos. O Center for Embedded Networked Sensing da UCLA [SENS, 2004] é um Centro de Ciência e Tecnologia da NSF cuja meta é utilizar tecnologia de rede de sensores incorporados em aplicações científicas e sociais críticas.

Serviço orientado para conexão

Quando uma aplicação usa o serviço orientado para conexão, o programa cliente e o programa servidor (que residem em sistemas finais diferentes) enviam pacotes de controle um para o outro antes de remeter pacotes com os dados reais que deverão ser transferidos. Esse procedimento, denominado apresentação, alerta o cliente e o servidor, permitindo que se preparem para uma rajada de pacotes. Uma vez concluído o procedimento de apresentação, diz-se que foi estabelecida uma conexão entre os dois sistemas finais.

É interessante notar que esse procedimento inicial de apresentação se assemelha ao protocolo usado na interação entre seres humanos. A troca de 'ois' que vimos na Figura 1.2 é um exemplo de protocolo humano de apresentação (mesmo que as duas pessoas não estejam literalmente trocando um aperto de mãos). No caso da interação pela Web também mostrada na Figura 1.2, as duas primeiras mensagens trocadas também são mensagens de apresentação. As duas mensagens subseqüentes — a mensagem GET e a mensagem de resposta contendo o arquivo — incluem dados reais e são enviadas somente após a conexão ter sido estabelecida.

Por que se utiliza a terminologia *serviço orientado para conexão* e não simplesmente *serviço de conexão*? Essa terminologia deve-se ao fato de que sistemas finais são conectados de maneira muito livre. Em particular, somente os próprios sistemas finais estão cientes dessa conexão; os comutadores de pacotes dentro da Internet ficam completamente alheios a ela. Na verdade, uma conexão na Internet nada mais é do que buffers alocados e variáveis de estado nos sistemas finais; os comutadores de pacotes intervenientes não mantêm nenhuma informação de estado de conexão.

Os serviços orientados para conexão providos pela Internet vêm conjugados com diversos outros serviços, inclusive transferência de dados confiável, controle de fluxo e controle de congestionamento. A expressão **transferência de dados confiável** quer dizer que uma aplicação pode confiar que a conexão entregará todos os seus dados sem erro e na ordem certa. A confiabilidade na Internet é conseguida por meio da utilização de confirmações e retransmissões. Para ter uma idéia preliminar de como a Internet implementa o serviço de transporte confiável, considere uma aplicação que tenha estabelecido uma conexão entre os sistemas finais A e B. Quando o sistema final B recebe um pacote de A, envia uma confirmação; quando o sistema final A recebe a confirmação, sabe que o pacote correspondente foi definitivamente recebido. Quando o sistema final A não recebe uma confirmação, presume que o pacote enviado não foi recebido por B e, por conseguinte, o retransmite. O **controle de fluxo** garante que nenhum dos lados de uma conexão sobrecarregue o outro enviando demasiados pacotes muito rapidamente. Veremos no Capítulo 3 que a Internet implementa o serviço de controle de fluxo utilizando buffers de envio e de recebimento nos sistemas finais comunicantes. O serviço de **controle de congestionamento** da Internet ajuda a evitar que ela trave. Quando um comutador de pacotes fica congestionado, seus buffers podem transbordar e pode ocorrer perda de pacotes. Nessas circunstâncias, se todo par de sistemas finais comunicantes continuar a transmitir pacotes para a rede o mais rápido que puder, ocorrerá um travamento e poucos pacotes chegarão ao seu destino. A Internet evita esse problema forçando sistemas finais a reduzir a velocidade em que enviam pacotes à rede durante períodos de congestionamento. Sistemas finais são alertados da existência de congestionamento intenso quando param de receber confirmações para os pacotes que enviaram.

Enfatizamos neste texto que, embora o serviço orientado para conexão da Internet venha conjugado com serviços de transferência de dados confiável, controle de fluxo e controle de congestionamento, essas três características não são, de modo algum, componentes essenciais de um serviço orientado para conexão. Um tipo diferente de rede de computadores pode prover um serviço orientado para conexão às suas aplicações sem conjugá-lo com uma ou mais dessas características. Na verdade, qualquer protocolo que execute apresentação entre as entidades comunicantes antes de transferir dados é um serviço orientado para conexão [Iren, 1999].

O serviço orientado para conexão da Internet tem um nome — **Protocolo de Controle de Transmissão** (Transmission Control Protocol — TCP). A versão inicial do protocolo TCP é definida no RFC 793 da Internet [RFC 793]. Entre os serviços que o TCP provê a uma aplicação estão transporte confiável, controle de fluxo e controle de congestionamento. O protocolo provê uma **abstração de corrente de bytes**, entregando de modo confiável uma corrente de bytes do remetente ao receptor. É importante notar que uma aplicação precisa se preocupar somente com os serviços fornecidos; não precisa se preocupar com o *modo como* o TCP implementa confiabilidade, controle de fluxo ou controle de congestionamento. Nós, é claro, estamos *muito* interessados em como o TCP implementa esses serviços, e abordaremos esses tópicos detalhadamente no Capítulo 3.

Serviço não orientado para conexão

Não há apresentação mútua no serviço não orientado para conexão da Internet. Quando um lado de uma aplicação quer enviar pacotes ao outro lado, o programa remetente simplesmente os envia. Como não há procedimento de apresentação mútua antes da transmissão de pacotes de dados, os dados podem ser entregues mais rápido, o que torna o serviço não orientado para conexão ideal para aplicações simples orientadas para transação. Mas, como também não há nenhuma transferência confiável de dados, uma fonte nunca sabe com certeza quais pacotes chegaram ao destino. Além disso, o serviço não orientado para conexão da Internet não provê controle de fluxo, nem controle de congestionamento. O serviço de Internet não orientado para conexão é denominado **Protocolo de Datagrama do Usuário** (User Datagram Protocol — UDP), cuja definição pode ser encontrada no RFC 768.

A maioria das aplicações mais conhecidas da Internet usa TCP, o serviço orientado para conexão da Internet. Entre essas aplicações estão: Telnet (para login remoto), SMTP (para correio eletrônico), FTP (para transferência de arquivos) e HTTP (para a Web). Não obstante, UDP, o serviço não orientado para conexão da Internet, é usado por inúmeras aplicações, inclusive por inúmeras aplicações de multimídia que estão surgindo, como telefone por Internet e videoconferência.

1.3 O núcleo da rede

Após termos examinado os sistemas finais e o modelo de serviço de transporte fim-a-fim da Internet na Seção 1.3, agora examinaremos mais profundamente o interior da rede. Na próxima seção, estudaremos o núcleo da rede — a malha de roteadores que interconecta os sistemas finais da Internet. A Figura 1.4 destaca o núcleo da rede com linhas contínuas e sombreadas.

1.3.1 Comutação de circuitos e comutação de pacotes

Há duas abordagens fundamentais para montagem de um núcleo de rede: **comutação de circuitos** e **comutação de pacotes**. Em redes de comutação de circuitos, os recursos necessários ao longo de um caminho (buffers, taxa de transmissão de enlaces) para prover comunicação entre os sistemas finais são *reservados* pelo período da sessão de comunicação. Em redes de comutação de pacotes, esses recursos *não* são reservados; as mensagens de uma sessão usam os recursos por demanda e, como conseqüência, poderão ter de esperar (isto é, entrar na fila) para conseguir acesso a um enlace de comunicação. Como simples analogia, considere dois restaurantes — um que exige e outro que não exige nem aceita reserva. Se quisermos ir ao restaurante que exige reserva, teremos de passar pelo aborrecimento de telefonar antes de sair de casa. Mas, quando chegarmos lá, poderemos, em princípio, ser imediatamente atendidos e servidos. No restaurante que não exige reserva, não precisaremos nos dar ao trabalho de reservar mesa, porém, quando lá chegarmos, talvez tenhamos de esperar.

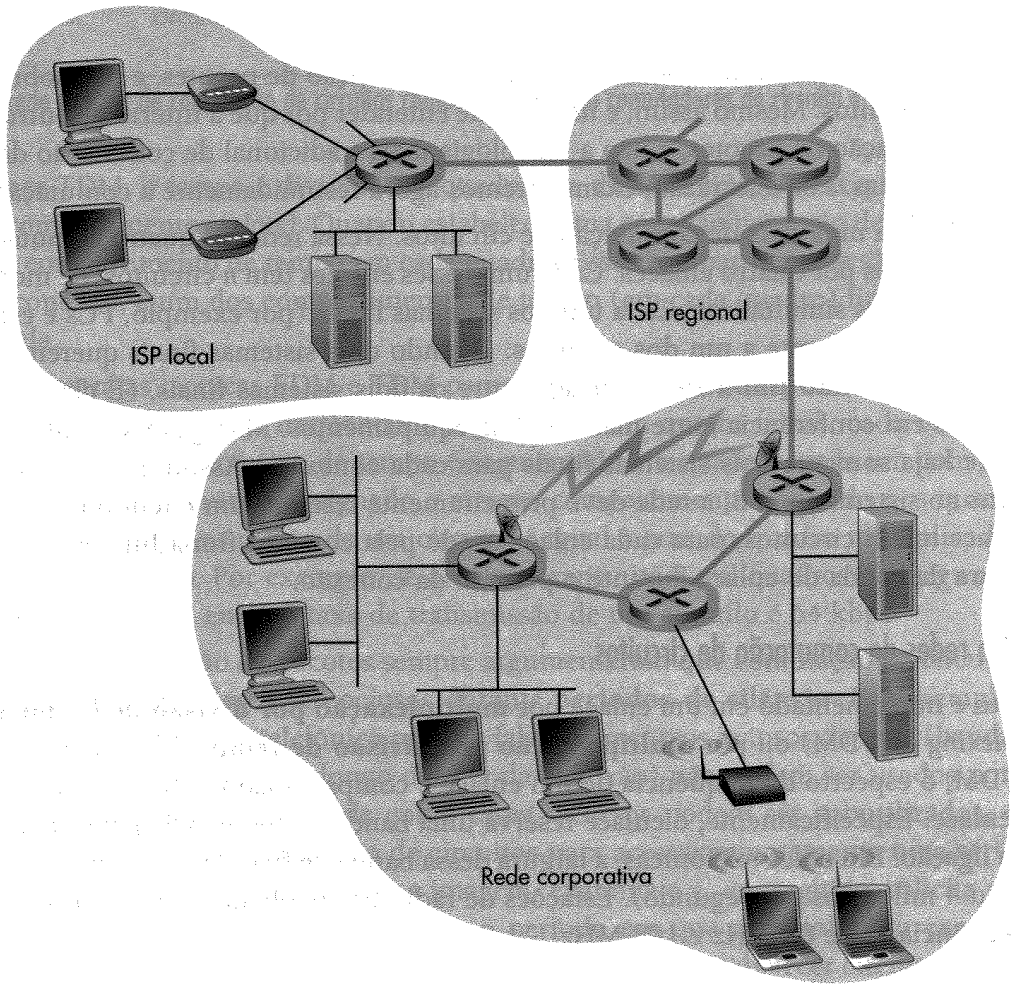


Figura 1.4 O núcleo da rede

As onipresentes redes de telefonia são exemplos de redes de comutação de circuitos. Considere o que acontece quando uma pessoa quer enviar a outra uma informação (por voz ou por fax) por meio de uma rede telefônica. Antes que o remetente possa enviar a informação, a rede precisa estabelecer uma conexão entre o remetente e o destinatário. Ao contrário da conexão TCP, que discutimos na seção anterior, essa é uma conexão forte, na qual os comutadores existentes no caminho entre o remetente e o destinatário mantêm o estado dessa conexão. No jargão da telefonia, essa conexão é denominada **circuito**. Quando a rede estabelece o circuito, também reserva uma taxa de transmissão constante nos enlaces da rede durante o período da conexão. Visto que foi reservada largura de banda para essa conexão remetente-destinatário, o remetente pode transferir dados ao destinatário a uma taxa constante *garantida*.

Hoje, a Internet é a quintessência das redes de comutação de pacotes. Considere o que ocorre quando um sistema final quer enviar um pacote a outro sistema final pela Internet. Como acontece na comutação de circuitos, o pacote é transmitido por uma série de enlaces de comunicação. Mas, na comutação de pacotes, o pacote é enviado à rede sem reservar nenhuma largura de banda. Se um dos enlaces estiver congestionado porque outros pacotes precisam ser transmitidos pelo enlace ao mesmo tempo, então nosso pacote terá de esperar em um buffer na extremidade de origem do enlace de transmissão e sofrerá um atraso. A Internet faz o *melhor esforço* para entregar os dados prontamente, mas não dá nenhuma garantia.

Nem todas as redes de telecomunicação podem ser classificadas exatamente como redes de comutação de circuitos puras ou redes de comutação de pacotes puras. Não obstante, essa classificação fundamental em redes de comutação de pacotes e de comutação de circuitos é um excelente ponto de partida para a compreensão da tecnologia de redes de telecomunicação.

Comutação de circuitos

Este livro é sobre redes de computadores, Internet e comutação de pacotes, e não sobre redes telefônicas e comutação de circuitos. Mesmo assim, é importante entender por que a Internet e outras redes de computadores usam comutação de pacotes, e não a tecnologia mais tradicional de comutação de circuitos utilizada nas redes telefônicas. Por essa razão, examinaremos agora resumidamente a comutação de circuitos.

A Figura 1.5 ilustra uma rede de comutação de circuitos. Nessa rede, os quatro comutadores de circuitos estão interconectados por quatro enlaces. Cada um desses enlaces tem n circuitos, de modo que cada um pode suportar n conexões simultâneas. Cada um dos sistemas finais (por exemplo, PCs e estações de trabalho) está conectado diretamente a um dos circuitos. Quando dois sistemas finais querem se comunicar, a rede estabelece uma **conexão fim-a-fim** dedicada entre os dois sistemas finais. (É claro que também são possíveis chamadas em conferência entre mais de dois equipamentos. Mas, para simplificar, por enquanto vamos supor que haja somente dois sistemas finais para cada conexão.) Assim, para que o sistema final A envie mensagens ao sistema final B, a rede deve primeiramente reservar um circuito em cada dois enlaces. Como cada enlace tem n circuitos, para cada enlace usado pela conexão fim-a-fim, esta fica com uma fração $1/n$ da largura de banda do enlace durante o período da conexão.

Multiplexação em redes de comutação de circuitos

Um circuito é implementado em um enlace por **multiplexação por divisão de freqüência** (*frequency-division multiplexing* — FDM) ou por **multiplexação por divisão de tempo** (*time-division multiplexing* — TDM). Com FDM, o espectro de freqüência de um enlace é compartilhado entre as conexões estabelecidas através desse enlace. Especificamente, o enlace reserva uma banda de freqüência para cada conexão durante o período da ligação. Em redes telefônicas, a largura dessa banda de freqüência normalmente é 4 kHz (isto é, 4 mil Hertz ou 4 mil ciclos por segundo). Estações de rádio FM também usam FDM para compartilhar o espectro de freqüência entre 88 MHz e 108 MHz.

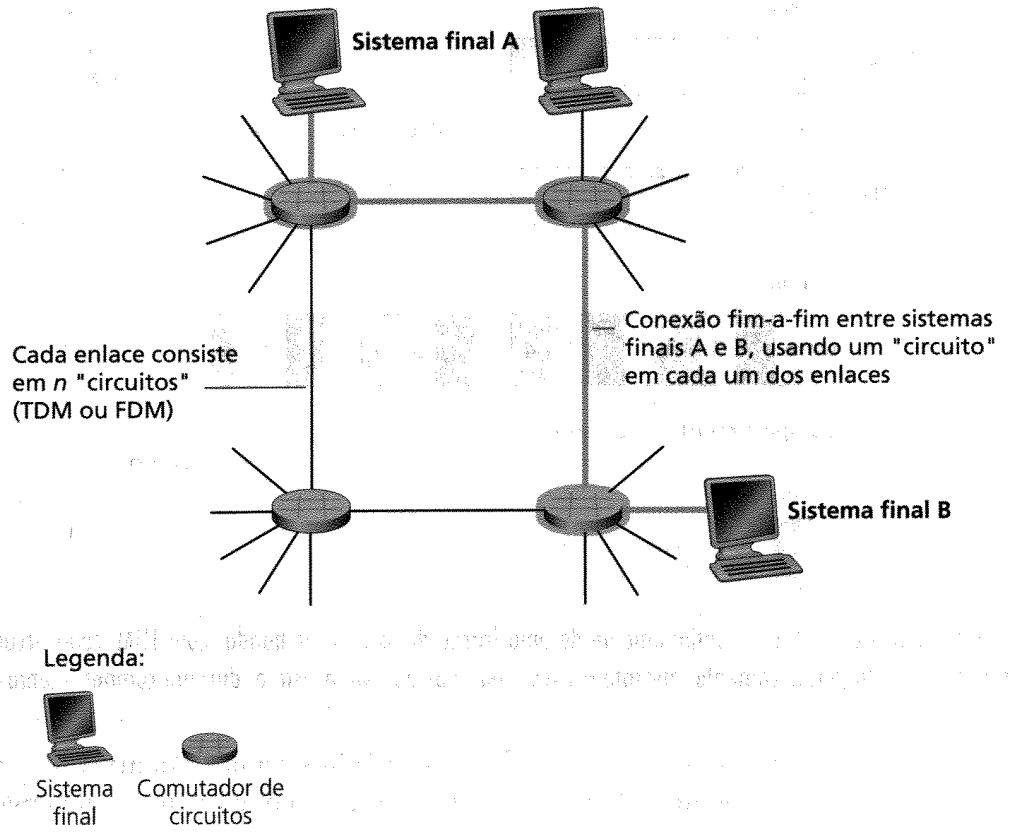


Figura 1.5 Uma rede simples de comutação de circuitos composta de quatro comutadores de circuito e quatro enlaces

Em um enlace TDM, o tempo é dividido em quadros de duração fixa, e cada quadro é dividido em um número fixo de compartimentos (*slots*). Quando estabelece uma conexão por meio de um enlace, a rede dedica à conexão um compartimento de tempo em cada quadro. Esses compartimentos são reservados para o uso exclusivo dessa conexão, e um dos compartimentos de tempo (em cada quadro) fica disponível para transmitir os dados dela.

A Figura 1.6 ilustra as técnicas FDM e TDM para um enlace de rede específico que suporta até quatro circuitos. Para FDM, o domínio de freqüência é segmentado em quatro faixas, cada uma com largura de banda de 4 kHz. Para TDM, o domínio de tempo é segmentado em quadros, cada um com quatro compartimentos de tempo; a cada circuito é designado o mesmo compartimento dedicado nos quadros sucessivos TDM. Para TDM, a taxa de transmissão de um circuito é igual à taxa do quadro multiplicada pelo número de bits em um compartimento. Por exemplo, se o enlace transmite 8 mil quadros por segundo e cada compartimento consiste em 8 bits, então a taxa de transmissão de um circuito é 64 kbps.

Os defensores da comutação de pacotes sempre argumentaram que comutação de circuitos é desperdício, porque os circuitos dedicados ficam ociosos durante **períodos de silêncio**. Por exemplo, quando um dos participantes de uma conversa telefônica pára de falar, os recursos ociosos da rede (bandas de freqüências ou compartimentos nos enlaces ao longo da rota da conexão) não podem ser usados por outras conexões em curso. Para um outro exemplo de como esses recursos podem ser subutilizados, considere um radiologista que usa uma rede de comutação de circuitos para acessar remotamente uma série de radiografias. Ele estabelece uma conexão, requisita uma imagem, examina-a e, em seguida, solicita uma nova imagem. Recursos de rede são desperdiçados durante o período em que o radiologista examina a imagem. Defensores da comutação de pacotes também gostam de destacar que estabelecer circuitos fim-a-fim e reservar larguras de faixas fim-a-fim é complicado e exige software complexo de sinalização para coordenar a operação dos comutadores ao longo do caminho fim-a-fim.

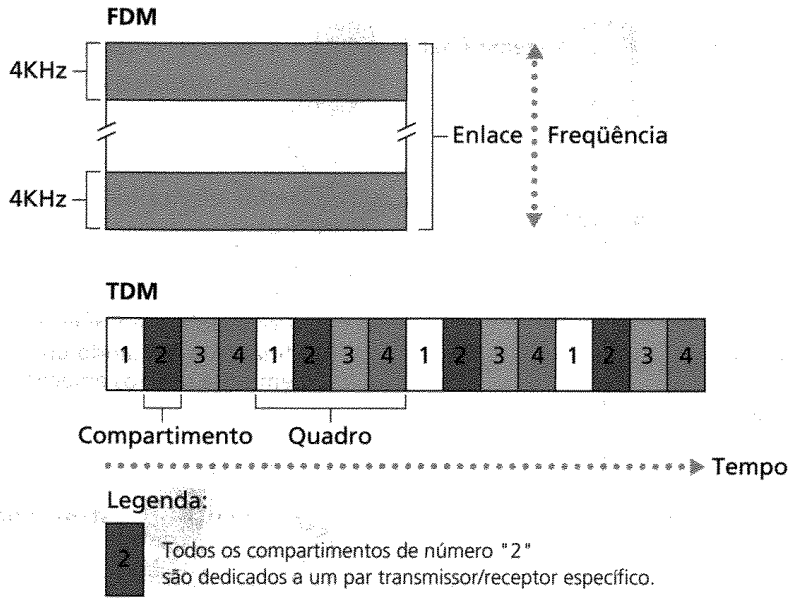


Figura 1.6 Com FDM, cada circuito dispõe continuamente de uma fração da largura de banda. Com TDM, cada circuito dispõe de toc a largura de banda periodicamente, durante breves intervalos de tempo (isto é, durante compartimentos de tempo)

Antes de encerrarmos esta discussão sobre comutação de circuitos, vamos examinar um exemplo num rico que deverá esclarecer melhor o assunto. Vamos considerar o tempo que levamos para enviar um arquiv de 640 kbits do sistema final A ao sistema final B por uma rede de comutação de circuitos. Suponha que tod os enlaces da rede usem TDM de 24 compartimentos e tenham uma taxa de 1,536 Mbps. Suponha tambér que um circuito fim-a-fim leva 500 milissegundos para ser ativado antes que A possa começar a transmit o arquivo. Em quanto tempo o arquivo será enviado? Cada circuito tem uma taxa de transmissão de (1,53 Mbps)/24 = 64 kbps; portanto, demorará (640 kbits)/64 kbps = 10 segundos para transmitir o arquivo. esses 10 segundos adicionamos o tempo de ativação do circuito, resultando 10,5 segundos para o envi Observe que o tempo de transmissão é independente do número de enlaces: o tempo de transmissão ser 10 segundos se o circuito fim-a-fim passasse por um ou por uma centena de enlaces. (O atraso real fim-a fim também inclui o atraso de propagação; ver Seção 1.6.)

Comutação de pacotes

Vimos na Seção 1.1 que aplicações trocam **mensagens** ao desempenhar sua tarefa. Mensagens poder conter qualquer característica que o projetista do protocolo queira. Podem desempenhar uma função d controle (por exemplo, as mensagens 'oi' no nosso exemplo de comunicação entre pessoas) ou podem cor ter dados, tal como uma mensagem de e-mail, uma imagem JPEG ou um arquivo de áudio MP3. Em rede de computadores modernas, o originador fragmenta mensagens longas em porções de dados menores denc minadas **pacotes**. Entre origem e destino, cada um desses pacotes percorre enlaces de comunicação e **comu tadores de pacotes** (há dois tipos principais de comutadores de pacotes: **roteadores** e **comutadores d camada de enlace**). Pacotes são transmitidos por cada enlace de comunicação a uma taxa igual à de trans missão *total* do enlace. A maioria dos comutadores de pacotes **armazena e reenvia** os pacotes nas entrada dos enlaces, numa técnica conhecida como armazena-e-reenvia e de acordo com a qual o comutador dev receber o pacote inteiro antes de poder começar a transmitir o primeiro bit do pacote para o enlace de saída Assim, comutadores de pacotes introduzem um atraso de armazenagem e reenvio na entrada de cada enla ce ao longo da rota do pacote. Esse atraso é proporcional ao comprimento do pacote em bits. Em particu lar, se um pacote consiste em L bits e deve ser reenviado por um enlace de saída de R bps, então o atraso d armazenagem e reenvio no comutador é L/R segundos.

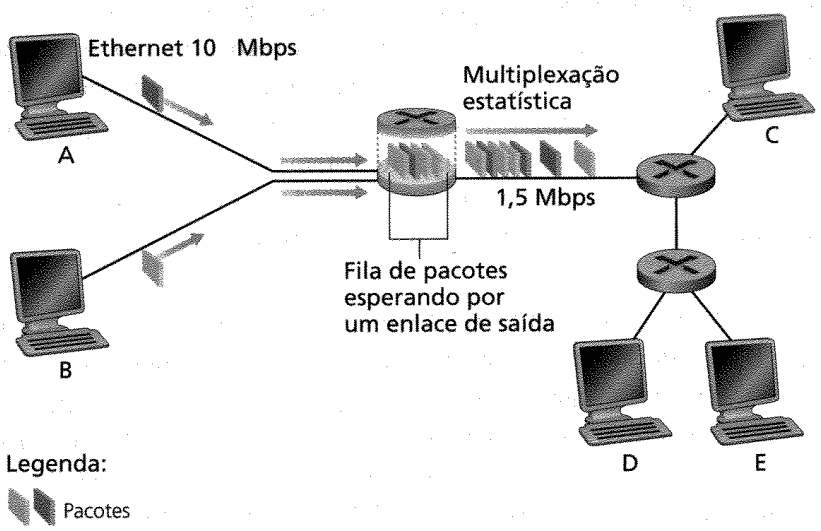


Figura 1.7 Comutação de pacotes

A cada comutador de pacotes estão ligados vários enlaces. Para cada um destes, o comutador de pacotes tem um **buffer de saída** (também denominado **fila de saída**), que armazena pacotes prestes a serem enviados pelo roteador para aquele enlace. Os buffers de saída desempenham um papel fundamental na comutação de pacotes. Se um pacote que está chegando precisa ser transmitido por um enlace, mas o encontra ocupado com a transmissão de outro pacote, deve aguardar no buffer de saída. Desse modo, além dos atrasos de armazenagem e reenvio, os pacotes sofrem **atrasos de fila** no buffer de saída. Esses atrasos são variáveis e dependem do grau de congestionamento da rede. Como o espaço do buffer é finito, um pacote que está chegando pode encontrá-lo completamente lotado de outros pacotes que estão esperando transmissão. Nesse caso, ocorrerá uma **perda de pacote** — um pacote que está chegando ou um dos que já estão na fila é descartado. Voltando à analogia do restaurante apresentada anteriormente nesta seção, o atraso de fila é análogo ao tempo gasto no bar do restaurante esperando uma mesa, enquanto a perda de pacote é análoga a ouvir do garçom que devemos desistir porque já há muitas pessoas ali.

A Figura 1.7 ilustra uma rede simples de comutação de pacotes. Nessa figura e nas subseqüentes, pacotes são representados por placas tridimensionais. A largura de uma placa representa o número de bits no pacote. Nessa figura todos os pacotes têm a mesma largura, portanto, o mesmo tamanho. Suponha que os sistemas finais A e B estejam enviando pacotes ao sistema final E. Os sistemas finais A e B primeiramente enviarão seus pacotes por enlaces Ethernet 10 Mbps até o primeiro comutador de pacotes, que vai direcioná-los para o enlace de 1,5 Mbps. Se a taxa de chegada de pacotes ao comutador for maior do que a taxa com que o comutador pode reenviar pacotes pelo enlace de saída de 1,5 Mbps, ocorrerá congestionamento, pois os pacotes formarão uma fila no buffer de saída do enlace antes de ser transmitidos para o enlace.

Vamos considerar agora quanto tempo demora para enviar um pacote de L bits de um sistema final para outro por uma rede de comutação de pacotes. Suponha que haja Q enlaces entre os dois sistemas finais, cada um com taxa de R bps. Admita que atrasos de fila e de propagação fim-a-fim sejam desprezíveis e que não haja estabelecimento de conexão. Primeiramente o pacote deve ser transmitido para o primeiro enlace que se origina do sistema final A, o que leva L/R segundos. Em seguida, ele tem de ser transmitido por cada um dos $Q - 1$ enlaces remanescentes, isto é, deve ser armazenado e reenviado $Q - 1$ vezes. Portanto, o atraso total é QL/R.

Comutação de pacotes *versus* comutação de circuitos: multiplexação estatística

Agora que já descrevemos comutação de pacotes e comutação de circuitos, vamos comparar as duas. Opositores da comutação de pacotes freqüentemente argumentam que ela não é adequada para serviços de

tempo real (por exemplo, ligações telefônicas e videoconferência) por causa de seus atrasos fim-a-fim variáveis e imprevisíveis (que se devem principalmente a atrasos de fila variáveis e imprevisíveis). Defensores da comutação de pacotes argumentam que (1) ela oferece melhor compartilhamento de largura de banda do que comutação de circuitos e (2) sua implementação é mais simples, mais eficiente e mais barata do que a implementação de comutação de circuitos. Uma discussão interessante sobre comutação de pacotes e comutação de circuitos pode ser encontrada em [Molinero-Fernandez, 2002]. De modo geral, quem não gosta de perder tempo fazendo reserva de mesa em restaurantes prefere comutação de pacotes à comutação de circuitos.

Por que a comutação de pacotes é mais eficiente? Vamos examinar um exemplo simples. Suponha que usuários compartilhem um enlace de 1 Mbps. Suponha também que cada usuário alterne períodos de atividade, quando gera dados a uma taxa constante de 100 kbps, e de inatividade, quando não gera dados. Suponha ainda que o usuário esteja ativo apenas 10 por cento do tempo (e fique ocioso, tomando cafezinho, durante os restantes 90 por cento). Com comutação de circuitos, devem ser *reservados* 100 kbps para *cada usuário* durante todo o tempo. Por exemplo, com TDM de comutação de circuitos, se um quadro de um segundo for dividido em 10 compartimentos de tempo de 100 milissegundos cada, então seria alocado um compartimento de tempo por quadro a cada usuário.

Desse modo, o enlace pode suportar somente 10 (= 1Mbps/100 kbps) usuários simultaneamente. Com comutação de pacotes, a probabilidade de haver um usuário específico ativo é 0,1 (isto é, 10 por cento). Se houver 35 usuários, a probabilidade de haver 11 ou mais usuários ativos simultaneamente é aproximadamente 0,0004. (O Problema 8 dos Exercícios de Fixação demonstra como essa probabilidade é calculada). Quando houver dez ou menos usuários ativos simultaneamente (a probabilidade de isso acontecer é 0,9996), a taxa agregada de chegada de dados é menor ou igual a 1 Mbps, que é a taxa de saída do enlace. Assim, quando houver dez ou menos usuários ativos, pacotes de usuários fluirão pelo enlace essencialmente sem atraso, como é o caso na comutação de circuitos. Quando houver mais de dez usuários ativos simultaneamente, a taxa agregada de chegada de pacotes excederá a capacidade de saída do enlace, e a fila de saída começará a crescer. (E continuará a crescer até que a velocidade agregada de entrada caia novamente para menos de 1 Mbps, ponto em que o comprimento da fila começará a diminuir). Como a probabilidade de haver mais de dez usuários ativos simultaneamente é ínfima nesse exemplo, a comutação de pacotes apresenta, essencialmente, o mesmo desempenho da comutação de circuitos, *mas o faz para mais de três vezes o número de usuários.*

Vamos considerar agora um segundo exemplo simples. Suponha que haja dez usuários e que um deles repentinamente gere mil pacotes de mil bits, enquanto os outros nove permanecem inativos e não geram pacotes. Com comutação de circuitos TDM de dez compartimentos de tempo por quadro, e cada quadro consistindo em mil bits, o usuário ativo poderá usar somente seu único compartimento por quadro para transmitir dados, enquanto os nove compartimentos restantes em cada quadro continuarão ociosos. Dez segundos se passarão antes que todo o milhão de bits de dados do usuário ativo seja transmitido. No caso da comutação de pacotes, o usuário ativo poderá enviá-los continuamente à taxa total de 1 Mbps, visto que não haverá outros usuários gerando pacotes que precisem ser multiplexados com os pacotes do usuário ativo. Nesse caso, todos os dados do usuário ativo serão transmitidos dentro de 1 segundo.

Os exemplos acima ilustram duas maneiras pelas quais o desempenho da comutação de pacotes pode ser superior à da comutação de circuitos. Também destacam a diferença crucial entre as duas formas de compartilhar a taxa de transmissão de um enlace entre várias correntes de bits. Comutação de circuitos aloca previamente a utilização do enlace de transmissão independentemente de demanda, com desperdício de tempo de enlace desnecessário alocado e não utilizado. Comutação de pacotes, por outro lado, aloca utilização de enlace *por demanda*. A capacidade de transmissão do enlace será compartilhada pacote por pacote somente entre usuários que tenham pacotes que precisem ser transmitidos pelo enlace. Tal compartilhamento de recursos por demanda (e não por alocação prévia) às vezes é denominado **multiplexação estatística** de recursos.

Embora tanto a comutação de pacotes quanto a comutação de circuitos predominem nas redes de telecomunicação de hoje, a tendência é, sem dúvida, a comutação de pacotes. Até mesmo muitas das atuais redes de telefonia de comutação de circuitos estão migrando lentamente para a comutação de pacotes. Em particular, redes telefônicas freqüentemente usam comutação de pacotes na parte cara de uma chamada telefônica para o exterior, isto é, na parte que não é processada em território nacional.

1.3.2 Redes de comutação de pacotes: redes de datagramas e redes de circuitos virtuais

Há duas grandes classes de redes de comutação de pacotes: redes de datagramas e redes de circuitos virtuais. A diferença entre elas é que seus comutadores usam endereços de destino ou números de circuitos virtuais para transmitir pacotes a seus destinos. Denominaremos **rede de datagramas** qualquer rede que transmita pacotes segundo endereços de sistemas finais de destino. Os roteadores na Internet transmitem pacotes segundo endereços dos sistemas finais de destino; portanto, a Internet é uma rede de datagramas. Denominaremos **rede de circuitos virtuais** qualquer rede que transmita pacotes segundo números de circuitos virtuais. Exemplos de tecnologias de comutação de pacotes que usam circuitos virtuais são redes X.25, frame relay e redes ATM (*asynchronous transfer mode* – modo de transferência assíncrono). Ainda que a diferença entre utilizar endereços de destino e números de circuitos virtuais pareça pouco importante, optar por um deles causa um grande impacto sobre o modo de montagem dos roteadores e sobre o gerenciamento do roteamento, como veremos adiante.

Redes de circuitos virtuais

Como o nome sugere, um **circuito virtual** (CV) pode ser imaginado como uma conexão virtual entre um sistema final de origem e um sistema final de destino. O importante é que estabelecer e manter esse CV envolverá não somente os dois sistemas finais, mas cada um dos comutadores ao longo do caminho do CV entre origem e destino. Um **identificador de circuito virtual** (*virtual-circuit identifier* — ID CV) será atribuído a um CV quando ele for estabelecido pela primeira vez entre a fonte e o destino. Qualquer pacote que faça parte do CV terá esse identificador em seu cabeçalho (ID CV). Cada comutador de pacotes tem uma tabela que mapeia ID CVs para enlaces de saída. Quando um pacote chega a um comutador de pacotes, este examina o ID CV, indexa sua tabela e transmite o pacote ao enlace de saída designado. Note que a origem e o destino de um CV são identificados apenas indiretamente pelo ID CV; os endereços reais dos sistemas finais de origem e destino não são necessários para executar a comutação. Isso significa que a comutação de pacotes pode ser realizada rapidamente (consultando um ID CV do pacote que chega na pequena tabela de tradução do CV, em vez de consultar um endereço de destino em um espaço de endereçamento potencialmente grande).

Como observamos anteriormente, um comutador em uma rede de CVs mantém **informação de estado** para suas conexões em curso. Especificamente, cada vez que uma nova conexão é ativada por um comutador, um novo registro de conexão deve ser adicionado à tabela de tradução dele; e cada vez que uma conexão é desativada, um registro deve ser removido da tabela. Mesmo que não haja nenhuma tradução de ID CVs, ainda assim é necessário manter informação de estado que associe números de CVs a números das interfaces de saída. A questão de um comutador de pacotes manter ou não informação de estado para cada conexão em curso é crucial — voltaremos a ela logo adiante.

Redes de datagramas

As redes de datagramas são análogas, em muitos aspectos, ao serviço postal. Quando um remetente envia uma carta a um destino, ele a coloca em um envelope e nele escreve o endereço do destinatário. Esse endereço tem uma estrutura hierárquica, por exemplo, país (caso seja enviada para o exterior), estado, cidade, logradouro e número. O correio usa o endereço do envelope para determinar a rota da carta até seu destino. Assim, se uma carta for enviada da França para um endereço em Filadélfia, nos Estados Unidos, então primeiramente os correios da França encaminham a carta a uma central dos correios nos Estados Unidos,

que, por sua vez, enviará a carta a uma central dos correios na Filadélfia. Por fim, um carteiro dessa cidade entregará a carta ao seu destino final.

Em uma rede de datagramas, cada pacote que transita por ela contém em seu cabeçalho o endereço de destino. Como acontece com os endereços postais, esse endereço tem uma estrutura hierárquica. Quando um pacote chega a um comutador de pacotes da rede, ele examina uma parte do endereço de destino do pacote e o remete a um comutador adjacente. Mais especificamente, cada comutador de pacotes tem uma tabela de transmissão que mapeia endereços de destino (ou partes deles) para um enlace de saída. Quando um pacote chega a um comutador, este analisa o endereço e indexa sua tabela com esse endereço de destino para descobrir qual é o enlace de saída adequado. Então, o comutador encaminha o pacote a esse enlace de saída.

O processo de roteamento fim-a-fim também é análogo a um motorista de carro que prefere pedir informações a usar um mapa. Por exemplo, suponha que Lucy queira ir da cidade A para a cidade B, no bairro N, rua X, nº 156. Primeiramente, ela vai até o posto de gasolina de seu bairro e pergunta como chegar até o nº 156 da rua X, no bairro N da cidade B. O frentista do posto extrai a parte do endereço que se refere à cidade B e informa que ela precisa pegar a rodovia A-B, cujo acesso é bem ao lado do posto de gasolina. Diz também que, assim que entrar na cidade B, ela deverá pedir mais informações a alguém de lá. Lucy então pega a rodovia A-B até chegar à cidade B, onde pede informações a outro frentista. Este identifica a parte do endereço que se refere ao bairro N e explica como faz para chegar à avenida principal dele, acrescentando que lá ela deverá pedir informações a alguém sobre a rua X. Chegando à avenida principal do bairro N, Lucy pára em uma banca de jornal e pergunta onde fica a rua X, nº 156. O jornaleiro extrai a parte da rua X e explica que é a quarta rua à esquerda. Assim que atinge a rua X, ela pergunta a uma criança de bicicleta onde é seu destino final. A criança identifica a parte do endereço que se refere ao nº 156 e indica a casa. Lucy então chega ao destino final.

Discutiremos detalhadamente neste livro a transmissão de pacotes em redes de datagramas. Mas, por enquanto, mencionamos que, ao contrário de redes de CVs, *redes de datagramas não mantêm informação de estado de conexão em seus comutadores*. Realmente, um comutador de pacotes em uma rede de datagramas pura fica completamente alheio a quaisquer fluxos de tráfego que possam estar passando por ele. Um comutador de pacotes em uma rede de datagramas toma decisões de transmissão com base no endereço de destino de um pacote e não segundo a conexão à qual o pacote pertence. Como as redes de circuito virtual têm de manter informação de estado de conexão — informação que deve ser instalada e removida conforme os circuitos vêem e vão, e limpa (removida) caso um CV termine anormalmente —, as redes de CVs requerem protocolos de manutenção de estado potencialmente complexos, que não são encontrados em redes de datagramas.

Você está realmente interessado em ver a rota fim-a-fim que esses pacotes tomam na Internet? Nós o convidamos a pôr a mão na massa e interagir com o programa Traceroute visitando o site `http://www.traceroute.org`. (Ver discussão do Traceroute na Seção 1.6.)

Taxonomia de rede

Já apresentamos vários conceitos importantes de rede: comutação de circuitos, comutação de pacotes, circuitos virtuais, serviços não orientados para conexão e serviços orientados para conexão. Como tudo isso se encaixa?

Primeiramente, em nossa visão simplista do mundo, uma rede de telecomunicações ou usa comutação de circuitos ou usa comutação de pacotes (ver Figura 1.8). Um enlace em uma rede de comutação de circuitos pode empregar ou FDM, ou TDM. Redes de comutação de pacotes ou são redes de circuitos virtuais, ou são redes de datagramas. Comutadores em redes de circuitos virtuais transmitem pacotes de acordo com seus números CVs e mantêm estado de conexão. Comutadores em redes de datagramas transmitem pacotes de acordo com seus endereços de destino e não mantêm estado de conexão.

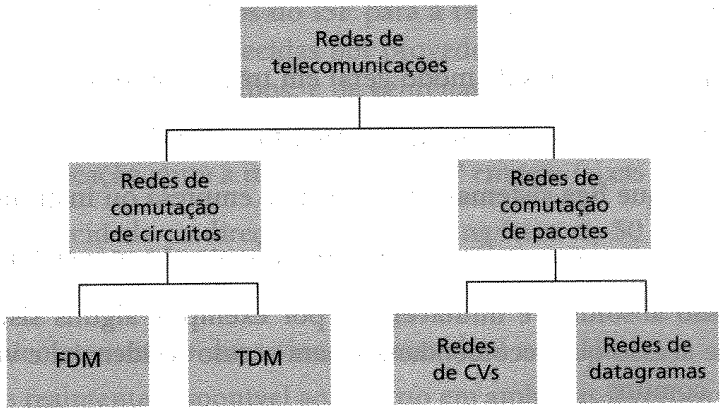


Figura 1.8 Taxonomia de redes de telecomunicações

1.4 Redes de acesso e meios físicos

Nas seções 1.2 e 1.3, examinamos os papéis que sistemas finais e comutadores de pacotes desempenham em uma rede de computadores. Nesta seção, consideramos **redes de acesso** — o enlace ou os enlaces físicos que conectam um sistema final a seu **roteador de borda**, que é o primeiro roteador de um caminho entre um sistema final e qualquer outro sistema final remoto. Assim, a rede de acesso provê a infra-estrutura para conectar o que denominamos instalações de clientes à infra-estrutura de rede. A Figura 1.9 mostra diversos tipos de enlaces de acesso de sistemas finais a roteador de borda; os enlaces de acesso são destacados por linhas cheias, sombreadas. Como a tecnologia de redes de acesso está estreitamente ligada à tecnologia de meios físicos (fibra, par coaxial, cabo telefônico de par trançado, espectro de rádio), consideramos esses dois tópicos em conjunto nesta seção.

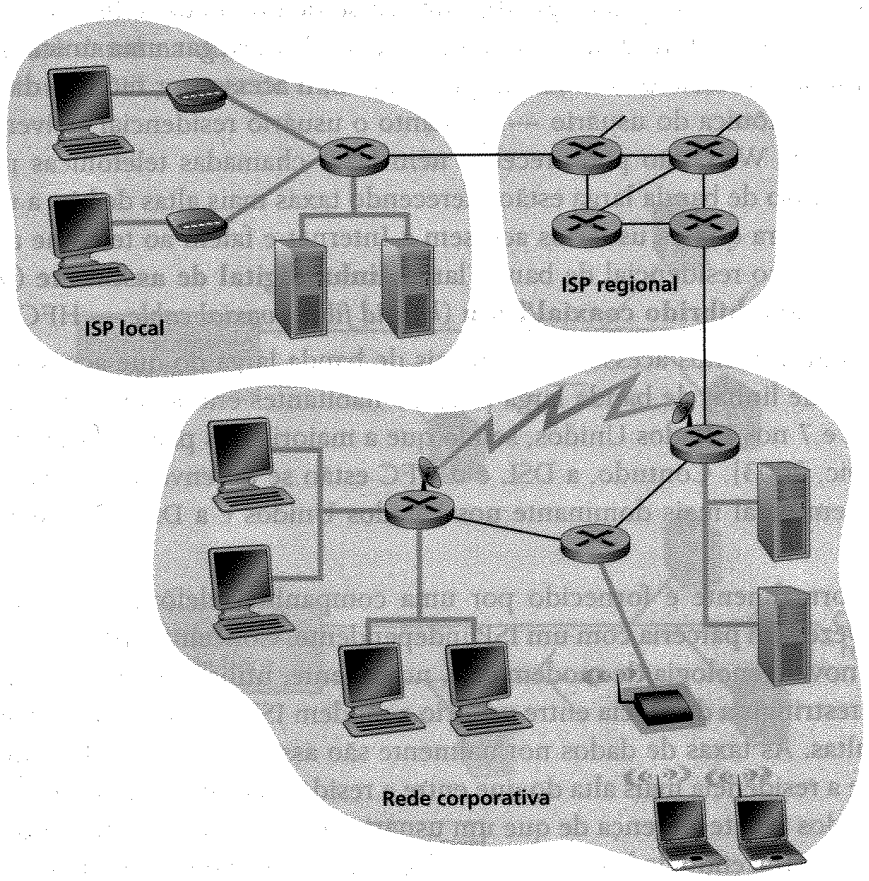


Figura 1.9 Redes de acesso

1.4.1 Redes de acesso

Redes de acesso podem ser divididas de modo geral em três categorias:

- **Acesso residencial**, que liga sistemas finais domésticos à rede.
- **Acesso corporativo**, que liga sistemas finais de uma empresa ou instituição educacional à rede.
- **Acesso sem fio**, que liga sistemas finais (que comumente são móveis) à rede.

Essas categorias não são absolutas e imutáveis — por exemplo, alguns sistemas finais corporativos podem usar a tecnologia de acesso que recomendamos para acesso residencial e vice-versa. As descrições a seguir valem para os casos comuns.

Acesso residencial

Acesso residencial refere-se à conexão de um sistema final residencial (normalmente um PC, mas cada vez mais uma rede doméstica, conforme você verá a seguir) a um roteador de borda. Uma forma comum de acesso residencial é o **modem discado** ligado por uma linha telefônica analógica a um ISP residencial (como America Online). O modem converte o sinal digital de saída do PC em formato analógico para transmissão pela linha telefônica analógica, que é um par de fios de cobre trançado e é a mesma linha telefônica utilizada para fazer chamadas telefônicas normais. (Discutiremos pares trançados mais adiante nesta seção). Na outra extremidade da linha telefônica analógica, um modem no ISP converte o sinal analógico novamente para sinal digital para entrar no roteador ISP. Assim, a rede de acesso é simplesmente um par de modens juntamente com uma linha telefônica ponto a ponto discada. As velocidades dos modens de hoje permitem acesso discado a taxas de até 56 kbps. Contudo, devido à baixa qualidade das linhas de par trançado existentes entre muitas residências e ISPs, a velocidade efetiva que muitos usuários conseguem é significativamente menor do que 56 kbps.

Muitos usuários residenciais acham os 50 kbps de acesso de um modem discado aflitivos de tão lentos. Por exemplo, são precisos aproximadamente 8 minutos para descarregar uma única música de 3 minutos por MP3 usando um modem discado de 56 kbps. Além disso, o acesso por modem discado impede a utilização normal da linha telefônica do usuário — enquanto o usuário residencial estiver usando um modem discado para navegar pela Web, não pode receber nem fazer chamadas telefônicas normais. Felizmente, novas tecnologias de acesso de banda larga estão oferecendo taxas mais altas de bits a usuários residenciais, além de fornecer meios para que os usuários acessem a Internet e falem ao telefone ao mesmo tempo. Há dois tipos comuns de acesso residencial de banda larga: **linha digital de assinante** (*digital subscriber line* — DSL) [DSL, 2004] e **cabo híbrido coaxial/fibra** (*hybrid fiber-coaxial cable* — HFC) [Cable Labs, 2004].

Até 2003, havia muito menos acessos residenciais de banda larga do que acessos por modem discado de 56 kbps. O número de linhas de banda larga por 100 habitantes era de aproximadamente 23 na Coréia do Sul, 13 no Canadá e 7 nos Estados Unidos, sendo que a maioria dos países europeus tinha menos de 10 por cento [Point Topic, 2003]. Contudo, a DSL e o HFC estão se desenvolvendo rapidamente por todo o mundo, com o HFC em geral mais dominante nos Estados Unidos e a DSL geralmente mais presente na Europa e na Ásia.

O acesso DSL normalmente é fornecido por uma companhia telefônica (por exemplo, Verizon ou France Telecom), às vezes em parceria com um ISP independente. Conceitualmente, similar aos modens discados, a DSL é uma nova tecnologia de modem que, novamente, utiliza linhas telefônicas de par trançado existentes. Mas, por restringir a distância entre usuário e modem ISP, a DSL pode transmitir e receber dados a taxas muito mais altas. As taxas de dados normalmente são assimétricas nas duas direções, sendo a taxa entre o roteador ISP e a residência mais alta do que entre a residência e o roteador ISP. A assimetria nas taxas de transferência de dados reflete a crença de que um usuário residencial será, mais provavelmente, um consumidor de informações (trazendo dados para si) do que um produtor de informações. Teoricamente, a DSL

pode oferecer taxas maiores do que 10 Mbps do ISP para a residência e mais do que 1 Mbps da residência para o ISP. Todavia, na prática, as taxas oferecidas por provedores DSL são muito menores. Até 2004, taxas típicas provedor-residência eram de 1 a 2 Mbps; e de centenas de kbps no sentido contrário.

A DSL usa multiplexação por divisão de freqüência, como descrito na seção anterior. Em particular, a DSL divide o enlace de comunicação entre a residência e o ISP em três faixas de freqüência sem sobreposição:

- um canal de alta velocidade na direção do usuário, na faixa de 50 kHz a 1 MHz;
- um canal de velocidade média na direção do provedor, na faixa de 4 kHz a 50 kHz;
- um canal telefônico comum de duas vias, na faixa de 0 a 4 kHz.

A taxa de transmissão realmente disponível ao usuário em ambas as direções é função da distância entre o modem doméstico e o modem do ISP, da bitola da linha de par trançado e do grau de interferência elétrica, entre outros aspectos. Diferentemente de modens discados, as DSLs foram explicitamente projetadas para distâncias curtas entre modens residenciais e modens de ISP, o que permite taxas de transmissão substancialmente mais altas do que as de acesso discado.

Enquanto DSL e modens discados usam linhas telefônicas comuns, redes de acesso HFC são extensões das redes de cabos existentes usadas para transmissão de TV a cabo. Em um sistema a cabo tradicional, um terminal (*headend*) da operadora transmite sinal para as residências por meio de uma rede de distribuição de cabo coaxial e amplificadores. Como ilustra a Figura 1.10, cabos de fibra ótica conectam o terminal aos entroncamentos (nós de fibra) mais próximos às residências, a partir dos quais são estendidos cabos coaxiais convencionais até as casas ou apartamentos individuais. Cada entroncamento (nó de fibra) normalmente suporta de 500 a 5 mil residências.

Como acontece com a DSL, o HFC requer modens especiais, denominados **modens a cabo**. Empresas que provêem acesso à Internet por cabo exigem que seus clientes comprem ou aluguem um modem. Normalmente, o modem a cabo é um equipamento externo que se conecta ao PC doméstico por uma porta Ethernet 10-BaseT. (Discutiremos Ethernet detalhadamente no Capítulo 5). Modens a cabo dividem a rede HFC em dois canais: um canal na direção do usuário (descida) e um canal na direção do provedor (subida). Como acontece com a DSL, a taxa de transmissão do canal de descida normalmente é maior do que a do canal de subida.

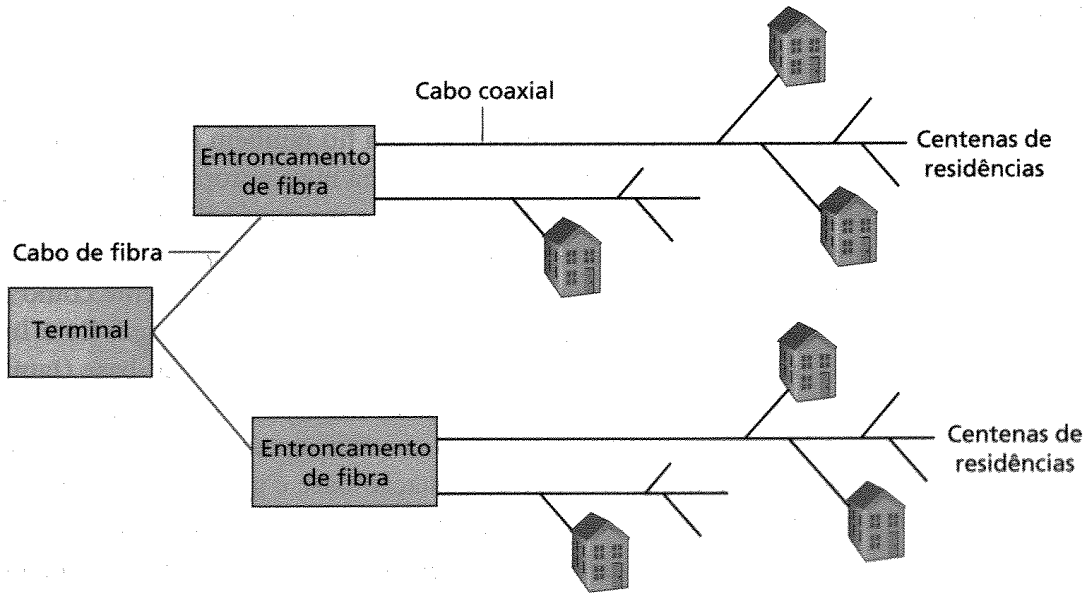


Figura 1.10 Uma rede de acesso de cabo híbrido de fibra-cabo coaxial

Uma característica importante da rede HFC é que ela é um meio de transmissão compartilhado. Em particular, cada pacote enviado pelo terminal provedor trafega por todos os enlaces até todas as residências e cada pacote enviado por uma residência viaja pelo canal de subida até o terminal provedor. Por essa razão, se vários usuários estiverem recebendo MP3 pelo canal de descida, a velocidade real com que cada usuário receberá seu arquivo será significativamente menor do que a velocidade nominal do canal. Por outro lado, se somente alguns usuários ativos estiverem navegando na Web, então cada um deles poderá receber páginas Web na velocidade de descida total, pois é raro que vários usuários solicitem uma página Web exatamente ao mesmo tempo. Como o canal de subida também é compartilhado, é preciso um protocolo de acesso múltiplo distribuído para coordenar transmissões e evitar colisões. (Discutiremos esse assunto da colisão mais detalhadamente quando analisarmos a Ethernet no Capítulo 5.) Os defensores da DSL se apressam em frisar que ela é uma conexão ponto a ponto entre a residência e o ISP e que, portanto, todas as larguras de banda da DSL são dedicadas, e não compartilhadas. Os defensores do cabo, no entanto, argumentam que uma rede HFC razoavelmente dimensionada provê taxas de transmissão mais altas do que a DSL. Não há dúvidas de que a guerra entre DSL e HFC pelo acesso residencial de alta velocidade já começou.

Um dos aspectos atraentes de DSL e HFC é que os serviços estão **sempre disponíveis:** isto é, o usuário pode ligar seu computador e ficar permanentemente conectado com um ISP e, ao mesmo tempo, fazer e receber chamadas telefônicas normais.

Acesso corporativo

Em empresas e *campi* universitários, normalmente é usada uma rede local (LAN) para ligar um sistema final ao roteador de borda. Como veremos no Capítulo 5, há muitos tipos de tecnologias de LAN. Contudo, hoje a tecnologia Ethernet é, de longe, a que predomina em redes corporativas. A Ethernet opera em 10 Mbps ou 100 Mbps (e agora até mesmo em 1 Gbps e 10 Gbps) e usa par de fios de cobre trançado ou cabo coaxial para conectar vários sistemas finais entre si e a um roteador de borda. O roteador de borda é responsável pelo roteamento de pacotes cujo destino é externo à LAN. Como o HFC, a Ethernet usa um meio compartilhado, de modo que usuários finais compartilham a velocidade de transmissão da LAN. Há pouco tempo, a tecnologia da Ethernet compartilhada começou a migrar em direção à tecnologia Ethernet comutada. A Ethernet comutada usa vários segmentos de par de fios de cobre trançado conectados a um 'comutador' que permite entregar a totalidade da velocidade de transmissão de uma Ethernet simultaneamente a diferentes usuários da mesma LAN. Examinaremos a Ethernet compartilhada e comutada mais detalhadamente no Capítulo 5.

Acesso sem fio

Acompanhando a revolução atual da Internet, a revolução sem fio também está causando um profundo impacto sobre o modo de vida e de trabalho das pessoas. Hoje, há mais europeus que possuem um telefone celular do que um PC ou um carro. A onda sem fio continua, e muitos analistas prevêem que equipamentos de mão sem fio (e muitas vezes móveis) — tais como telefones celulares e agendas eletrônicas (PDAs) — ultrapassarão a posição dominante que os computadores ocupam atualmente como dispositivos de acesso à Internet em todo o mundo. Hoje, há duas categorias amplas de acesso sem fio à Internet. Nas **LANs sem fio**, os usuários sem fio transmitem/recebem pacotes de/para uma estação-base (também conhecida como ponto de acesso sem fio) dentro de um raio de algumas dezenas de metros. A estação-base normalmente está ligada (por fio) à Internet, portanto, serve para conectar usuários sem fio a uma rede ligada por fio. Nas **redes sem fio de acesso de longa distância**, a estação-base é gerenciada por um provedor de telecomunicações e atende usuários dentro de um raio de dezenas de quilômetros.

Atualmente as LANs sem fio, baseadas em tecnologia IEEE 802.11 (também conhecida como Ethernet sem fio), estão se desenvolvendo rapidamente em departamentos universitários, escritórios comerciais, cafés e residências. As universidades em que ambos os autores trabalham instalaram, por exemplo, estações-base IEEE 802.11 em seus *campi*. Usando essa infra-estrutura de LAN sem fio, estudantes enviam e recebem e-mails ou navegam pela Web em qualquer lugar do *campus* (por exemplo, biblioteca, alojamento, sala

de aula ou um banco no jardim). A tecnologia 802.11, que discutiremos detalhadamente no Capítulo 6, provê uma velocidade de transmissão compartilhada de até 54 Mbps.

Hoje, muitas residências estão combinando acesso residencial de banda larga (isto é, modens a cabo ou DSL) com tecnologia barata de LAN sem fio para criar poderosas redes residenciais. A Figura 1.11 mostra um diagrama esquemático de uma rede residencial típica (na verdade, essa é a configuração da rede residencial de ambos os autores).

Essa rede residencial consiste em um laptop móvel e mais três PCs estacionários (dois ligados, um sem fio); uma estação-base (o ponto de acesso sem fio) que se comunica com o PC sem fio; um modem a cabo que provê acesso de banda larga à Internet; e um roteador, que interconecta a estação-base e o PC estacionário com o modem a cabo. Essa rede permite que os membros da família tenham acesso de banda larga à Internet, sendo que um deles é móvel e se desloca da cozinha para o quintal e para os quartos. O custo fixo total dessa rede é menos de 250 dólares (incluindo o modem a cabo/DSL).

Quando acessamos a Internet utilizando tecnologia de LAN sem fio, normalmente temos de estar a algumas dezenas de metros de uma estação-base, o que é viável para acesso em residências, em cafés e, de modo mais geral, em edifícios e nas suas vizinhanças. Mas, e se você estiver na praia ou dentro de um carro e precisar acessar a Internet? Para acesso em áreas tão distantes, usuários móveis da Internet usam a infra-estrutura de telefone portátil, acessando estações-base que estão a até dezenas de quilômetros de distância.

Há duas tecnologias que permitem acesso à Internet pela infra-estrutura de telefone portátil: o protocolo de acesso sem fio, versão 2 (*wireless access protocol* — WAP) [WAP, 2004], amplamente disponível na Europa, e a i-mode, muito usada no Japão. Parecidos com telefones sem fio comuns, mas com telas um pouco maiores, telefones WAP provêem acesso à Internet de baixa e média velocidade, bem como serviços de telefone portátil. Em vez de HTML, telefones WAP usam uma linguagem especial — a Linguagem de Markup WAP (WAP Markup Language — WML), que foi otimizada para telas pequenas e acessos de baixa velocidade. Na Europa, o protocolo WAP funciona sobre a muito bem-sucedida infra-estrutura de telefonia sem fio GMS, com o WAP 2.0 executando sobre uma pilha de protocolo TCP/IP. A tecnologia proprietária i-mode, semelhante ao WAP em conceito e funcionalidade, alcançou imenso sucesso no Japão.

Atualmente, empresas de telecomunicação estão fazendo enormes investimentos na Terceira Geração (3G) sem fio, que provê acesso à Internet em grandes áreas por meio da tecnologia de comutação de pacotes, com velocidades acima de 384 kbps [Kaaranen, 2001] [Korhonen, 2003]. Sistemas 3G provêem acesso de alta velocidade à Web e a vídeo interativo, e devem oferecer qualidade de voz melhor do que a oferecida por um telefone com fio.

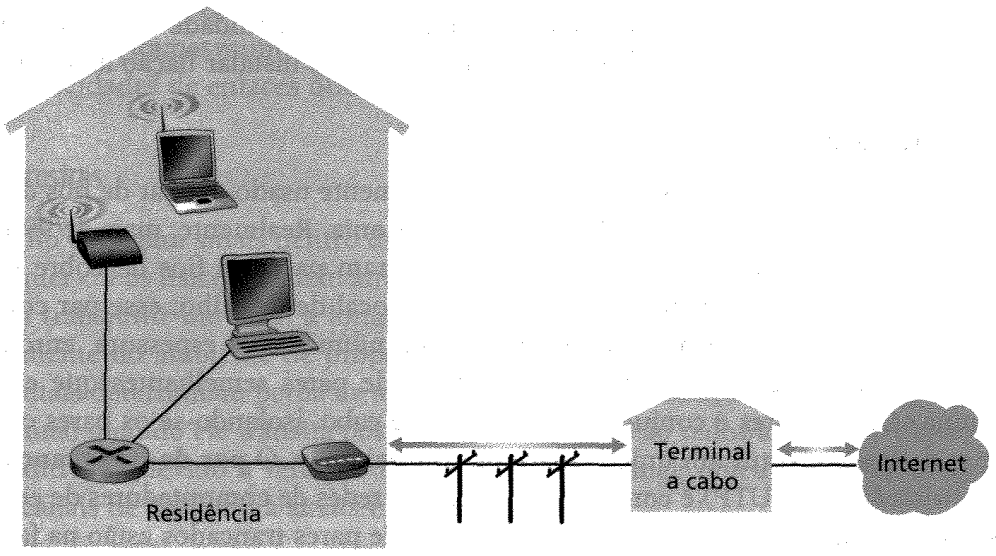


Figura 1.11 Um diagrama esquemático de uma rede residencial típica

Os primeiros sistemas 3G foram lançados no Japão. Os investimentos volumosos dedicados à tecnologia, infra-estrutura e licenças 3G levaram muitos analistas (e investidores!) a pensar seriamente se ela será o grande sucesso anunciado ou se perderá para tecnologias concorrentes como a IEEE 802.11. Ou, ainda, se as tecnologias 802.11 e 3G serão combinadas para prover acesso universal, porém heterogêneo. Ainda não se sabe o que vai acontecer. (Veja [Weinstein, 2002] e o item "História" na Seção 6.2). Abordaremos detalhadamente ambas as tecnologias 802.11 e 3G no Capítulo 6.

1.4.2 Meios físicos

Na subseção anterior, apresentamos uma visão geral de algumas das mais importantes tecnologias de acesso a redes na Internet. Ao descrever essas tecnologias, indicamos também os meios físicos utilizados por elas. Por exemplo, dissemos que o HFC usa uma combinação de cabo de fibra ótica e cabo coaxial e que modens discados de 56 kbps e ADSL usam par de fios de cobre trançado. Dissemos também que redes de acesso móveis usam o espectro de rádio. Nesta subseção damos uma visão geral desses e de outros meios de transmissão comumente empregados na Internet.

Para definir o que significa meio físico, vamos pensar na curta vida de um bit. Considere um bit saindo de um sistema final, transitando por uma série de enlaces e roteadores e chegando a outro sistema final. Esse pobre bit é transmitido muitas e muitas vezes. Primeiramente, o sistema final originador transmite o bit, e logo em seguida o primeiro roteador da série recebe-o; então, o primeiro roteador envia-o para o segundo roteador e assim por diante. Assim, nosso bit, ao viajar da origem ao destino, passa por uma série de pares transmissores–receptores, que o recebem por meio de ondas eletromagnéticas ou pulsos óticos que se propagam por um **meio físico**. Com muitos aspectos e formas possíveis, o meio físico não precisa ser obrigatoriamente do mesmo tipo para cada par transmissor–receptor ao longo do caminho. Alguns exemplos de meios físicos são: par de fios de cobre trançado, cabo coaxial, cabo de fibra ótica multiuso, espectro de rádio terrestre e espectro de rádio por satélite. Os meios físicos se enquadram em duas categorias: **meios guiados** e **meios não guiados**. Nos meios guiados, as ondas são dirigidas ao longo de um meio sólido, tal como um cabo de fibra ótica, um par de fios de cobre trançado ou um cabo coaxial. Nos meios não guiados, as ondas se propagam na atmosfera e no espaço, como é o caso de uma LAN sem fio ou de um canal digital de satélite.

Porém, antes de examinar as características dos vários tipos de meios, vamos discutir um pouco os valores. O custo real de um enlace físico (fio de cobre, cabo de fibra ótica e assim por diante) é em geral relativamente insignificante em comparação a outros custos da rede. Em particular, o custo da mão-de-obra de instalação do enlace físico pode ser várias vezes maior do que o do material. Por essa razão, muitos construtores instalam pares de fios trançados, fibra ótica e cabo coaxial em todas as salas de um edifício. Mesmo que apenas um dos meios seja usado inicialmente, há uma boa probabilidade de outro meio ser usado no futuro próximo — portanto, poupa-se dinheiro por não ser preciso instalar fiação adicional no futuro.

Par de fios de cobre trançado

O meio de transmissão guiado mais barato e mais comumente usado é o par de fios de cobre trançado, que vem sendo usado há mais de cem anos nas redes de telefonia. Realmente, mais de 99 por cento da fiação que conecta aparelhos telefônicos a centrais locais utilizam pares de fios de cobre trançados. Quase todos nós já vimos um par de fios trançado em casa ou no local de trabalho; esse par constituído de dois fios de cobre isolados, cada um com aproximadamente um milímetro de espessura, enrolados em espiral. Os fios são trançados para reduzir a interferência elétrica de pares semelhantes que estejam próximos. Normalmente, uma série de pares é conjugada dentro de um cabo, isolando-se os pares com blindagem de proteção. Um par de fios constitui um único enlace de comunicação. O **par de fios trançado sem blindagem** (*unshielded twisted pair* — UTP) é comumente usado em redes de computadores de edifícios, isto é, em LANs. Hoje, as taxas de transmissão de dados para as LANs de pares trançados estão na faixa de 10 Mbps a 1 Gbps. As taxas de transmissão de dados que podem ser alcançadas dependem da bitola do fio e da distância entre transmissor e receptor.

Quando a tecnologia da fibra ótica surgiu na década de 1980, muitos depreciaram o par trançado devido às suas taxas de transmissão de bits relativamente baixas. Alguns até acharam que a tecnologia da fibra ótica o substituiria completamente. Mas o par trançado não desistiu assim tão facilmente. Sua moderna tecnologia, tal como UTP categoria 5, pode alcançar taxas de transmissão de dados de 100 Mbps para distâncias de até algumas centenas de metros. E, em distâncias mais curtas, podem ser atingidas taxas ainda maiores. No final, o par trançado firmou-se como a solução dominante para LANs de alta velocidade.

Como vimos na seção sobre redes de acesso, o par trançado também é comumente usado para acesso residencial à Internet. Vimos que a tecnologia do modem discado habilita taxas de acesso de até 56 kbps com pares trançados. Vimos também que a tecnologia DSL (linha digital de assinante) habilitou usuários residenciais a acessar a Internet em velocidades maiores do que 6 Mbps com pares de fios trançados (quando as residências estão próximas a um modem de ISP).

Cabo coaxial

Como o par trançado, o cabo coaxial é constituído de dois condutores de cobre, porém concêntricos e não paralelos. Com essa configuração e isolamento e blindagem especiais, pode alcançar taxas altas de bits. Cabos coaxiais são muito comuns em sistemas de televisão a cabo. Como já comentamos, recentemente sistemas de televisão a cabo foram acoplados com modens a cabo para prover usuários residenciais de acesso à Internet a velocidades de 1 Mbps ou mais altas. Em televisão a cabo e acesso a cabo à Internet, o transmissor passa o sinal digital para uma banda de freqüência específica e o sinal analógico resultante é enviado do transmissor para um ou mais receptores. O cabo coaxial pode ser utilizado como um **meio compartilhado** guiado. Especificamente, vários sistemas finais podem ser conectados diretamente ao cabo, e todos eles recebem qualquer sinal que seja enviado pelos outros sistemas finais.

Fibras óticas

A fibra ótica é um meio delgado e flexível que conduz pulsos de luz, sendo que cada um desses pulsos representa um bit. Uma única fibra ótica pode suportar taxas de transmissão elevadíssimas, de até dezenas ou mesmo centenas de gigabits por segundo. Fibras óticas são imunes à interferência eletromagnética, têm baixíssima atenuação de sinal de até cem quilômetros e são muito difíceis de derivar. Essas características fizeram da fibra ótica o meio preferido para a transmissão guiada de grande alcance, em particular para cabos submarinos. Hoje, muitas redes telefônicas de longa distância dos Estados Unidos e de outros países usam exclusivamente fibras óticas, que também predominam no backbone da Internet. Contudo, o alto custo de equipamentos óticos — como transmissores, receptores e comutadores — vem impedindo sua utilização para transporte a curta distância, como em LANs ou em redes de acesso residenciais. [IEC Optical, 2003] [Goralski, 2001], [Ramaswami, 1998] e [Mukherjee, 1997] oferecem uma revisão de vários aspectos de redes óticas. Velocidades de enlaces óticos podem atingir dezenas de gigabits por segundo.

Canais de rádio terrestres

Canais de rádio carregam sinais dentro do espectro eletromagnético. São um meio atraente porque sua instalação não requer cabos físicos, podem atravessar paredes, dão conectividade ao usuário móvel e, potencialmente, podem transmitir um sinal a longas distâncias. As características de um canal de rádio dependem significativamente do ambiente de propagação e da distância pela qual o sinal deve ser transmitido. Condições ambientais determinam perda de sinal no caminho e atenuação por efeito de sombra (que reduz a intensidade do sinal quando ele transita por distâncias longas e ao redor/através de objetos interferentes), atenuação por multivias (devido à reflexão do sinal quando atinge objetos interferentes) e interferência (devido a outros canais de rádio ou a sinais eletromagnéticos).

Canais de rádio terrestres podem ser classificados, de modo geral, em dois grupos: os de pequeno alcance, que funcionam em locais próximos, normalmente abrangendo de dez a algumas centenas de metros, e os de longo alcance, que abrangem dezenas de quilômetros. As LANS sem fio descritas na Seção 1.4.1 uti-

lizam canais de rádio de pequeno alcance; WAP, i-mode e tecnologias 3G, também discutidas na Seção 1.4.1, usam canais de rádio de longo alcance. Para um inventário e uma discussão da tecnologia e produtos, veja [Dornan, 2001]. Discutiremos canais de rádio detalhadamente no Capítulo 6.

Canais de rádio por satélite

Um satélite de comunicação liga dois ou mais transmissores–receptores de microondas baseados na Terra, denominados estações terrestres. Ele recebe transmissões em uma faixa de freqüência, gera novamente o sinal usando um repetidor (sobre o qual falaremos a seguir) e o transmite em outra freqüência. Satélites podem prover taxas de transmissão na faixa de gigabits por segundo. Dois tipos de satélites são usados para comunicações: **satélites geoestacionários** e **satélites de baixa altitude**.

Os satélites geoestacionários ficam permanentemente sobre o mesmo lugar da Terra. Essa presença estacionária é conseguida colocando-se o satélite em órbita a 36 mil quilômetros acima da superfície terrestre. Essa enorme distância da estação terrestre ao satélite e de seu caminho de volta à estação terrestre traz um substancial atraso de propagação de sinal de 250 milissegundos. Mesmo assim, enlaces por satélite, que podem funcionar a velocidades de centenas de Mbps, são freqüentemente usados em redes de telefonia e no backbone da Internet.

Os satélites de baixa altitude são posicionados muito mais próximos da Terra e não ficam permanentemente sobre um único lugar. Eles giram ao redor da Terra exatamente como a Lua. Para prover cobertura contínua em determinada área, é preciso colocar muitos satélites em órbita. Hoje, existem muitos sistemas de comunicação de baixa altitude em desenvolvimento. A página Web referente à constelação de satélites da Lloyd [Wood, 1999] fornece e coleta informações sobre esses sistemas para comunicações. A tecnologia de satélites de baixa altitude poderá ser utilizada para acesso à Internet no futuro.

1.5 ISPs e backbones da Internet

Vimos anteriormente que sistemas finais (PCs de usuários, PDAs, servidores Web, servidores de correio eletrônico e assim por diante) conectam-se à Internet por meio de uma rede de acesso. Lembre-se de que a rede de acesso pode ser local com ou sem fio (por exemplo, em uma empresa, escola ou biblioteca) ou um ISP residencial (por exemplo, AOL ou MSN), alcançado por meio de modem discado, modem a cabo ou DSL. Mas conectar usuários finais e provedores de conteúdo a redes de acesso é apenas uma pequena peça do quebra-cabeça que é conectar as centenas de milhões de usuários e centenas de milhares de redes que compõem a Internet. A Internet é uma *rede de redes* — entender essa frase é a chave para resolver esse jogo.

Na Internet pública, redes de acessos situadas na borda da Internet são conectadas ao restante segundo uma hierarquia de níveis de ISPs, como mostra a Figura 1.12. Os ISPs de acesso (por exemplo, ISPs residenciais, como AOL, e ISPs corporativos que usam LANs) estão no nível mais baixo dessa hierarquia. No topo dela está um número relativamente pequeno de ISPs denominados **ISPs de nível 1**. Sob muitos aspectos, um ISP de nível 1 é igual a qualquer rede — tem enlaces e roteadores e está conectado a outras redes. Mas, considerando-se outros aspectos, ISPs de nível 1 são especiais. As velocidades de seus enlaces muitas vezes alcançam 622 Mbps ou mais, tendo os maiores deles enlaces na faixa de 2,5 a 10 Gbps. Conseqüentemente, seus roteadores são capazes de transmitir pacotes a taxas extremamente altas. ISPs de nível 1 também apresentam as seguintes características:

- conectam-se diretamente a *cada* um dos outros ISPs de nível 1,
- conectam-se a um grande número de ISPs de nível 2 e a outras redes clientes,
- têm cobertura internacional.

Esses ISPs também são conhecidos como redes de **backbone da Internet**. Citamos, como exemplos, Sprint, MCI (anteriormente UUNet/WorldCom), AT&T, Level3 (que adquiriu o Genuity), Qwest e Cable & Wireless. Em meados de 2002, a WorldCom era, de longe, o maior ISP de nível 1 existente — mais de duas vezes maior do que seu rival mais próximo, segundo diversas medições de tamanho [Teleography, 2002]. O interessante é que nenhum grupo sanciona, oficialmente, seu status de nível 1. Como se costuma dizer, se você tiver de perguntar se é um membro de um grupo, provavelmente não é.

Um ISP de nível 2 normalmente tem alcance regional ou nacional e (o que é importante) conecta-se apenas a uns poucos ISPs de nível 1 (Ver Figura 1.12).

Assim, para alcançar uma grande parcela da Internet global, um ISP de nível 2 tem de direcionar o tráfego por um dos ISPs de nível 1 com o qual está conectado. Um ISP de nível 2 é denominado um **cliente** do ISP de nível 1 com o qual está conectado, que, por sua vez, é denominado **provedor** de seu cliente. Muitas empresas de grande porte e instituições conectam suas redes corporativas diretamente a um provedor de nível 1 ou 2, tornando-se, desse modo, cliente daquele ISP. O provedor ISP cobra uma tarifa de seu cliente, que normalmente depende da taxa de transmissão do enlace que interliga ambos. Uma rede de nível 2 também pode preferir conectar-se diretamente a outras redes de mesmo nível, caso em que o tráfego pode fluir entre as duas sem ter de passar por uma rede de nível 1. Abaixo dos ISPs de nível 2 estão os de níveis mais baixos que se conectam à Internet por meio de um ou mais ISPs de nível 2 e, na parte mais baixa da hierarquia, estão os ISPs de acesso. Para complicar ainda mais as coisas, alguns provedores de nível 1 também são provedores de nível 2 (isto é, integrados verticalmente) e vendem acesso para Internet diretamente a usuários finais e provedores de conteúdo, bem como os ISPs de níveis mais baixos. Quando dois ISPs estão ligados diretamente um ao outro são denominados **pares** (*peers*) um do outro. Um estudo interessante [Subramanian, 2002] procura definir mais exatamente a estrutura em níveis da Internet estudando sua topologia em termos de relacionamentos cliente-provedor e entre parceiros (*peer-peer*).

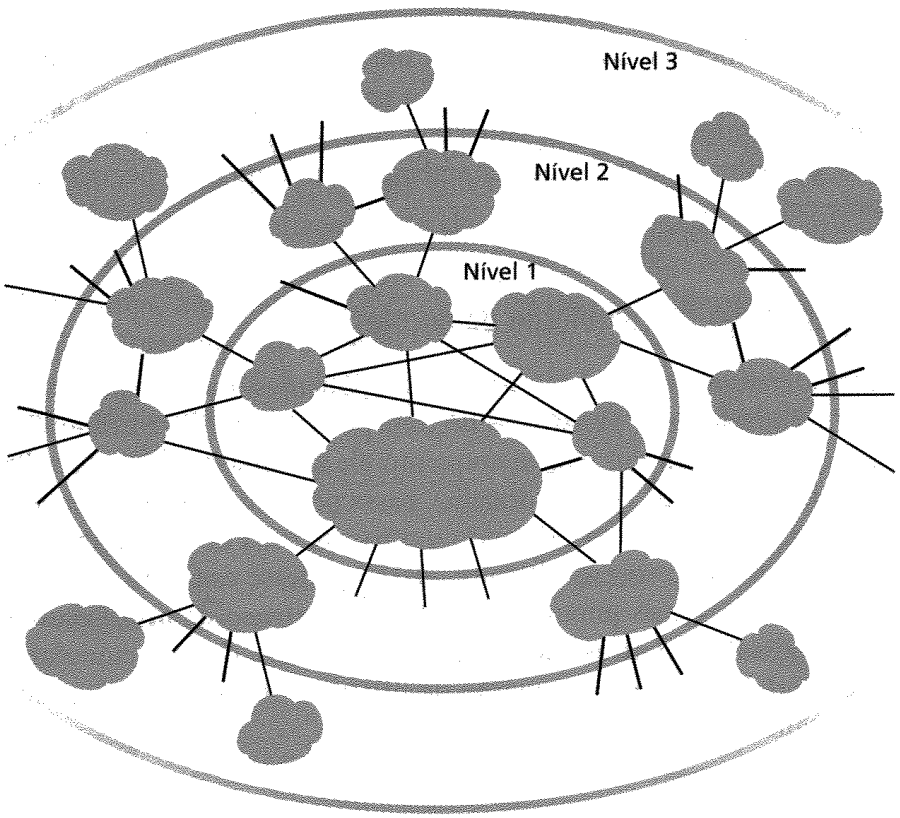


Figura 1.12 Interconexão de ISPs

Dentro da rede de um ISP, os pontos em que ele se conecta a outros ISPs (seja abaixo, acima ou n mesmo nível na hierarquia) são conhecidos como **pontos de presença** (*points of presence* — POPs). U POP é simplesmente um grupo de um ou mais roteadores na rede do ISP com os quais roteadores em outr ISPs, ou em redes pertencentes a clientes do ISP, podem se conectar. Um provedor de nível 1 normalmen tem muitos POPs espalhados por diferentes localidades geográficas em sua rede e várias redes clientes outros ISPs ligados a cada POP. Para se conectar ao POP de um provedor, uma rede cliente normalmen aluga um enlace de alta velocidade de um provedor de telecomunicações de terceiros e conecta um de se roteadores diretamente a um roteador no POP do provedor. Dois ISPs de nível 1 também podem formar u par conectando um par de POPs, cada um proveniente de um dos dois ISPs. Além disso, dois ISPs pode ter vários pontos de emparelhamento conectando-se um ao outro em dois ou mais de pares de POPs.

Além de se conectarem entre si em pontos privados de formação de par, ISPs muitas vezes se interc nectam em **Pontos de Acesso de Rede** (Network Access Points — NAPs), podendo cada um deles ser co trolado e operado por alguma empresa privada de telecomunicações ou por um provedor de backbone Internet. Os NAPs trocam enormes volumes de tráfego entre muitos ISPs. Contudo, é cada vez maior número de ISPs de nível 1 que evitam os NAPs e interconectam-se diretamente em pontos privados emparelhamento [Kende, 2000]. A tendência indica que ISPs de nível 1 farão interconexão entre si diret mente em pontos privados de emparelhamento e que ISPs de nível 2 farão interconexão com outros ISPs nível 2 e de nível 1 em NAPs. Como retransmitem e trocam volumes enormes de tráfego, os NAPs são, e si, redes de comutação complexas de alta velocidade, muitas vezes concentradas em um único local.

Resumindo, a topologia da Internet é complexa, consistindo em dezenas de ISPs de níveis 1 e 2 e milh res de ISPs de níveis mais baixos. A cobertura dos ISPs é bastante diversificada; alguns abrangem vários co tinentes e oceanos e outros se limitam a pequenas regiões do mundo. Os ISPs de níveis mais baixos cone tam-se a ISPs de níveis mais altos e estes (normalmente) se interconectam em pontos privados emparelhamento e NAPs. Usuários e provedores de conteúdo são clientes de ISPs de níveis mais baixos estes são clientes de ISPs de níveis mais altos.

Concluímos esta seção mencionando que qualquer um de nós pode se tornar um ISP de acesso tão log tenhamos uma conexão com a Internet. Basta comprar o equipamento necessário (por exemplo, roteador banco de modens) para que outros usuários se conectem conosco. Desse modo, novos níveis e ramificaçõ podem ser adicionados à topologia da Internet exatamente como uma nova pecinha pode ser encaixada e uma estrutura montada com Lego.

1.6 Atraso e perda em redes de comutação de pacotes

Agora que já consideramos resumidamente os componentes mais importantes da arquitetura Internet — aplicações, sistemas finais, protocolos de transporte fim-a-fim, roteadores e enlaces —, vam examinar o que pode acontecer com um pacote enquanto ele transita da origem ao destino. Lembre-se que um pacote começa em um sistema final (a origem), passa por uma série de roteadores e termina sua jo nada em um outro sistema final (o destino). Quando um pacote viaja de um nó (sistema final ou roteado ao nó subseqüente (sistema final ou roteador), sofre, ao longo desse caminho, diversos tipos de atraso e *cada* nó existente no caminho. Os mais importantes deles são o **atraso de processamento nodal**, o **atras de fila**, o **atraso de transmissão** e o **atraso de propagação**; juntos, eles se acumulam para formar o **atr so nodal total**. Para entender a fundo a comutação de pacotes e redes de computadores, é preciso ente der a natureza e a importância desses atrasos.

1.6.1 Tipos de atraso

Vamos examinar esses atrasos no contexto da Figura 1.13. Como parte de sua rota fim-a-fim entre or gem e destino, um pacote é enviado do nó anterior por meio do roteador A até o roteador B. Nossa meta caracterizar o atraso nodal no roteador A. Note que este tem um enlace de saída que leva ao roteador B. Ess

Dentro da rede de um ISP, os pontos em que ele se conecta a outros ISPs (seja abaixo, acima ou no mesmo nível na hierarquia) são conhecidos como **pontos de presença** (*points of presence* — POPs). Um POP é simplesmente um grupo de um ou mais roteadores na rede do ISP com os quais roteadores em outros ISPs, ou em redes pertencentes a clientes do ISP, podem se conectar. Um provedor de nível 1 normalmente tem muitos POPs espalhados por diferentes localidades geográficas em sua rede e várias redes clientes e outros ISPs ligados a cada POP. Para se conectar ao POP de um provedor, uma rede cliente normalmente aluga um enlace de alta velocidade de um provedor de telecomunicações de terceiros e conecta um de seus roteadores diretamente a um roteador no POP do provedor. Dois ISPs de nível 1 também podem formar um par conectando um par de POPs, cada um proveniente de um dos dois ISPs. Além disso, dois ISPs podem ter vários pontos de emparelhamento conectando-se um ao outro em dois ou mais de pares de POPs.

Além de se conectarem entre si em pontos privados de formação de par, ISPs muitas vezes se interconectam em **Pontos de Acesso de Rede** (Network Access Points — NAPs), podendo cada um deles ser controlado e operado por alguma empresa privada de telecomunicações ou por um provedor de backbone de Internet. Os NAPs trocam enormes volumes de tráfego entre muitos ISPs. Contudo, é cada vez maior o número de ISPs de nível 1 que evitam os NAPs e interconectam-se diretamente em pontos privados de emparelhamento [Kende, 2000]. A tendência indica que ISPs de nível 1 farão interconexão entre si diretamente em pontos privados de emparelhamento e que ISPs de nível 2 farão interconexão com outros ISPs de nível 2 e de nível 1 em NAPs. Como retransmitem e trocam volumes enormes de tráfego, os NAPs são, em si, redes de comutação complexas de alta velocidade, muitas vezes concentradas em um único local.

Resumindo, a topologia da Internet é complexa, consistindo em dezenas de ISPs de níveis 1 e 2 e milhares de ISPs de níveis mais baixos. A cobertura dos ISPs é bastante diversificada; alguns abrangem vários continentes e oceanos e outros se limitam a pequenas regiões do mundo. Os ISPs de níveis mais baixos conectam-se a ISPs de níveis mais altos e estes (normalmente) se interconectam em pontos privados de emparelhamento e NAPs. Usuários e provedores de conteúdo são clientes de ISPs de níveis mais baixos e estes são clientes de ISPs de níveis mais altos.

Concluímos esta seção mencionando que qualquer um de nós pode se tornar um ISP de acesso tão logo tenhamos uma conexão com a Internet. Basta comprar o equipamento necessário (por exemplo, roteador e banco de modens) para que outros usuários se conectem conosco. Desse modo, novos níveis e ramificações podem ser adicionados à topologia da Internet exatamente como uma nova pecinha pode ser encaixada em uma estrutura montada com Lego.

1.6 Atraso e perda em redes de comutação de pacotes

Agora que já consideramos resumidamente os componentes mais importantes da arquitetura da Internet — aplicações, sistemas finais, protocolos de transporte fim-a-fim, roteadores e enlaces —, vamos examinar o que pode acontecer com um pacote enquanto ele transita da origem ao destino. Lembre-se de que um pacote começa em um sistema final (a origem), passa por uma série de roteadores e termina sua jornada em um outro sistema final (o destino). Quando um pacote viaja de um nó (sistema final ou roteador) ao nó subseqüente (sistema final ou roteador), sofre, ao longo desse caminho, diversos tipos de atraso em *cada* nó existente no caminho. Os mais importantes deles são o **atraso de processamento nodal**, o **atraso de fila**, o **atraso de transmissão** e o **atraso de propagação**; juntos, eles se acumulam para formar o **atraso nodal total**. Para entender a fundo a comutação de pacotes e redes de computadores, é preciso entender a natureza e a importância desses atrasos.

1.6.1 Tipos de atraso

Vamos examinar esses atrasos no contexto da Figura 1.13. Como parte de sua rota fim-a-fim entre origem e destino, um pacote é enviado do nó anterior por meio do roteador A até o roteador B. Nossa meta é caracterizar o atraso nodal no roteador A. Note que este tem um enlace de saída que leva ao roteador B. Esse

enlace é precedido de uma fila (também conhecida como buffer). Quando o pacote chega ao roteador A, vindo do nó anterior, o roteador examina o cabeçalho do pacote para determinar o enlace de saída apropriado e então o direciona ao enlace. Nesse exemplo, o enlace de saída para o pacote é o que leva ao roteador B. Um pacote pode ser transmitido por um enlace somente se não houver nenhum outro pacote sendo transmitido por ele e se não houver outros pacotes à sua frente na fila. Se o enlace estiver ocupado, ou com pacotes à espera, o pacote recém-chegado entrará na fila.

Atraso de processamento

O tempo requerido para examinar o cabeçalho do pacote e determinar para onde direcioná-lo é parte do **atraso de processamento**, que pode também incluir outros fatores, como o tempo necessário para verificar os erros em bits existentes no pacote que ocorreram durante a transmissão dos bits desde o nó anterior ao roteador A. Atrasos de processamento em roteadores de alta velocidade normalmente são da ordem de microssegundos, ou menos. Depois desse processamento nodal, o roteador direciona o pacote à fila que precede o enlace com o roteador B. (No Capítulo 4, estudaremos os detalhes de operação de um roteador.)

Atraso de fila

O pacote sofre um **atraso de fila** enquanto espera para ser transmitido no enlace. O tamanho desse atraso para um pacote específico dependerá da quantidade de outros pacotes que chegarem antes e que já estiverem na fila esperando pela transmissão. Se a fila estiver vazia, e nenhum outro pacote estiver sendo transmitido naquele momento, então o tempo de atraso de fila de nosso pacote será zero. Por outro lado, se o tráfego estiver pesado e houver muitos pacotes também esperando para ser transmitidos, o atraso de fila será longo. Em breve, veremos que o número de pacotes que um determinado pacote provavelmente encontrará ao chegar é uma função da intensidade e da natureza do tráfego que está chegando à fila. Na prática, atrasos de fila podem ser da ordem de micro a milissegundos.

Atraso de transmissão

Admitindo-se que pacotes são transmitidos segundo a estratégia de "o primeiro a chegar será o primeiro a ser processado", como é comum em redes de comutação de pacotes, nosso pacote somente poderá ser transmitido depois que todos aqueles que chegaram antes tenham sido enviados. Denominemos o tamanho do pacote como L bits e a velocidade de transmissão do enlace do roteador A ao roteador B como R bits/s. A velocidade R é determinada pela velocidade de transmissão do enlace ao roteador B. Por exemplo, para um enlace Ethernet de 10 Mbps, a velocidade é $R = 10$ Mbps; para um enlace Ethernet de 100 Mbps, a velocidade é $R = 100$ Mbps. O **atraso de transmissão** (também denominado atraso de armazenamento e reenvio, como discutimos na Seção 1.3) é L/R. Esta é a quantidade de tempo requerida para empurrar (isto é, transmitir) todos os bits do pacote para o enlace. Na prática, atrasos de transmissão são comumente da ordem de micro a milissegundos.

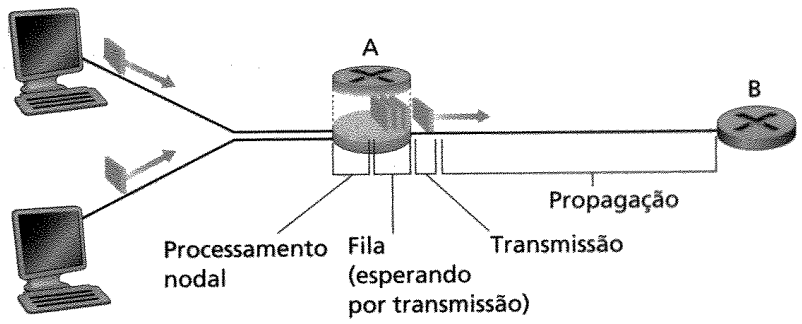


Figura 1.13 O atraso nodal no roteador A

Atraso de propagação

Assim que é lançado no enlace, um bit precisa se propagar até o roteador B. O tempo necessário para propagar o bit desde o início do enlace até o roteador B é o **atraso de propagação**. O bit se propaga à velocidade de propagação do enlace, a qual depende do meio físico do enlace (isto é, fibra ótica, par de fios de cobre trançado e assim por diante) e está na faixa de

$$2 \cdot 10^8 \text{ m/s a } 3 \cdot 10^8 \text{ m/s},$$

que é igual à velocidade da luz, ou um pouco menor. O atraso de propagação é a distância entre dois roteadores dividida pela velocidade de propagação. Isto é, o atraso de propagação é d/s, onde d é a distância entre o roteador A e o roteador B e s é a velocidade de propagação do enlace. Assim que o último bit do pacote se propagar até o nó B, ele e todos os outros bits precedentes do pacote serão armazenados no roteador B. Então, o processo inteiro continua, agora com o roteador B executando a retransmissão. Em redes WAN, os atrasos de propagação são da ordem de milissegundos.

Comparação entre atrasos de transmissão e de propagação

Os principiantes na área de redes de computadores às vezes têm dificuldade para entender a diferença entre atraso de transmissão e atraso de propagação. A diferença é sutil, mas importante. O atraso de transmissão é a quantidade de tempo requerida para o roteador empurrar o pacote para fora; é uma função do comprimento do pacote e da taxa de transmissão do enlace, mas nada tem a ver com a distância entre os dois roteadores. O atraso de propagação, por outro lado, é o tempo que leva para um bit se propagar de um roteador até o seguinte; é uma função da distância entre os dois roteadores, mas nada tem a ver com o comprimento do pacote ou com a taxa de transmissão do enlace.

Podemos esclarecer melhor as noções de atrasos de transmissão e de propagação com uma analogia. Considere uma rodovia que tenha um posto de pedágio a cada 100 quilômetros. Imagine que os trechos da rodovia entre os postos de pedágio sejam enlaces e que os postos de pedágio sejam roteadores. Suponha que os carros trafeguem (isto é, se propaguem) pela rodovia a uma velocidade de 100 km/h (isto é, quando o carro sai de um posto de pedágio, acelera instantaneamente até 100 km/h e mantém essa velocidade entre os dois postos de pedágio). Agora, suponha que dez carros viajem em comboio, um atrás do outro, em ordem fixa. Imagine que cada carro seja um bit e que o comboio seja um pacote. Suponha ainda que cada posto de pedágio libere (isto é, transmita) um carro a cada 12 segundos, que seja tarde da noite e que os carros do comboio sejam os únicos na estrada. Por fim, suponha que, ao chegar a um posto de pedágio, o primeiro carro do comboio aguarde na entrada até que os outros nove cheguem e formem uma fila atrás dele. (Assim, o comboio inteiro deve ser 'armazenado' no posto de pedágio antes de começar a ser 'reenviado'.) O tempo necessário para que todo o comboio passe pelo posto de pedágio e volte à estrada é de (10 carros)/(5 carros/minuto) = 2 minutos. Esse tempo é análogo ao atraso de transmissão em um roteador. O tempo necessário para um carro trafegar da saída de um posto de pedágio até o próximo posto de pedágio é de (100 km)/(100 km/h) = 1 hora. Esse tempo é análogo ao atraso de propagação. Portanto, o tempo decorrido entre o instante em que o comboio é 'armazenado' em frente a um posto de pedágio até o instante em que é 'armazenado' em frente ao seguinte é a soma do atraso de transmissão e do atraso de propagação — nesse exemplo, 62 minutos.

Vamos explorar um pouco mais essa analogia. O que aconteceria se o tempo de liberação do comboio no posto de pedágio fosse maior do que o tempo que um carro leva para trafegar entre dois postos? Por exemplo, suponha que os carros trafeguem a uma velocidade de 1.000 km/h e que o pedágio libere um carro por minuto. Então, o atraso de tráfego entre dois postos de pedágio é de 6 minutos e o tempo de liberação do comboio no posto de pedágio é de 10 minutos. Nesse caso, os primeiros carros do comboio chegarão ao segundo posto de pedágio antes que os últimos carros saiam do primeiro posto. Essa situação também acon-

tece em redes de comutação de pacotes — os primeiros bits de um pacote podem chegar a um roteador enquanto muitos dos remanescentes ainda estão esperando para ser transmitidos pelo roteador precedente.

Se d_{proc}, d_{fila}, d_{trans} e d_{prop} forem, respectivamente, os atrasos de processamento, de fila, de transmissão e de propagação, então o atraso nodal total é dado por:

$$d_{nodal} = d_{proc} + d_{fila} + d_{trans} + d_{prop}$$

A contribuição desses componentes do atraso pode variar significativamente. Por exemplo, d_{prop} pode ser desprezível (por exemplo, dois microssegundos) para um enlace que conecta dois roteadores no mesmo *campus* universitário; contudo, é de centenas de milissegundos para dois roteadores interconectados por um enlace de satélite geoestacionário e pode ser o termo dominante no d_{nodal}. De maneira semelhante, d_{trans} pode variar de desprezível a significativo. Sua contribuição normalmente é desprezível para velocidades de transmissão de 10 Mbps e mais altas (por exemplo, em LANs); contudo, pode ser de centenas de milissegundos para grandes pacotes de Internet enviados por enlaces de modens discados de baixa velocidade. O atraso de processamento, d_{proc}, é quase sempre desprezível; no entanto, tem forte influência sobre a produtividade máxima de um roteador, que é a velocidade máxima com que ele pode encaminhar pacotes.

1.6.2 Atraso de fila e perda de pacote

O mais complicado e interessante componente do atraso nodal é o atraso de fila, d_{fila}. Realmente, o atraso de fila é tão importante e interessante em redes de computadores que milhares de artigos e numerosos livros já foram escritos sobre ele [Bertsekas, 1991; Daigle, 1991; Kleinrock, 1975, 1976; Ross, 1995]. Neste livro, faremos apenas uma discussão intuitiva, de nível avançado, sobre o atraso de fila; o leitor mais curioso pode consultar alguns dos livros citados (ou até mesmo escrever uma tese sobre o assunto!). Diferentemente dos três outros atrasos (a saber, d_{proc}, d_{trans} e d_{prop}), o atraso de fila pode variar de pacote a pacote. Por exemplo, se dez pacotes chegarem a uma fila vazia ao mesmo tempo, o primeiro pacote transmitido não sofrerá nenhum atraso, ao passo que o último pacote sofrerá um atraso relativamente grande (enquanto espera que os outros nove pacotes sejam transmitidos). Por conseguinte, para se caracterizar um atraso de fila, normalmente são utilizadas medições estatísticas, tais como atraso de fila médio e variância do atraso de fila, e a probabilidade de ele exceder um valor especificado.

Quando o atraso de fila é grande ou insignificante? A resposta a essa pergunta depende da velocidade de transmissão do enlace, da taxa com que o tráfego chega à fila e de sua natureza, isto é, se chega intermitentemente, em rajadas. Para entendermos melhor, vamos adotar a para representar a taxa média com que os pacotes chegam à fila (a é medida em pacotes/segundo). Lembre-se de que R é a taxa de transmissão, isto é, a taxa (em bits/segundo) com que os bits são retirados da fila. Suponha também, para simplificar, que todos os pacotes tenham L bits. Então, a taxa média com que os bits chegam à fila é La bits/s. Por fim, imagine que a fila seja muito longa, de modo que, essencialmente, possa conter um número infinito de bits. A razão La/R, denominada **intensidade de tráfego**, freqüentemente desempenha um papel importante na estimativa do tamanho do atraso de fila. Se $La/R > 1$, então a velocidade média com que os bits chegam à fila excederá a velocidade com que eles podem ser transmitidos para fora da fila. Nessa situação desastrosa, a fila tenderá a aumentar sem limite e o atraso de fila tenderá ao infinito! Por conseguinte, uma das regras de ouro da engenharia de tráfego é: *projete seu sistema de modo que a intensidade de tráfego não seja maior do que 1.*

Agora, considere o caso em que La/R é $\leq$ a 1. Aqui, a natureza do tráfego influencia o atraso de fila. Por exemplo, se pacotes chegarem periodicamente — isto é, se chegar um pacote a cada L/R segundos — então todos os pacotes chegarão a uma fila vazia e não haverá atraso. Por outro lado, se pacotes chegarem em rajadas, mas periodicamente, poderá haver um significativo atraso de fila médio. Por exemplo, suponha que N pacotes cheguem ao mesmo tempo a cada $(L/R)N$ segundos. Então, o primeiro pacote transmitido não sofrerá atraso de fila; o segundo pacote transmitido terá um atraso de fila de L/R segundos e, de modo

mais geral, o enésimo pacote transmitido terá um atraso de fila de $(n - 1)L/R$ segundos. Deixamos como exercício para o leitor o cálculo do atraso de fila médio para esse exemplo.

Os dois exemplos de chegadas periódicas que acabamos de descrever são um tanto acadêmicos. Na realidade, o processo de chegada a uma fila é *aleatório* — isto é, não segue um padrão e os intervalos de tempo entre os pacotes são ao acaso. Nessa hipótese mais realista, a quantidade La/R normalmente não é suficiente para caracterizar por completo a estatística do atraso. Não obstante, é útil para entender intuitivamente a extensão do atraso de fila. Em particular, se a intensidade de tráfego for próxima de zero, então as chegadas de pacotes serão poucas e bem espaçadas e é improvável que um pacote que esteja chegando encontre outro na fila. Conseqüentemente, o atraso de fila médio será próximo de zero. Por outro lado, quando a intensidade de tráfego for próxima de 1, haverá intervalos de tempo em que a velocidade de chegada excederá a capacidade de transmissão (devido às chegadas em rajadas) e uma fila será formada. À medida que a intensidade de tráfego se aproxima de 1, o comprimento médio da fila fica cada vez maior. A dependência qualitativa entre o atraso de fila médio e a intensidade de tráfego é mostrada na Figura 1.14.

Um aspecto importante a observar na Figura 1.14 é que, quando a intensidade de tráfego se aproxima de 1, o atraso de fila médio aumenta rapidamente. Uma pequena porcentagem de aumento na intensidade resulta em um aumento muito maior no atraso, em termos de porcentagem. Talvez já tenha percebido esse fenômeno na estrada. Se você dirige regularmente por uma estrada que normalmente está congestionada, o fato de ela estar sempre assim significa que a intensidade de tráfego é próxima de 1. Se algum evento causar um tráfego ligeiramente maior do que o usual, as demoras que você sofrerá poderão ser enormes.

Perda de pacote

Na discussão anterior, admitimos que a fila é capaz de conter um número infinito de pacotes. Na realidade, a capacidade da fila que precede um enlace é finita, embora a sua formação dependa bastante do projeto e do custo do comutador. Como a capacidade da fila é finita, na verdade os atrasos de pacote não se aproximam do infinito quando a intensidade de tráfego se aproxima de 1. O que realmente acontece é que um pacote pode chegar e encontrar uma fila cheia. Sem espaço disponível para armazená-lo, o roteador **descartará** esse pacote; isto é, ele será **perdido**. Do ponto de vista de um sistema final, o pacote foi transmitido para o núcleo da rede, mas sem nunca ter emergido dele no destino. A fração de pacotes perdidos aumenta com o aumento da intensidade de tráfego. Por conseguinte, o desempenho em um nó é freqüentemente medido não apenas em termos de atraso, mas também em termos da probabilidade de perda de pacotes. Como discutiremos nos capítulos subseqüentes, um pacote perdido pode ser retransmitido fim-a-fim, seja pela aplicação, seja pelo protocolo de camada de transporte.

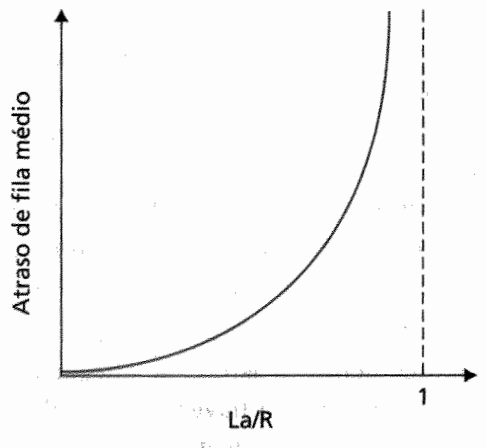


Figura 1.14 Dependência entre atraso de fila médio e intensidade de tráfego

Atraso fim-a-fim

Até o momento, nossa discussão focalizou o atraso nodal, isto é, o atraso em um único roteador. Concluiremos essa discussão considerando brevemente o atraso da origem ao destino. Para entender esse conceito, suponha que haja $N - 1$ roteadores entre a máquina de origem e a máquina de destino. Imagine também que a rede não esteja congestionada (e, portanto, os atrasos de fila sejam desprezíveis), que o atraso de processamento em cada roteador e na máquina de origem seja d_{proc}, que a taxa de transmissão de saída de cada roteador e da máquina de origem seja R bits/s e que o atraso de propagação em cada enlace seja d_{prop}. Os atrasos nodais se acumulam e resultam em um atraso fim-a-fim

$$d_{\text{fim-a-fim}} = N\,(d_{\text{proc}} + d_{\text{trans}} + d_{\text{prop}})$$

onde, mais uma vez, $d_{trans} = L/R$ e L é o tamanho do pacote. Convidamos você a generalizar essa fórmula para o caso de atrasos heterogêneos nos nós e para o caso de um atraso de fila médio em cada nó.

1.6.3 Atraso e rotas na Internet

Para perceber o que é realmente o atraso em uma rede de computadores, podemos utilizar o Traceroute, programa de diagnóstico que pode ser executado em qualquer máquina da Internet. Quando o usuário especifica um nome de hospedeiro de destino, o programa no hospedeiro de origem envia vários pacotes especiais em direção àquele destino. Ao seguir seu caminho até o destino, esses pacotes passam por uma série de roteadores. Um deles recebe um desses pacotes especiais e envia uma curta mensagem à origem. Essa mensagem contém o nome e o endereço do roteador.

Mais especificamente, suponha que haja $N - 1$ roteadores entre a origem e o destino. Então, a fonte enviará N pacotes especiais à rede e cada um deles estará endereçado ao destino final. Esses N pacotes especiais serão marcados de 1 a N, sendo a marca do primeiro pacote 1 e a do último, N. Assim que o *enésimo* roteador recebe o enésimo pacote com a marca N, não envia o pacote a seu destino, mas uma mensagem à origem. Quando o hospedeiro de destino recebe o pacote N, também envia uma mensagem à origem, que registra o tempo transcorrido entre o envio de um pacote e o recebimento da mensagem de retorno correspondente. A origem registra também o nome e o endereço do roteador (ou do hospedeiro de destino) que retorna a mensagem. Dessa maneira, a origem pode reconstruir a rota tomada pelos pacotes que vão da origem ao destino e pode determinar os atrasos de ida e volta para todos os roteadores intervenientes. Na prática, o programa Traceroute repete o processo que acabamos de descrever três vezes, de modo que a fonte envia, na verdade, $3 \cdot N$ pacotes ao destino. O RFC 1393 descreve detalhadamente o Traceroute.

Eis um exemplo de resultado do programa Traceroute, no qual a rota traçada ia do hospedeiro de origem gaia.cs.umass.edu (na Universidade de Massachusetts) até cis.poly.edu (na Polytechnic University no Brooklyn). O resultado tem seis colunas: a primeira coluna é o valor n descrito acima, isto é, o número do roteador ao longo da rota; a segunda coluna é o nome do roteador; a terceira coluna é o endereço do roteador (na forma xxx.xxx.xxx.xxx); as últimas três colunas são os atrasos de ida e volta para três tentativas. Se a fonte receber menos do que três mensagens de qualquer roteador determinado (devido à perda de pacotes na rede), o Traceroute coloca um asterisco logo após o número do roteador e registra menos do que três tempos de duração de viagens de ida e volta para aquele roteador.

```
1  cs-gw  (128.119.240.254)  1.009 ms  0.899  ms  0.993  ms
2  128.119.3.154  (128.119.3.154)  0.931  ms  0.441  ms  0.651  ms
3  border4-rt-gi-1-3.gw.umass.edu  (128.119.2.194)  1.032  ms  0.484  ms  0.451  ms
4  acr1-ge-2-1-0.Boston.cw.net  (208.172.51.129)  10.006  ms  8.150  ms  8.460  ms
5  agr4-loopback.NewYork.cw.net  (206.24.194.104)  12.272  ms  14.344  ms  13.267  ms
6  acr2-loopback.NewYork.cw.net  (206.24.194.62)  13.225  ms  12.292  ms  12.148  ms
```

```
7  pos10-2.core2.NewYork1.Level3.net  (209.244.160.133)  12.218  ms  11.823  ms  11.793  ms
8  gige9-1-52.hsipaccess1.NewYork1.Level3.net  (64.159.17.39)  13.081  ms  11.556  ms  13.297
9  p0-0.polyu.bbnplanet.net  (4.25.109.122)  12.716  ms  13.052  ms  12.786  ms
10  cis.poly.edu  (128.238.32.126)  14.080  ms  13.035  ms  12.802  ms
```

No exemplo anterior há nove roteadores entre a origem e o destino. Quase todos eles têm um nom todos têm endereço. Por exemplo, o nome do Roteador 3 é `border4-rt-gi-1-3.gw.umass.edu` e s endereço é `128.119.2.194`. Examinando os dados apresentados para esse roteador, verificamos que na p meira das três tentativas, o atraso de ida e volta entre a origem e o roteador foi de 1,03 milissegundos. atrasos para as duas tentativas subseqüentes foram 0,48 e 0,45 milissegundos e incluem todos os atra que acabamos de discutir, ou seja, atrasos de transmissão, de propagação, de processamento do roteado de fila. Como o atraso de fila varia com o tempo, o atraso de ida e volta do pacote n enviado a um rotead n na verdade pode ser maior do que o do pacote $n+1$ enviado ao roteador $n+1$. Realmente, observe exemplo que, aparentemente, os atrasos registrados para o Roteador 6 são maiores do que os atrasos veri cados no Roteador 7. Note que isso é um erro do processo de medição — qualquer pacote que vá para Roteador 7 deve necessariamente passar através do Roteador 6.

Você quer experimentar o Traceroute por conta própria? Recomendamos *muito* que visite o site `http:` `www.traceroute.org`, que provê uma interface Web para uma extensa lista de fontes para traçar rotas. Escol uma fonte, forneça o nome de hospedeiro para qualquer destino e o programa Traceroute fará todo o traball

1.7 Camadas de protocolo e seus modelos de serviço

Até aqui, nossa discussão demonstrou que a Internet é um sistema *extremamente* complicado e que pc sui muitos componentes: inúmeras aplicações e protocolos, vários tipos de sistemas finais e conexões ent eles, roteadores, além de vários tipos de meios físicos de enlace. Dada essa enorme complexidade, há alg ma esperança de organizar a arquitetura de rede ou, ao menos, nossa discussão sobre ela? Felizmente, a re posta a ambas as perguntas é sim.

1.7.1 Arquitetura de camadas

Antes de tentarmos organizar nosso raciocínio sobre a arquitetura da Internet, vamos procurar un analogia humana. Na verdade, lidamos com sistemas complexos o tempo todo em nosso dia-a-dia. Imagi se alguém pedisse que você descrevesse, por exemplo, o sistema de uma companhia aérea. Como vo encontraria a estrutura para descrever esse sistema complexo que tem agências de emissão de passagen pessoal para embarcar a bagagem para ficar no portão de embarque, pilotos, aviões, controle de tráfego aér e um sistema mundial de roteamento de aeronaves? Um modo de descrever esse sistema poderia ser apr sentar a relação de uma série de ações que você realiza (ou que outros realizam para você) quando voa p uma empresa aérea. Você compra a passagem, despacha suas malas, dirige-se ao portão de embarque e, fina mente, entra no avião, que decola e segue uma rota até seu destino. Após a aterrissagem, você desembar no portão designado e recupera suas malas. Se a viagem foi ruim, você reclama na agência que lhe vende a passagem (esforço em vão). Esse cenário é ilustrado na Figura 1.15.

Já podemos notar aqui algumas analogias com redes de computadores: você está sendo despachado d origem ao destino pela companhia aérea; um pacote é despachado da máquina de origem à máquina de de tino na Internet. Mas essa não é exatamente a analogia que buscamos. Estamos tentando encontrar algum *estrutura* na Figura 1.15. Observando essa figura, notamos que há uma função referente à passagem em cad ponta; há também uma função de bagagem para passageiros que já apresentaram a passagem e uma funçã de portão de embarque para passageiros que já apresentaram a passagem e despacharam a bagagem. Pa passageiros que já passaram pelo portão de embarque (isto é, aqueles que já apresentaram a passagem, de pacharam a bagagem e passaram pelo portão), há uma função de decolagem e de aterrissagem e, durante

```
7  pos10-2.core2.NewYork1.Level3.net  (209.244.160.133)  12.218  ms  11.823  ms  11.793  ms
8  gige9-1-52.hsipaccess1.NewYork1.Level3.net  (64.159.17.39)  13.081  ms  11.556  ms  13.297  ms
9  p0-0.polyu.bbnplanet.net  (4.25.109.122)  12.716  ms  13.052  ms  12.786  ms
10  cis.poly.edu  (128.238.32.126)  14.080  ms  13.035  ms  12.802  ms
```

No exemplo anterior há nove roteadores entre a origem e o destino. Quase todos eles têm um nome e todos têm endereço. Por exemplo, o nome do Roteador 3 é `border4-rt-gi-1-3.gw.umass.edu` e seu endereço é `128.119.2.194`. Examinando os dados apresentados para esse roteador, verificamos que na primeira das três tentativas, o atraso de ida e volta entre a origem e o roteador foi de 1,03 milissegundos. Os atrasos para as duas tentativas subseqüentes foram 0,48 e 0,45 milissegundos e incluem todos os atrasos que acabamos de discutir, ou seja, atrasos de transmissão, de propagação, de processamento do roteador e de fila. Como o atraso de fila varia com o tempo, o atraso de ida e volta do pacote n enviado a um roteador n na verdade pode ser maior do que o do pacote $n+1$ enviado ao roteador $n+1$. Realmente, observe no exemplo que, aparentemente, os atrasos registrados para o Roteador 6 são maiores do que os atrasos verificados no Roteador 7. Note que isso é um erro do processo de medição — qualquer pacote que vá para o Roteador 7 deve necessariamente passar através do Roteador 6.

Você quer experimentar o Traceroute por conta própria? Recomendamos *muito* que visite o site `http://www.traceroute.org`, que provê uma interface Web para uma extensa lista de fontes para traçar rotas. Escolha uma fonte, forneça o nome de hospedeiro para qualquer destino e o programa Traceroute fará todo o trabalho.

1.7 Camadas de protocolo e seus modelos de serviço

Até aqui, nossa discussão demonstrou que a Internet é um sistema *extremamente* complicado e que possui muitos componentes: inúmeras aplicações e protocolos, vários tipos de sistemas finais e conexões entre eles, roteadores, além de vários tipos de meios físicos de enlace. Dada essa enorme complexidade, há alguma esperança de organizar a arquitetura de rede ou, ao menos, nossa discussão sobre ela? Felizmente, a resposta a ambas as perguntas é sim.

1.7.1 Arquitetura de camadas

Antes de tentarmos organizar nosso raciocínio sobre a arquitetura da Internet, vamos procurar uma analogia humana. Na verdade, lidamos com sistemas complexos o tempo todo em nosso dia-a-dia. Imagine se alguém pedisse que você descrevesse, por exemplo, o sistema de uma companhia aérea. Como você encontraria a estrutura para descrever esse sistema complexo que tem agências de emissão de passagens, pessoal para embarcar a bagagem para ficar no portão de embarque, pilotos, aviões, controle de tráfego aéreo e um sistema mundial de roteamento de aeronaves? Um modo de descrever esse sistema poderia ser apresentar a relação de uma série de ações que você realiza (ou que outros realizam para você) quando voa por uma empresa aérea. Você compra a passagem, despacha suas malas, dirige-se ao portão de embarque e, finalmente, entra no avião, que decola e segue uma rota até seu destino. Após a aterrissagem, você desembarca no portão designado e recupera suas malas. Se a viagem foi ruim, você reclama na agência que lhe vendeu a passagem (esforço em vão). Esse cenário é ilustrado na Figura 1.15.

Já podemos notar aqui algumas analogias com redes de computadores: você está sendo despachado da origem ao destino pela companhia aérea; um pacote é despachado da máquina de origem à máquina de destino na Internet. Mas essa não é exatamente a analogia que buscamos. Estamos tentando encontrar alguma *estrutura* na Figura 1.15. Observando essa figura, notamos que há uma função referente à passagem em cada ponta; há também uma função de bagagem para passageiros que já apresentaram a passagem e uma função de portão de embarque para passageiros que já apresentaram a passagem e despacharam a bagagem. Para passageiros que já passaram pelo portão de embarque (isto é, aqueles que já apresentaram a passagem, despacharam a bagagem e passaram pelo portão), há uma função de decolagem e de aterrissagem e, durante o

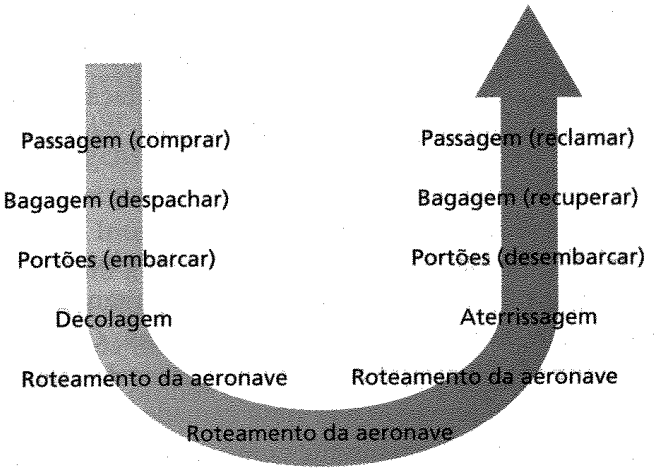


Figura 1.15 Uma viagem de avião: ações

vôo, uma função de roteamento do avião. Isso sugere que podemos examinar a funcionalidade na Figura 1.15 na *horizontal*, como mostra a Figura 1.16.

A Figura 1.16 dividiu a funcionalidade da linha aérea em camadas, provendo uma estrutura com a qual podemos discutir a viagem aérea. Note que cada camada, combinada com as camadas abaixo dela, implementa alguma funcionalidade, algum *serviço*. Na camada da passagem aérea e abaixo dela, é realizada a transferência 'balcão-de-linha-aérea–balcão-de-linha-aérea' de um passageiro. Na camada de bagagem e abaixo dela, é realizada a transferência 'despacho-de-bagagem–recuperação-de-bagagem' de um passageiro e de suas malas. Note que a camada da bagagem provê esse serviço apenas para a pessoa que já apresentou a passagem. Na camada do portão, é realizada a transferência 'portão-de-embarque–portão-de-desembarque' de um passageiro e de sua bagagem. Na camada de decolagem/aterrissagem, é realizada a transferência 'pista-a-pista' de passageiros e de suas bagagens. Cada camada provê seu serviço (1) realizando certas ações dentro da camada (por exemplo, na camada do portão, embarcar e desembarcar pessoas de um avião) e (2) utilizando os serviços da camada imediatamente inferior (por exemplo, na camada do portão, aproveitando o serviço de transferência 'pista-a-pista' de passageiros da camada de decolagem/aterrissagem).

Uma arquitetura de camadas nos permite discutir uma parcela específica e bem definida de um sistema grande e complexo. Essa simplificação tem considerável valor intrínseco, pois provê modularidade fazendo com que fique muito mais fácil modificar a implementação do serviço prestado pela camada. Contanto que a camada forneça o mesmo serviço para a que está acima dela e use os mesmos serviços da camada abaixo dela, o restante do sistema permanece inalterado quando a sua implementação é modificada. (Note que modificar a implementação de um serviço é muito diferente de mudar o serviço em si!) Por exemplo, se as funções de portão fossem modificadas (digamos que passassem a embarcar e desembarcar passageiros por ordem de altura), o restante do sistema da linha aérea permaneceria inalterado, já que a camada do portão

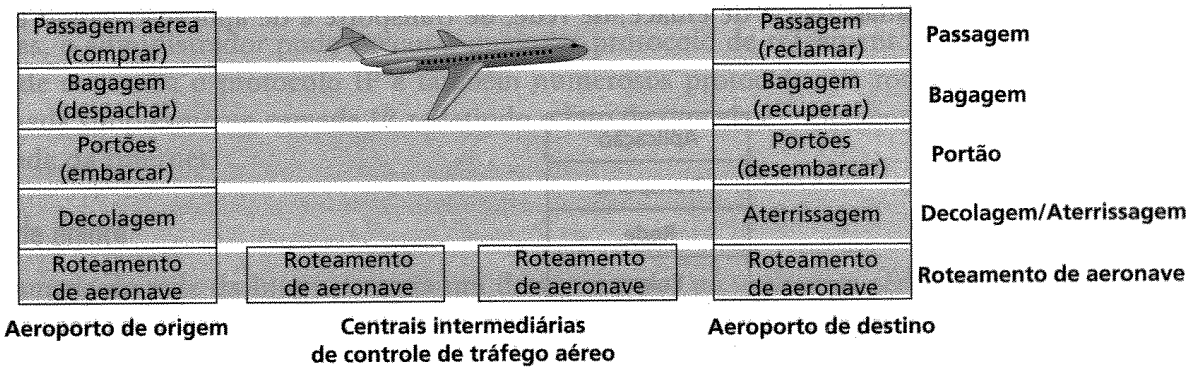


Figura 1.16 Camadas horizontais da funcionalidade de linha aérea

continuaria a prover a mesma função (embarcar e desembarcar passageiros); ela simplesmente implementaria aquela função de maneira diferente após a modificação. Para sistemas grandes e complexos que são atualizados constantemente, a capacidade de modificar a implementação de um serviço sem afetar outros componentes do sistema é outra vantagem importante da divisão em camadas.

Camadas de protocolo

Mas chega de linhas aéreas! Vamos agora voltar nossa atenção a protocolos de rede. Para prover uma estrutura para o projeto de protocolos de rede, projetistas de rede organizam protocolos — e o hardware e o software de rede que implementam os protocolos — em **camadas**. Cada protocolo pertence a uma das camadas, exatamente como cada função na arquitetura de linha aérea da Figura 1.16 pertencia a uma camada. Novamente estamos interessados nos **serviços** que uma camada oferece à camada acima dela — denominado **modelo de serviço** de uma camada. Exatamente como no nosso exemplo da linha aérea, cada camada provê seu serviço (1) executando certas ações dentro da camada e (2) utilizando os serviços da camada diretamente abaixo dela. Por exemplo, os serviços providos pela camada n podem incluir entrega confiável de mensagens de uma extremidade da rede à outra, que pode ser implementada utilizando um serviço não confiável de entrega de mensagem fim-a-fim da camada $n - 1$ e adicionando funcionalidade da camada n para detectar e retransmitir mensagens perdidas.

Uma camada de protocolo pode ser implementada em software, em hardware, ou em uma combinação dos dois. Protocolos de camada de aplicação como HTTP e SMTP — quase sempre são implementados em software em sistemas finais; o mesmo acontece com protocolos de camada de transporte. Como a camada física e as camadas de enlace de dados são responsáveis pelo manuseio da comunicação por um enlace específico, normalmente são implementadas em uma placa de interface de rede (por exemplo, placas de interface Ethernet ou Wi-Fi) associadas a um determinado enlace. A camada de rede quase sempre é uma implementação mista de hardware e software. Note também que, exatamente como as funções na arquitetura em camadas da linha aérea eram distribuídas entre os vários aeroportos e centrais de controle de tráfego aéreo que compunham o sistema, também um protocolo de camada n é distribuído entre os sistemas finais, comutadores de pacote e outros componentes que formam a rede. Isto é, há sempre uma parcela de um protocolo de camada n em cada um desses componentes de rede.

O sistema de camadas de protocolos tem vantagens conceituais e estruturais. Como vimos, a divisão em camadas proporciona um modo estruturado de discutir componentes de sistemas. A modularidade facilita a atualização de componentes de sistema. Devemos mencionar, no entanto, que alguns pesquisadores e engenheiros de rede se opõem veementemente ao sistema de camadas [Wakeman, 1992]. Uma desvantagem potencial desse sistema é que uma camada pode duplicar a funcionalidade de uma camada inferior. Por exemplo, muitas pilhas de protocolos oferecem serviço de recuperação de erros na camada de enlace e também fim-a-fim. Uma segunda desvantagem potencial é que a funcionalidade em uma camada pode necessitar de informações (por exemplo, um valor de carimbo de tempo) que estão presentes somente em uma outra camada, o que infringe o objetivo de separação de camadas.

Quando tomados em conjunto, os protocolos das várias camadas são denominados **pilha de protocolo**, que é formada por cinco camadas: física, de enlace, de rede, de transporte e de aplicação, como mostra a Figura 1.17.

Figura 1.17 A pilha de protocolo da Internet

Camada de aplicação

A camada de aplicação é onde residem aplicações de rede e seus protocolos. Ela inclui muitos protocolos, tais como o protocolo HTTP (que provê requisição e transferência de documentos pela Web), o SMTP (que provê transferência de mensagens de correio eletrônico) e o FTP (que provê a transferência de arquivos entre dois sistemas finais). Veremos que certas funções de rede, como a tradução de nomes fáceis de entender dados a sistemas finais da Internet (por exemplo, gaia.cs.umass.edu) para um endereço de rede de 32 bits, também são executadas com a ajuda de um protocolo de camada de aplicação, no caso, o sistema de nomes de domínio (*domain name system* — DNS). Veremos no Capítulo 2 que é muito fácil criar nossos próprios novos protocolos de camada de aplicação.

Lembre-se da nossa definição de protocolo na Seção 1.1, onde dissemos que as entidades distribuídas que implementam um protocolo trocam mensagens. Neste livro, essas mensagens serão denominadas mensagens de camada de aplicação.

Camada de transporte

A camada de transporte da Internet transporta mensagens da camada de aplicação entre os lados do cliente e servidor de uma aplicação. Há dois protocolos de transporte na Internet: TCP e UDP, e qualquer um deles pode levar mensagens de camada de aplicação. O TCP provê serviços orientados para conexão para suas aplicações. Alguns desses serviços são a entrega garantida de mensagens da camada de aplicação ao destino e controle de fluxo (isto é, compatibilização das velocidades remetente/receptor). O TCP também fragmenta mensagens longas em segmentos mais curtos e provê mecanismo de controle de congestionamento, de modo que uma origem regula sua velocidade de transmissão quando a rede está congestionada. O protocolo UDP provê serviço não orientado para conexão a suas aplicações. Como vimos na Seção 1.2, esse serviço é muito semelhante a um serviço sem supérfluos. Neste livro, um pacote de camada de transporte será denominado **segmento**.

Camada de rede

A camada de rede da Internet é responsável pela movimentação, de uma máquina para outra, de pacotes de camada de rede conhecidos como *datagramas*. O protocolo de camada de transporte da Internet (TCP ou UDP) em uma máquina de origem passa um segmento de camada de transporte e um endereço de destino à camada de rede, exatamente como você passaria ao serviço de correios uma carta com um endereço de destinatário. A camada de rede então provê o serviço de entrega do segmento à camada de transporte na máquina destinatária.

A camada de rede da Internet tem dois componentes principais. Um deles é um protocolo que define os campos no datagrama, bem como o modo como os sistemas finais e os roteadores agem nesses campos. Este é o famoso protocolo IP. Existe somente um único protocolo IP, e todos os componentes da Internet que têm uma camada de rede devem executar esse protocolo. O outro componente importante é o protocolo de roteamento que determina as rotas que os datagramas seguem entre origens e destinos. A Internet tem muitos protocolos de roteamento. Como vimos na Seção 1.5, a Internet é uma rede de redes e, dentro de uma delas, o administrador pode executar qualquer protocolo de roteamento que queira. Embora a camada de rede contenha o protocolo IP e também numerosos protocolos de roteamento, ela quase sempre é denominada simplesmente camada IP, refletindo o fato de que ela é o elemento fundamental que mantém a integridade da Internet.

Camada de enlace

A camada de rede da Internet roteia um datagrama por meio de uma série de comutadores de pacotes (denominados *roteadores* na Internet) entre a origem e o destino. Para levar um pacote de um nó (sistema final ou comutador de pacotes) ao nó seguinte na rota, a camada de rede depende dos serviços da camada de

enlace. Em particular, em cada nó, a camada de rede passa o datagrama para a camada de enlace, que o entrega, ao longo da rota, ao nó seguinte, no qual o datagrama é passado da camada de enlace para a de rede.

Os serviços prestados pela camada de enlace dependem do protocolo específico empregado no enlace. Alguns protocolos provêem entrega garantida entre enlaces, isto é, desde o nó transmissor, passando por um único enlace, até o nó receptor. Note que esse serviço confiável de entrega é diferente do serviço de entrega garantida do TCP, que provê serviço de entrega garantida de um sistema final a outro. Exemplos de protocolos de camadas de enlace são Ethernet e PPP (*point-to-point protocol* — protocolo ponto-a-ponto). Como datagramas normalmente precisam transitar por diversos enlaces para irem da origem ao destino, serão manuseados por diferentes protocolos de camada de enlace em diferentes enlaces ao longo de sua rota, podendo ser manuseados por Ethernet em um enlace e por PPP no seguinte. A camada de rede receberá um serviço diferente de cada um dos variados protocolos de camada de enlace. Neste livro, pacotes de camada de enlace serão denominados **quadros**.

Camada física

Enquanto a tarefa da camada de enlace é movimentar quadros inteiros de um elemento da rede até um elemento adjacente, a da camada física é movimentar os *bits individuais* que estão dentro do quadro de um nó para o seguinte. Os protocolos nessa camada novamente dependem do enlace e, além disso, dependem do próprio meio de transmissão do enlace (por par de fios de cobre trançado ou fibra ótica monomodal). Por exemplo, a Ethernet tem muitos protocolos de camada física: um para par de fios de cobre trançado, outro para cabo coaxial, um outro para fibra e assim por diante. Em cada caso, o bit é movimentado pelo enlace de um modo diferente.

Examinando o sumário deste livro, você verá que o organizamos segundo uma estrutura semelhante à das camadas da pilha do protocolo da Internet. Adotamos uma abordagem ***top-down* (de cima para baixo)**, começando com a camada de aplicação e prosseguindo para as camadas de baixo.

1.7.2 Camadas, mensagens, segmentos, datagramas e quadros

A Figura 1.18 apresenta o caminho físico que os dados percorrem: para baixo na pilha de protocolos de um sistema final emissor, para cima e para baixo nas pilhas de protocolos de um comutador de camada de enlace interveniente e de um roteador e então para cima na pilha de protocolos do sistema final receptor. Como discutiremos mais adiante neste livro, ambos, comutadores de camada de enlace e roteadores, são comutadores de pacotes. De modo semelhante a sistemas finais, roteadores e comutadores de camada de enlace organizam seu hardware e software de rede em camadas. Mas roteadores e comutadores de camada de enlace não implementam *todas* as camadas da pilha de protocolos; normalmente implementam apenas as camadas de baixo. Como mostra a Figura 1.18, comutadores de camada de enlace implementam as camadas 1 e 2; roteadores implementam as camadas 1, 2 e 3. Isso significa, por exemplo, que roteadores da Internet são capazes de implementar o protocolo IP (da camada 3), mas comutadores de camada de enlace não. Veremos mais adiante que, embora não reconheçam endereços IP, comutadores de camada de enlace são capazes de reconhecer endereços de camada 2, tais como endereços da Ethernet. Note que sistemas finais implementam todas as cinco camadas, o que é consistente com a noção de que a arquitetura da Internet concentra sua complexidade na periferia da rede.

A Figura 1.18 também ilustra o importante conceito de **encapsulamento**. Uma **mensagem de camada de aplicação** na máquina emissora (M na Figura 1.18) é passada para a camada de transporte. No caso mais simples, esta pega a mensagem e anexa informações adicionais (denominadas informações de cabeçalho de camada de transporte, H_t na Figura 1.18) que serão usadas pela camada de transporte do lado receptor. A mensagem de camada de aplicação e as informações de cabeçalho da camada de transporte, juntas, constituem o **segmento de camada de transporte**, que encapsula a mensagem de camada de aplicação. As informações adicionadas podem incluir dados que habilitem a camada de transporte do lado do receptor a

entregar a mensagem à aplicação apropriada, além de bits de detecção de erro que permitem que o receptor determine se os bits da mensagem foram modificados em trânsito. A camada de transporte então passa o segmento à camada de rede, que adiciona informações de cabeçalho de camada de rede (H_n na Figura 1.18), como endereços de sistemas finais de origem e de destino, criando um **datagrama de camada de rede**. Este é então passado para a camada de enlace, que (é claro!), adicionará suas próprias informações de cabeçalho e criará um **quadro de camada de enlace.**

Uma analogia útil que podemos usar aqui é o envio pelo correio de um memorando entre escritórios de um empresa. O memorando representa a mensagem da camada de aplicação e é colocado em um envelope de correspondência interna em cuja face são escritos o nome e o departamento do destinatário. Essas informações ajudarão a central de distribuição de correspondência do escritório receptor a dirigir o memorando à pessoa certa. O envelope, que contém informações de cabeçalho (o nome do destinatário e de seu departamento) e encapsula a mensagem de camada de aplicação (o memorando) é análogo ao segmento de camada de transporte. A central de correspondência do escritório emissor coloca o memorando dentro de um envelope adequado para envio pelo correio, escreve os endereços postais dos escritórios remetente e destinatário e adiciona um selo. Neste ponto, o envelope postal é análogo ao datagrama — encapsula o segmento de camada de transporte (o envelope do memorando e seu conteúdo), que por sua vez encapsula a mensagem original (o memorando). O correio entrega o envelope postal à central de correspondência do escritório destinatário, onde o envelope é então aberto e é retirado o envelope de correspondência interna. Este é encaminhado para a pessoa correta, que o abre e retira o memorando.

O processo de encapsulamento pode ser mais complexo do que o descrito acima. Por exemplo, uma mensagem grande pode ser dividida em vários segmentos de camada de transporte (que também podem ser divididos em vários datagramas de camada de rede). Na extremidade receptora, cada segmento deve ser reconstruído a partir dos datagramas que o compõem.

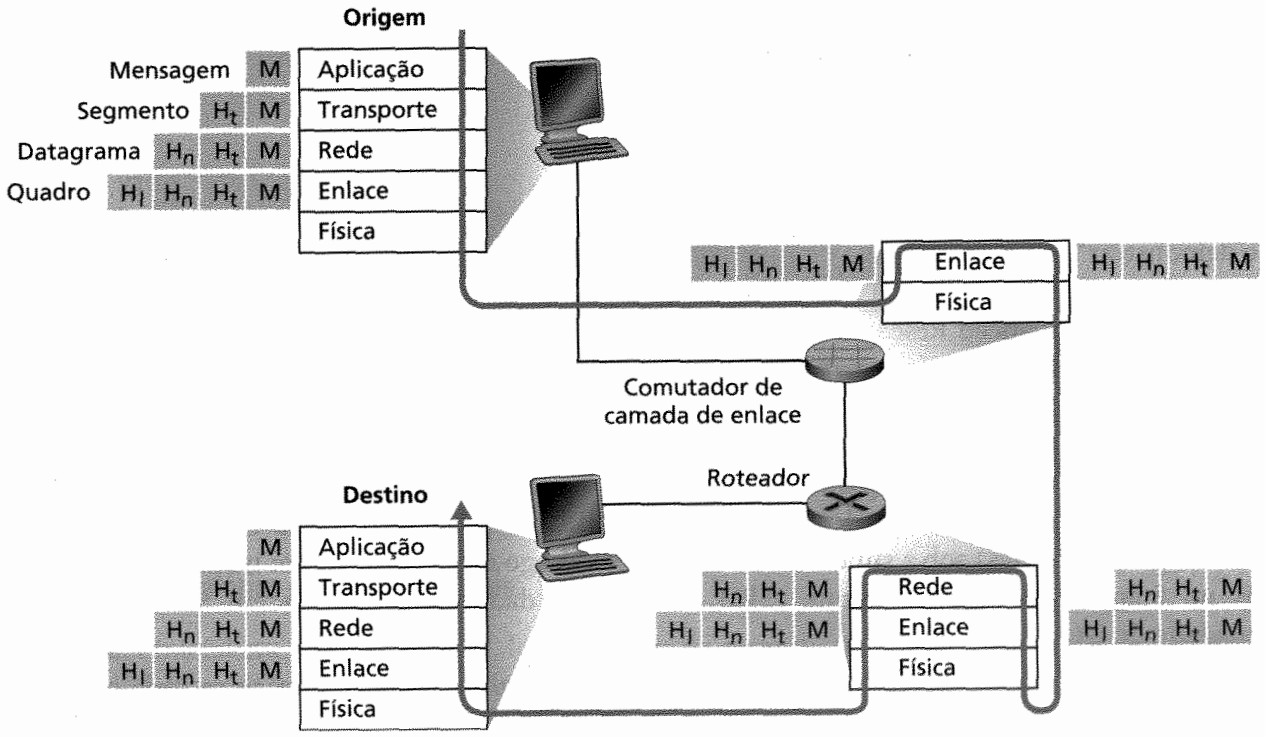


Figura 1.18 Hospedeiros, roteadores e comutadores de camada de enlace; cada um contém um conjunto diferente de camadas, refletindo suas diferenças em funcionalidade

1.8 História das redes de computadores e da Internet

Da seção 1.1 à 1.7, apresentamos um panorama da tecnologia de redes de computadores e da Internet. Agora, você já deve saber o suficiente para impressionar sua família e amigos. Contudo, se realmente quiser ser o maior sucesso do próximo coquetel, você deve rechear seu discurso com pérolas da fascinante história da Internet [Segaller, 1998].

1.8.1 Desenvolvimento da comutação de pacotes: 1961-1972

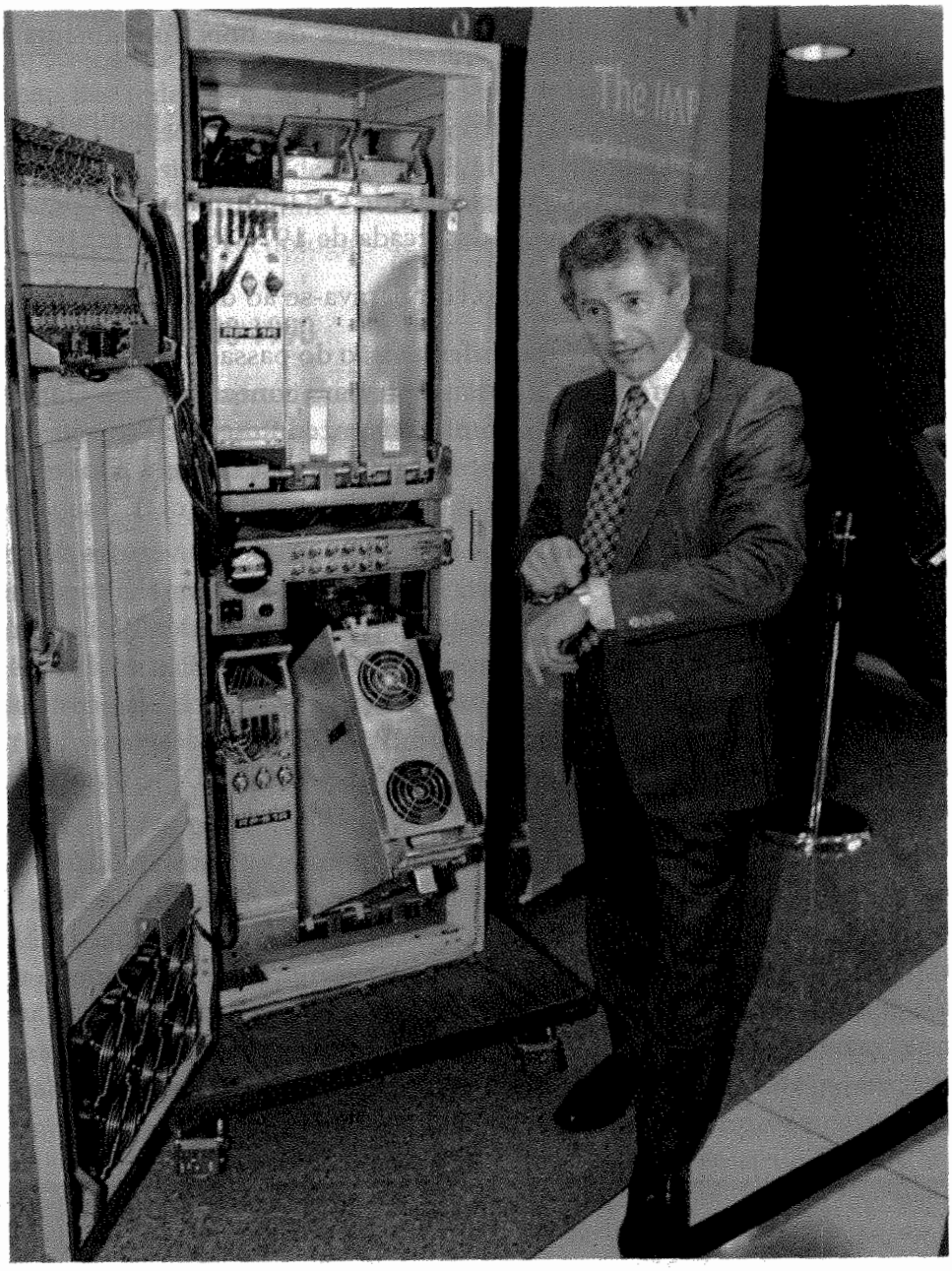
Os primeiros passos da disciplina de redes de computadores e da Internet podem ser traçados desde o início da década de 1960, quando a rede telefônica era a rede de comunicação dominante no mundo inteiro. Lembre-se de que na Seção 1.3 dissemos que a rede de telefonia usa comutação de circuitos para transmitir informações entre uma origem e um destino — uma escolha acertada, já que a voz é transmitida a uma taxa constante entre a origem e o destino. Dada a importância cada vez maior (e o alto custo) dos computadores no início da década de 1960 e o advento de computadores com multiprogramação (*time-sharing*), nada seria mais natural (agora que temos uma visão perfeita do passado) do que considerar a questão de como interligar computadores para que pudessem ser compartilhados entre usuários distribuídos em localizações geográficas diferentes. O tráfego gerado por esses usuários provavelmente era intermitente, por *rajadas* — períodos de atividade, como o envio de um comando a um computador remoto, seguidos de períodos de inatividade, como a espera por uma resposta ou o exame de uma resposta recebida.

Três grupos de pesquisa ao redor do mundo, sem que nenhum tivesse conhecimento do trabalho do outro [Leiner, 1998], começaram a inventar a comutação de pacotes como uma alternativa poderosa e eficiente à de circuitos. O primeiro trabalho publicado sobre técnicas de comutação de pacotes foi o de Leonard Kleinrock [Kleinrock, 1961, 1964], que, naquela época, era um doutorando do MIT. Usando a teoria de filas, o trabalho de Kleinrock demonstrou, com elegância, a eficácia da abordagem da comutação de pacotes para fontes de tráfego intermitentes (*em rajadas*). Em 1964, Paul Baran [Baran, 1964], do Rand Institute, começou a investigar a utilização de comutação de pacotes na transmissão segura de voz pelas redes militares, ao mesmo tempo que Donald Davies e Roger Scantlebury desenvolviam suas idéias sobre esse assunto no National Physical Laboratory, na Inglaterra.

Os trabalhos desenvolvidos no MIT, no Rand Institute e no National Physical Laboratory foram os alicerces do que hoje é a Internet. Mas a Internet também tem uma longa história de atitudes do tipo "construir e demonstrar", que também data do início da década de 1960. J.C.R. Licklider [DEC, 1990] e Lawrence Roberts, ambos colegas de Kleinrock no MIT, foram adiante e lideraram o programa de ciência de computadores na ARPA (Advanced Research Projects Agency — Agência de Projetos de Pesquisa Avançada), nos Estados Unidos. Roberts publicou um plano geral para a ARPAnet [Roberts, 1967], a primeira rede de computadores por comutação de pacotes e uma ancestral direta da Internet pública de hoje. Os primeiros comutadores de pacotes eram conhecidos como **processadores de mensagens de interface** (*interface message processors* — IMPs), e o contrato para a fabricação desses comutadores foi entregue à empresa BBN. Em 1969, no Dia do Trabalho nos Estados Unidos, foi instalado o primeiro IMP na UCLA (Universidade da Califórnia em Los Angeles) sob a supervisão de Kleinrock. Logo em seguida foram instalados três IMPs adicionais no Stanford Research Institute (SRI), na Universidade da Califórnia em Santa Bárbara e na Universidade de Utah (Figura 1.19).

O incipiente precursor da Internet tinha quatro nós no final de 1969. Kleinrock recorda que a primeiríssima utilização da rede foi fazer um login remoto entre a UCLA e o SRI, derrubando o sistema [Kleinrock, 2004].

Em 1972, a ARPAnet tinha aproximadamente 15 nós e foi apresentada publicamente pela primeira vez por Robert Kahn na Conferência Internacional sobre Comunicação por Computadores (International Conference on Computer Communications) daquele ano. O primeiro protocolo fim-a-fim entre sistemas finais da ARPAnet, conhecido como protocolo de controle de rede (*network-control protocol* — NCP), esta-



gura 1.19 Um dos primeiros processadores de mensagens de interface (IMP) e L. Kleinrock (Mark J. Terrill, AP/Wide World Photos)

i concluído [RFC 001] e a partir desse momento a escrita de aplicações tornou-se possível. Em 1972, Ray mlinson, da BBN, escreveu o primeiro programa de e-mail.

8.2 Redes proprietárias e trabalho em rede: 1972-1980

A ARPAnet inicial era um rede isolada, fechada. Para se comunicar com uma máquina da ARPAnet, era eciso estar ligado a um outro IMP dessa rede. Do início a meados de 1970, surgiram novas redes de comu-;ão de pacotes:

- ALOHAnet, uma rede de microondas ligando universidades das ilhas do Havaí [Abramson, 1970], bem como as redes de pacotes por satélite [RFC 829] e por rádio [Kahn, 1978] da DARPA [Kahn, 1978]

- Telenet, uma rede comercial de comutação de pacotes da BBN fundamentada na tecnologia ARPAnet
- Cyclades, uma rede de comutação de pacotes pioneira na França, montada por Louis Pouzin [Think, 2002]
- Redes de tempo compartilhado como a Tymnet e a rede GE Information Services, entre outras que surgiram no final da década de 1960 e início da década de 1970 [Schwartz, 1977]
- Rede SNA da IBM (1969–1974), cujo trabalho comparava-se ao da ARPAnet [Schwartz, 1977].

O número de redes estava crescendo. Hoje, com perfeita visão do passado, podemos perceber que aquela era a hora certa para desenvolver uma arquitetura abrangente para conectar redes. O trabalho pioneiro de interconexão de redes, sob o patrocínio da DARPA (Defense Advanced Research Projects Agency — Agência de Projetos de Pesquisa Avançada de Defesa), criou em essência *uma rede de redes* e foi realizado por Vinton Cerf e Robert Kahn [Cerf, 1974]; o termo *internetting* foi cunhado para descrever esse trabalho.

Esses princípios de arquitetura foram incorporados ao TCP. As primeiras versões desse protocolo, contudo, eram muito diferentes do TCP de hoje. Aquelas versões combinavam uma entrega seqüencial confiável de dados via retransmissão por sistema final (que ainda faz parte do TCP de hoje) com funções de envio (que hoje são desempenhadas pelo IP). As primeiras experiências com o TCP, combinadas com o reconhecimento da importância de um serviço de transporte fim-a-fim não confiável, sem controle de fluxo, para aplicações como voz em pacotes, levaram à separação entre IP e TCP e ao desenvolvimento do protocolo UDP. Os três protocolos fundamentais da Internet que temos hoje — TCP, UDP e IP — estavam conceitualmente disponíveis no final da década de 1970.

Além das pesquisas sobre a Internet realizadas pela DARPA, muitas outras atividades importantes relacionadas ao trabalho em rede estavam em curso. No Havaí, Norman Abramson estava desenvolvendo a ALOHAnet, uma rede de pacotes por rádio que permitia que vários lugares remotos das ilhas havaianas se comunicassem entre si. O ALOHA [Abramson, 1970] foi o primeiro protocolo de acesso múltiplo que permitiu que usuários distribuídos em diferentes localizações geográficas compartilhassem um único meio de comunicação broadcast (uma freqüência de rádio). O trabalho de Abramson sobre protocolo de múltiplo acesso foi aprimorado por Metcalfe e Boggs com o desenvolvimento do protocolo Ethernet [Metcalfe, 1976] para redes compartilhadas de transmissão broadcast por fio; veja a Figura 1.20.

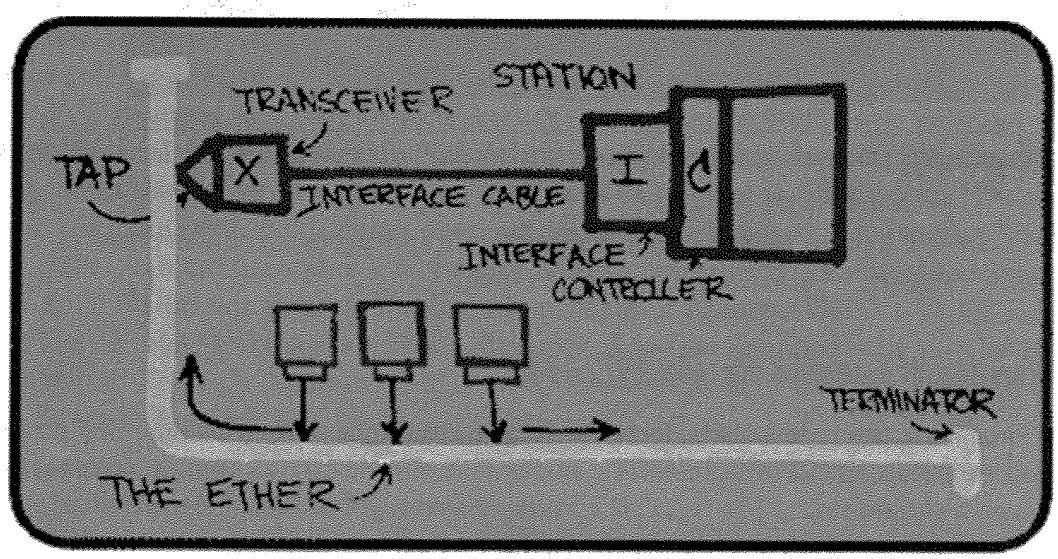


Figura 1.20 A concepção original de Metcalfe para a Ethernet

O interessante é que o protocolo Ethernet de Metcalfe e Boggs foi motivado pela necessidade de conectar vários PCs, impressoras e discos compartilhados [Perkins, 1994]. Há 25 anos, bem antes da revolução do PC e da explosão das redes, Metcalfe e Boggs estavam lançando as bases para as LANs de PCs de hoje. A tecnologia Ethernet representou uma etapa importante para o trabalho em redes interconectadas. Cada rede Ethernet local era, em si, uma rede, e, à medida que o número de LANs aumentava, a necessidade de interconectar essas redes foi se tornando cada vez mais importante. Discutiremos detalhadamente a tecnologia Ethernet, ALOHA e outras tecnologias de LAN no Capítulo 5.

1.8.3 Proliferação de redes: 1980-1990

Ao final da década de 1970, aproximadamente 200 máquinas estavam conectadas à ARPAnet. Ao final da década de 1980, o número de máquinas ligadas à Internet pública, uma confederação de redes muito parecida com a Internet de hoje, alcançaria cem mil. A década de 1980 seria uma época de formidável crescimento.

Grande parte daquele crescimento foi conseqüência de vários esforços distintos para criar redes de computadores para interligar universidades. A BITNET processava e-mails e fazia transferência de arquivos entre diversas universidades do nordeste dos Estados Unidos. A CSNET (computer science network — rede da ciência de computadores) foi formada para interligar pesquisadores de universidades que não tinham acesso à ARPAnet. Em 1986, foi criada a NSFNET para prover acesso a centros de supercomputação patrocinados pela NSF. Partindo de uma velocidade inicial de 56 kbps, ao final da década o backbone da NSFNET estaria funcionando a 1,5 Mbps e servindo como backbone primário para a interligação de redes regionais.

Na comunidade da ARPAnet, já estavam sendo encaixados muitos dos componentes finais da arquitetura da Internet de hoje. No dia 1º de janeiro de 1983, o TCP/IP foi adotado oficialmente como o novo padrão de protocolo de máquinas para a ARPAnet (em substituição ao protocolo NCP). Devido à importância do evento, o dia da transição do NCP para o TCP/IP [RFC 801] foi marcado com antecedência — a partir daquele dia todas as máquinas tiveram de adotar o TCP/IP. No final da década de 1980, foram agregadas importantes extensões ao TCP para implementação do controle de congestionamento baseado em hospedeiros [Jacobson, 1988]. Também foi desenvolvido o sistema de nomes de domínios (DNS) utilizado para mapear nomes da Internet fáceis de entender (por exemplo, gaia.cs.umass.edu) para seus endereços IP de 32 bits [RFC 1034]. Paralelamente ao desenvolvimento da ARPAnet (que em sua maior parte deve-se aos Estados Unidos), no início da década de 1980 os franceses lançaram o projeto Minitel, um plano ambicioso para levar as redes de dados para todos os lares. Patrocinado pelo governo francês, o sistema Minitel consistia em uma rede pública de comutação de pacotes (baseada no conjunto de protocolos X.25, que usava circuitos virtuais), servidores Minitel e terminais baratos com modens de baixa velocidade embutidos. O Minitel transformou-se em um enorme sucesso em 1984, quando o governo francês forneceu, gratuitamente, um terminal para toda residência francesa que quisesse. O sistema Minitel incluía sites de livre acesso — como o da lista telefônica — e também sites particulares, que cobravam uma taxa de cada usuário baseada no tempo de utilização. No seu auge, em meados de 1990, o Minitel oferecia mais de 20 mil serviços, que iam desde home banking até bancos de dados especializados para pesquisa. Era usado por mais de 20 por cento da população da França, gerava receita de mais de um bilhão de dólares por ano e criou dez mil empregos. Estava presente em grande parte dos lares franceses dez anos antes de a maioria dos norte-americanos ouvir falar de Internet.

1.8.4 A explosão da Internet: a década de 1990

A década de 1990 estreou com vários eventos que simbolizaram a evolução contínua e a comercialização iminente da Internet. A ARPAnet, a progenitora da Internet, deixou de existir. Durante a década de 1980, a MILNET e a Defense Data Network (Rede de Dados de Defesa) cresceram e passaram a carregar a maior parte do tráfego do Departamento de Defesa dos Estados Unidos e a NSFNET começou a servir como uma rede de backbone conectando redes regionais nos Estados Unidos com nacionais no exterior. Em 1991,

a NSFNET extinguiu as restrições que impunha à sua utilização com finalidades comerciais, mas, em 1995, perderia seu mandato quando o tráfego de backbone da Internet passou a ser carregado por provedores de serviços de Internet.

O principal evento da década de 1990, no entanto, foi o surgimento da World Wide Web, que levou a Internet para os lares e as empresas de milhões de pessoas no mundo inteiro. A Web serviu também como plataforma para a habilitação e a disponibilização de centenas de novas aplicações, inclusive negociação de ações e serviços bancários on-line, serviços multimídia em tempo real e serviços de recuperação de informações. Para um breve histórico dos primórdios da Web, consulte [W3C, 1995].

A Web foi inventada no CERN (European Center for Nuclear Physics — Centro Europeu para Física Nuclear) por Tim Berners-Lee entre 1989 e 1991 [Berners-Lee, 1989], com base em idéias originadas de trabalhos anteriores sobre hipertexto realizados por Bush [Bush, 1945], na década de 1940, e por Ted Nelson [Ziff-Davis, 1998], na década de 1960. Berners-Lee e seus companheiros desenvolveram versões iniciais de HTML, HTTP, um servidor para a Web e um browser — os quatro componentes fundamentais da Web. Os browsers originais do CERN ofereciam apenas uma interface de linha de comando. Perto do final de 1992 havia aproximadamente 200 servidores Web em operação, e esse conjunto de servidores era apenas uma amostra do que estava por vir. Nessa época, vários pesquisadores estavam desenvolvendo browsers da Web com interfaces GUI (*graphical user interface* — interface gráfica de usuário), entre eles Marc Andreesen, que liderou o desenvolvimento do popular browser Mosaic. Em 1994, Marc Andreesen e Jim Clark formaram a Mosaic Communications, que mais tarde se transformou na Netscape Communications Corporation [Cusumano, 1998; Quittner, 1998]. Em 1995, estudantes universitários estavam usando browsers Mosaic e Netscape para navegar na Web diariamente. Nessa época, empresas — grandes e pequenas — começaram a operar servidores Web e a realizar transações comerciais pela Web. Em 1996, a Microsoft começou a fabricar browsers, dando início à guerra dos browsers entre Netscape e Microsoft, vencida pela última alguns anos mais tarde.

A segunda metade da década de 1990 foi um período de tremendo crescimento e inovação para a Internet, com grandes corporações e milhares de novas empresas criando produtos e serviços para a Internet. O correio eletrônico pela Internet (e-mail) continuou a evoluir com leitores ricos em recursos provendo agendas de endereços, anexos, hot links e transporte de multimídia. No final do milênio a Internet dava suporte a centenas de aplicações populares, entre elas quatro de enorme sucesso:

- e-mail, incluindo anexos e correio eletrônico com acesso pela Web;
- a Web, incluindo navegação pela Web e comércio pela Internet;
- serviço de mensagem instantânea, com listas de contato, cujo pioneiro foi o ICQ;
- compartilhamento peer-to-peer de arquivos MP3, cujo pioneiro foi o Napster.

O interessante é que as duas primeiras dessas aplicações de sucesso arrasador vieram da comunidade de pesquisas, ao passo que as duas últimas foram criadas por alguns jovens empreendedores.

No período de 1995 a 2001, a Internet realizou uma viagem vertiginosa nos mercados financeiros. Antes mesmo de se mostrarem lucrativas, centenas de novas empresas da Internet faziam suas ofertas públicas iniciais de ações e começavam a ser negociadas em bolsas de valores. Muitas empresas eram avaliadas em bilhões de dólares sem ter nenhum fluxo significativo de receita. As ações da Internet sofreram uma queda também vertiginosa em 2000-2001, e muitas novas empresas fecharam. Não obstante, várias empresas surgiram como grandes vencedoras no mundo da Internet (mesmo que os preços de suas ações tivessem sofrido com aquela queda), entre elas Microsoft, Cisco, AOL, Yahoo, e-Bay e Amazon.

Durante a década de 1990, a pesquisa e o desenvolvimento de redes também fizeram progressos significativos nas áreas de roteadores e roteamento de alta velocidade (Capítulo 4) e de LANs (Capítulo 5). A comunidade técnica enfrentou os problemas da definição e implantação de um modelo de serviço de Internet para tráfego que exigisse limitações em tempo real, como aplicações de mídia contínua (Capítulo

). A necessidade de dar segurança e gerenciar a infra-estrutura da Internet (capítulos 8 e 9) também adquiiu extrema importância, pois as aplicações de comércio eletrônico proliferaram e a Internet se tornou um omponente central da infra-estrutura mundial de telecomunicações.

.8.5 Desenvolvimentos recentes

A inovação na área de redes de computadores continua a passos largos. Há progressos em todas as frenes, incluindo desenvolvimento de novas aplicações, distribuição de conteúdo, telefonia por Internet, velocidades de transmissão mais altas em LANs e roteadores mais rápidos. Mas três desenvolvimentos merecem tenção especial: a proliferação de redes de acesso de alta velocidade (incluindo acesso sem fio), a segurana e as redes P2P.

Como discutimos na Seção 1.4, a penetração cada vez maior do acesso residencial de banda larga à nternet via modem a cabo e DSL está montando o cenário para uma profusão de novas aplicações multinídia, entre elas vídeo por demanda em tempo real e videoconferência interativa de alta qualidade. A crescente onipresença de redes Wi-Fi públicas de alta velocidade (11 Mbps e maiores) e de acesso de média elocidade (centenas de kbps) à Internet por redes de telefonia celular não está apenas possibilitando conexão constante, mas também habilitando um novo conjunto muito interessante de serviços específicos para ocalizações determinadas. Abordaremos redes sem fio e redes móveis no Capítulo 6.

Em seguida a uma série de ataques de recusa de serviço em importantes servidores Web no final la década de 1990 e à proliferação de ataques de warms (por exemplo, o Blaster) que infectam sistemas inais e emperram a rede com tráfego excessivo, a segurança da rede tornou-se uma questão extremamente mportante. Esses ataques resultaram no desenvolvimento de sistemas de detecção de invasores capazes de revenir ataques com antecedência, na utilização de firewalls para filtrar tráfego indesejado antes que entre a rede, e na utilização de investigação IP (*traceback*) para localizar a origem dos ataques. Abordaremos ários tópicos importantes relacionados à segurança no Capítulo 8.

A última inovação que queremos destacar são as redes P2P. Uma aplicação de rede P2P explora os recuros de computadores de usuários — armazenagem, conteúdo, ciclos de CPU e presença humana — e tem ignificativa autonomia em relação a servidores centrais. A conectividade dos computadores de usuários isto é, dos pares, ou peers) normalmente é intermitente. Quando escrevíamos este livro, o KaZaA era o sisema P2P de compartilhamento de arquivos mais popular. Há mais de 4 milhões de sistemas finais conecados à sua rede e seu tráfego constitui entre 20 e 50 por cento do tráfego de toda a Internet [Saroiu, 2002].

.9 Resumo

Neste capítulo, abordamos uma quantidade imensa de assuntos. Examinamos as várias peças de hardvare e software que compõem a Internet, em particular, e redes de computadores, em geral. Começamos ela periferia da rede, examinando sistemas finais e aplicações, além do serviço de transporte fornecido às plicações que executam nos sistemas finais. Usando como exemplo as aplicações distribuídas baseadas em edes, apresentamos a noção de um protocolo — um conceito fundamental para redes. Em seguida, mergulhamos no interior da rede e chegamos ao seu núcleo, identificando comutação de pacotes e comutação de ircuitos como as duas abordagens básicas do transporte de dados por uma rede de telecomunicações, xpondo os pontos fortes e fracos de cada uma delas. Examinamos, então, as partes mais inferiores (do onto de vista da arquitetura) da rede — as tecnologias de camada de enlace e os meios físicos comumente encontrados na rede de acesso. Examinamos também a estrutura da Internet global e aprendemos que ela uma rede de redes. Vimos que a estrutura hierárquica da Internet, composta de ISPs de níveis mais altos mais baixos, permitiu que ela se expandisse e incluísse milhares de redes.

Na segunda parte deste capítulo introdutório, abordamos diversos tópicos fundamentais da área de rede le computadores. Primeiramente examinamos as causas de atrasos e perdas de pacotes em uma rede de

comutação de pacotes. Desenvolvemos modelos quantitativos simples de atrasos de transmissão, de propagação e de fila; esses modelos de atrasos serão muito usados nos problemas propostos em todo o livro. Em seguida examinamos camadas de protocolo e modelos de serviço, princípios fundamentais de arquitetura de redes aos quais voltaremos a nos referir neste livro. Concluímos nossa introdução a redes com um breve histórico. Este primeiro capítulo é, em si, um minicurso de redes de computadores.

Portanto, percorremos realmente um extraordinário caminho neste primeiro capítulo! Se você estiver um pouco assustado, não se preocupe. Abordaremos todas essas idéias em detalhes nos capítulos seguintes (é uma promessa, e não uma ameaça!). Por enquanto, esperamos que, ao encerrar este capítulo, você tenha adquirido uma noção, ainda que incipiente, das peças que formam uma rede, um domínio ainda em desenvolvimento do vocabulário (não se acanhe de voltar aqui para consulta) e um desejo cada vez maior de aprender mais sobre redes. Esta é a tarefa que nos espera no restante deste livro.

O guia deste livro

Antes de iniciarmos qualquer viagem, sempre é bom consultar um guia para nos familiarizar com as estradas principais e desvios que encontraremos pela frente. O destino final da viagem que estamos prestes a empreender é um entendimento profundo do como, do quê e do porquê das redes de computadores. Nosso guia é a seqüência de capítulos deste livro:

1. Redes de computadores e a Internet
2. Camada de aplicação
3. Camada de transporte
4. Camada de rede
5. Camada de enlace e redes locais (LANs)
6. Sem fio e mobilidade
7. Redes multimídia
8. Segurança em redes de computadores
9. Gerenciamento de rede

Os capítulos 2 a 5 são os quatro capítulos centrais deste livro. Note que esses capítulos estão organizados segundo as quatro camadas superiores da pilha de cinco camadas do protocolo da Internet, com um capítulo para cada camada. Note também que nossa jornada começará no topo da pilha do protocolo da Internet, a saber, a camada de aplicação, e prosseguirá daí para baixo. O princípio racional que orienta essa jornada de cima para baixo é que, entendidas as aplicações, podemos compreender os serviços de rede necessários para dar suporte a elas. Então, poderemos examinar, um por um, os vários modos como esses serviços poderiam ser implementados por uma arquitetura de rede. Assim, o estudo das aplicações logo no início dá motivação para o restante do livro.

A segunda metade deste livro — capítulos 6 a 9 — aborda quatro tópicos extremamente importantes (e de certa maneira independentes) de redes modernas. No Capítulo 6, examinamos tecnologia sem fio e mobilidade, incluindo LANs Wi-Fi, GSM e IP móvel. No Capítulo 7 (Redes multimídia), examinamos aplicações de áudio e vídeo, como telefone por Internet, videoconferência e recepção de mídia armazenada. Examinamos também como uma rede de comutação de pacotes pode ser projetada para prover serviço de qualidade consistente para aplicações de áudio e vídeo. No Capítulo 8 (Segurança em redes de computadores), analisamos, primeiramente, os fundamentos da criptografia e da segurança de redes e, em seguida, de que modo a teoria básica está sendo aplicada a um amplo leque de contextos da Internet. No último capítulo (Gerenciamento de redes), examinamos as questões fundamentais do gerenciamento de redes, bem como os protocolos primários da Internet utilizados para esse fim.

Exercícios de fixação

Capítulo 1 Questões de revisão

Seções 1.1 a 1.5

1. Qual é a diferença entre um hospedeiro e um sistema final? Cite os tipos de sistemas finais. Um servidor Web é um sistema final?
2. A palavra *protocolo* é muito usada para descrever relações diplomáticas. Dê um exemplo de um protocolo diplomático.
3. O que é um programa cliente? O que é um programa servidor? Um programa servidor requisita e recebe serviços de um programa cliente?
4. Quais são os dois tipos de serviços de transporte que a Internet provê às suas aplicações? Cite algumas características de cada um desses serviços.
5. Afirma-se que controle de fluxo e controle de congestionamento são equivalentes. Isso é válido para o serviço orientado para conexão da Internet? Os objetivos do controle de fluxo e do controle de congestionamento são os mesmos?
6. Utilizando uma analogia com nossos atos, faça uma breve descrição de como o serviço orientado para conexão da Internet provê transporte confiável.
7. Qual é a vantagem de uma rede de comutação de circuitos em relação a uma de comutação de pacotes? Quais são as vantagens da TDM sobre a FDM em uma rede de comutação de circuitos?
8. Por que se afirma que comutação de pacotes emprega multiplexação estatística? Compare a multiplexação estatística com a multiplexação que ocorre em TDM.
9. Suponha que exista exatamente um comutador de pacotes entre um computador de origem e um de destino. As taxas de transmissão entre a máquina de origem e o comutador e entre este e a máquina de destino são R_1 e R_2, respectivamente. Admitindo que um roteador use comutação de pacotes do tipo armazena-e-reenvia, qual é o atraso total fim-a-fim para enviar um pacote de comprimento L? (Desconsidere formação de fila, atraso de propagação e atraso de processamento.)
10. O que quer dizer informação de estado de conexão em uma rede de circuitos virtuais? Se a taxa de estabelecimento e interrupção de conexões em um comutador de uma rede de CVs for de uma conexão por milissegundo (em média), a que taxa a tabela de transmissão do comutador precisa ser modificada?
11. Suponha que você esteja desenvolvendo o padrão para um novo tipo de rede de comutação de pacotes e precisa decidir se sua rede usará CVs ou roteamento de datagramas. Quais são os prós e os contras da utilização de CVs?
12. Cite seis tecnologias de acesso. Classifique cada uma delas nas categorias acesso residencial, acesso corporativo ou acesso móvel.
13. Qual é principal diferença que distingue ISPs de nível 1 e de nível 2?
14. Qual é a diferença entre um POP e um NAP?
15. A taxa de transmissão HFC é dedicada ou é compartilhada entre usuários? É possível haver colisões na direção provedor-usuário de um canal HFC? Por quê?
16. Qual é a taxa de transmissão de LANs Ethernet? Para uma dada taxa de transmissão, cada usuário da LAN pode transmitir continuamente a essa taxa?
17. Cite alguns meios físicos utilizados para instalar a Ethernet.
18. Modens discados, HFC e ADSL são usados para acesso residencial. Para cada uma dessas tecnologias de acesso, cite uma faixa de taxas de transmissão e comente se a largura de banda é compartilhada ou dedicada.

Seções 1.6 a 1.8

19. Considere o envio de um pacote de uma máquina de origem a uma de destino por uma rota fixa. Relacione os componentes do atraso que formam o atraso fim-a-fim. Quais deles são constantes e quais são variáveis?

20. Cite cinco tarefas que uma camada pode executar. É possível que uma (ou mais) dessas tarefas seja(m) realizada(s) por duas (ou mais) camadas?

21. Quais são as cinco camadas da pilha do protocolo da Internet? Quais as principais responsabilidades de cada uma dessas camadas?

22. O que é uma mensagem de camada de aplicação? Um segmento de camada de transporte? Um datagrama de camada de rede? Um quadro de camada de enlace?

23. Que camadas da pilha do protocolo da Internet um roteador implementa? Que camadas um comutador de camada de enlace implementa? Que camadas um sistema final implementa?

Problemas

1. Projete e descreva um protocolo de nível de aplicação para ser usado entre um caixa automático e o computador central de um banco. Esse protocolo deve permitir verificação do cartão e da senha de um usuário, consulta do saldo de sua conta (que é mantido no computador central) e saque de dinheiro da conta corrente (isto é, entrega de dinheiro ao usuário). As entidades do protocolo devem estar habilitadas a resolver o caso comum em que não há dinheiro suficiente na conta do usuário para cobrir o saque. Faça uma especificação de seu protocolo relacionando as mensagens trocadas e as ações realizadas pelo caixa automático ou pelo computador central do banco na transmissão e recepção de mensagens. Esquematize a operação de seu protocolo para o caso de um saque simples sem erros, usando um diagrama semelhante ao da Figura 1.2. Descreva explicitamente o que seu protocolo espera do serviço de transporte fim-a-fim.

2. Considere uma aplicação que transmita dados a uma taxa constante (por exemplo, a origem gera uma unidade de dados de N bits a cada k unidades de tempo, onde k é pequeno e fixo). Considere também que, quando essa aplicação começa, continuará em funcionamento por um período de tempo relativamente longo. Responda às seguintes perguntas, dando uma breve justificativa para sua resposta:

- **a.** O que seria mais apropriado para essa aplicação: uma rede de comutação de circuitos ou uma rede de comutação de pacotes? Por quê?
- **b.** Suponha que seja usada uma rede de comutação de pacotes e que o único tráfego dessa rede venha de aplicações como a descrita anteriormente. Além disso, admita que a soma das velocidades de dados da aplicação seja menor do que as capacidades de cada um dos enlaces. Será necessário algum tipo de controle de congestionamento? Por quê?

3. Considere a rede de comutação de circuitos da Figura 1.5. Lembre-se de que há n circuitos em cada enlace.

- **a.** Qual é o número máximo de conexões simultâneas que podem estar em curso a qualquer instante nessa rede?
- **b.** Suponha que todas as conexões sejam entre o comutador do canto superior esquerdo e o comutador do canto inferior direito. Qual é o número máximo de conexões simultâneas que podem estar em curso?

4. Considere novamente a analogia do comboio de carros da Seção 1.6. Admita novamente uma velocidade de propagação de 100 km/h.

a. Suponha que o comboio viaje 200 km, começando em frente a um dos postos de pedágio, passando por um segundo e terminando em um terceiro. Qual é o atraso fim-a-fim?

b. Repita o item 'a' admitindo agora que haja sete carros no comboio em vez de dez.

5. Considere o envio de um pacote de F bits por um caminho de Q enlaces. Cada enlace transmite a uma velocidade de R bps. A rede está levemente carregada, portanto não há atrasos de fila. O atraso de propagação é desprezível.

a. Suponha que a rede seja de comutação de pacotes por circuitos virtuais. Designe o tempo de estabelecimento de CVs em t_s segundos. Suponha que as camadas de envio agreguem um total de h bits de cabeçalho a cada pacote. Quanto tempo demoraria para enviar o arquivo da origem ao destino?

b. Suponha que a rede seja de datagramas por comutação de pacotes e que o serviço utilizado não orientado para conexão. Agora suponha que cada pacote tenha $2h$ bits de cabeçalho. Quanto tempo demora para enviar o pacote?

c. Finalmente, suponha que a rede seja de comutação de circuitos e que a velocidade de transmissão do circuito entre origem e destino seja R bps. Admitindo tempo de conexão t_s e h bits de cabeçalho anexados ao pacote, quanto tempo levará para enviar esse pacote?

6. Este problema elementar começa a explorar atrasos de propagação e de transmissão, dois conceitos centrais em redes de computadores. Considere dois computadores, A e B, conectados por um único enlace de taxa R bps. Suponha que esses computadores estejam separados por m metros e que a velocidade de propagação ao longo do enlace seja de s metros/segundo. O computador A tem de enviar um pacote de L bits ao computador B.

a. Expresse o atraso de propagação, d_{prop}, em termos de m e s.

b. Determine o tempo de transmissão do pacote, d_{trans}, em termos de L e R.

c. Ignorando os atrasos de processamento e de fila, obtenha uma expressão para o atraso fim-a-fim.

d. Suponha que o computador A comece a transmitir o pacote no instante $t = 0$. No instante $t = d_{trans}$, onde estará o último bit do pacote?

e. Suponha que d_{prop} seja maior do que d_{trans}. Onde estará o primeiro bit do pacote no instante $t = d_{trans}$?

f. Suponha que d_{prop} seja menor do que d_{trans}. Onde estará o primeiro bit do pacote no instante $t = d_{trans}$?

g. Suponha $s = 2{,}5 \cdot 10^8$, $L = 100$ bits e $R = 28$ kbps. Encontre a distância m de forma que d_{prop} seja igual a d_{trans}.

7. Neste problema, consideramos o envio de voz do computador A para o computador B por meio de uma rede de comutação de pacotes (por exemplo, telefone por Internet). O computador A converte voz analógica para uma cadeia digital de bits de 64 kbps e, em seguida, agrupa os bits em pacotes de 48 bytes. Há apenas um enlace entre os computadores A e B; sua taxa de transmissão é de 1 Mbps e seu atraso de propagação, de 2 milissegundos. Assim que o computador A recolhe um pacote, ele o envia ao computador B. Quando recebe um pacote completo, o computador B converte os bits do pacote em um sinal analógico. Quanto tempo decorre entre o momento em que um bit é criado (a partir do sinal analógico no computador A) e o momento em que ele é decodificado (como parte do sinal analógico no computador B)?

8. Suponha que usuários compartilhem um enlace de 1 Mbps e que cada usuário precise de 100 kbps para transmitir, mas que transmita apenas durante 10 por cento do tempo. (Veja a discussão comutação de pacotes *versus* comutação de circuitos na Seção 1.3.)

a. Quando é utilizada comutação de circuitos, quantos usuários podem ter suporte?

b. Para o restante deste problema, suponha que seja utilizada comutação de pacotes. Determine a probabilidade de um dado usuário estar transmitindo.

c. Suponha que haja 40 usuários. Determine a probabilidade de, a qualquer momento, n usuários transmitirem simultaneamente. (*Dica*: use a distribuição binomial.)

d. Determine a probabilidade de haver 11 ou mais usuários transmitindo simultaneamente.

9. Considere a discussão na Seção 1.3 sob o título 'Comutação de Pacotes *versus* Comutação de Circuitos', na qual é dado um exemplo com um enlace de 1 Mbps. Quando em atividade, os usuários estão gerando dados a uma taxa de 100 kpbs; mas a probabilidade de estarem em atividade, gerando dados, é de $p = 0{,}1$. Suponha que o enlace de 1 Mbps seja substituído por um enlace de 1 Gbps.

a. Qual é o número máximo de usuários, N, que pode ser suportado simultaneamente por comutação de pacotes?

b. Agora considere comutação de circuitos e um número M de usuários. Elabore uma fórmula (em termos de p, M, N) para a probabilidade de que mais de N usuários estejam enviando dados.

10. Considere o atraso de fila em um buffer de roteador (antes de um enlace de saída). Suponha que todos os pacotes tenham L bits, que a taxa de transmissão seja de R bits e que N pacotes cheguem simultaneamente ao buffer a cada LN/R segundos. Determine o atraso de fila médio para um pacote. (*Dica*: o atraso de fila para o primeiro pacote é zero; para o segundo pacote, L/R; para o terceiro pacote, $2L/R$. O pacote de ordem N já terá sido transmitido quando o segundo lote de pacotes chegar.)

11. Considere o atraso de fila em um buffer de roteador, sendo I a intensidade de tráfego; isto é, $I = La/R$. Suponha que o atraso de fila tome a forma de $IL/R\ (1 - I)$ para $I < 1$.

a. Deduza uma fórmula para o atraso total, isto é, para o atraso de fila mais o atraso de transmissão.

b. Faça um gráfico do atraso total como uma função de L/R.

12. a. Generalize a fórmula para o atraso fim-a-fim dada na Seção 1.6 para taxas de processamento, atrasos de propagação e taxas de transmissão heterogêneos.

b. Repita o item 'a', mas suponha também que haja um atraso de fila médio de d_{fila} em cada nó.

13. Execute o programa Traceroute para verificar a rota entre uma origem e um destino, no mesmo continente, para três horários diferentes do dia.

a. Determine a média e o desvio padrão dos atrasos de ida e volta para cada um dos três horários.

b. Determine o número de roteadores no caminho para cada um dos três. Os caminhos mudaram em algum dos horários?

c. Tente identificar o número de redes ISPs pelas quais o pacote do Traceroute passa entre origem e destino. Roteadores com nomes semelhantes e/ou endereços IP semelhantes devem ser considerados como parte do mesmo ISP. Em suas respostas, os maiores atrasos ocorrem nas interfaces de formação de pares entre ISPs adjacentes?

d. Faça o mesmo para uma origem e um destino em continentes diferentes. Compare os resultados dentro do mesmo continente com os resultados entre continentes diferentes.

14. Suponha que dois computadores, A e B, estejam separados a uma distância de 10 mil quilômetros e conectados por um enlace direto de $R = 1$ Mbps. Suponha que a velocidade de propagação pelo enlace seja de $2.5 \cdot 10^8$ metros por segundo.

a. Calcule o produto largura de banda-atraso $R \cdot t_{\text{prop}}$.

b. Considere o envio de um arquivo de 400 mil bits do computador A para o computador B. Suponha que o arquivo seja enviado continuamente, como se fosse uma única grande mensagem. Qual é o número máximo de bits que estará no enlace a qualquer dado instante?

c. Interprete o produto largura de banda-atraso.

d. Qual é o comprimento (em metros) de um bit no enlace? É maior do que a de um campo de futebol?

e. Derive uma expressão geral para o comprimento de um bit em termos da velocidade de propagação s, da velocidade de transmissão R e do comprimento do enlace m.

15. Com referência ao problema 14, suponha que possamos modificar R. Para qual valor de R o comprimento de um bit será o mesmo que o comprimento do enlace?

16. Considere o problema 14, mas agora com um enlace de $R = 1$ Gbps.

a. Calcule o produto largura de banda-atraso, $R \cdot t_{prop}$.

b. Considere o envio de um arquivo de 400 mil bits do computador A para o computador B. Suponha que o arquivo seja enviado continuamente, como se fosse uma única grande mensagem. Qual será o número máximo de bits que estará no enlace a qualquer dado instante?

c. Qual é o comprimento (em metros) de um bit no enlace?

17. Novamente com referência ao problema 14.

a. Quanto tempo demora para enviar o arquivo, admitindo que ele seja enviado continuamente?

b. Suponha agora que o arquivo seja fragmentado em dez pacotes e que cada pacote contenha 40 mil bits. Suponha que cada pacote seja verificado pelo receptor e que o tempo de transmissão de uma verificação de pacote seja desprezível. Finalmente, admita que o emissor não possa enviar um pacote até que o anterior tenha sido reconhecido. Quanto tempo demorará para enviar o arquivo?

c. Compare os resultados de 'a' e 'b'.

18. Suponha que haja um enlace de microondas de 10 Mbps entre um satélite geoestacionário e sua estação-base na Terra. A cada minuto o satélite tira uma foto digital e a envia à estação-base. Admita uma velocidade de propagação de $2.4 \cdot 10^8$ metros por segundo.

a. Qual é o atraso de propagação do enlace?

b. Qual é o produto largura de banda-atraso, $R \cdot t_{prop}$?

c. Seja x o tamanho da foto. Qual é o valor mínimo de x para que o enlace de microondas transmita continuamente?

19. Considere a analogia da viagem aérea que utilizamos em nossa discussão sobre camadas na Seção 1.7, e a adição de cabeçalhos a unidades de dados de protocolo enquanto passam por sua pilha. Existe uma noção equivalente de adição de informações de cabeçalho à movimentação de passageiros e suas malas pela pilha do protocolo da linha aérea?

20. Em redes modernas de comutação de pacotes, a máquina de origem segmenta mensagens longas de camada de aplicação (por exemplo, uma imagem ou um arquivo de música) em pacotes menores e os envia pela rede. A máquina destinatária, então, monta novamente os pacotes restaurando a mensagem original. Denominamos esse processo *segmentação de mensagem*. A Figura 1.21 ilustra o transporte fim-a-fim de uma mensagem com e sem segmentação. Considere que uma mensagem de $7{,}5 \cdot 10^6$ bits de comprimento tenha de ser enviada da origem ao destino na Figura 1.21. Suponha que a velocidade de cada enlace da figura seja 1,5 Mbps. Ignore atrasos de propagação, de fila e de processamento.

a. Considere o envio da mensagem da origem ao destino *sem* segmentação. Quanto tempo essa mensagem levará para ir da máquina de origem até o primeiro comutador de pacotes? Tendo em mente que cada comutador usa comutação de pacotes do tipo armazena-e-reenvia, qual é o tempo total para levar a mensagem da máquina de origem à máquina de destino?

b. Agora suponha que a mensagem seja segmentada em 5 mil pacotes, cada um com 1.500 bits de comprimento. Quanto tempo demorará para o primeiro pacote ir da máquina de origem até o primeiro comutador? Quando o primeiro pacote está sendo enviado do primeiro ao segundo comutador, o segundo pacote está sendo enviado da máquina de origem ao primeiro comutador. Em que instante o segundo pacote terá sido completamente recebido no primeiro comutador?

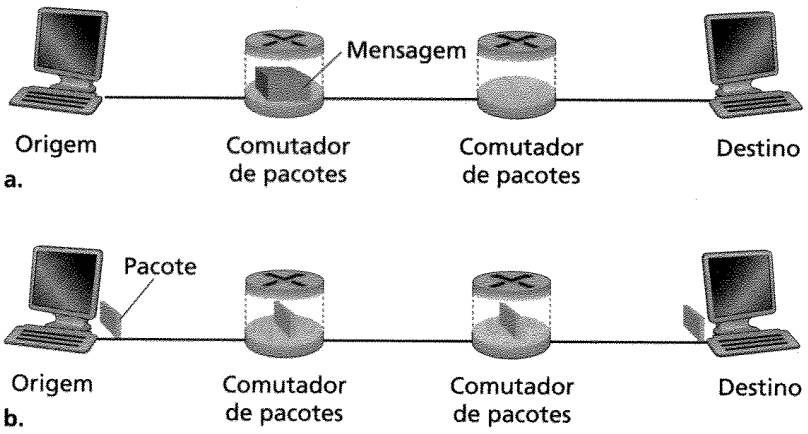


Figura 1.21 Transporte fim-a-fim de mensagem: a) sem segmentação de mensagem; (b) com segmentação de mensagem

c. Quanto tempo demorará para movimentar o arquivo da máquina de origem até a máquina de destino quando é usada segmentação de mensagem? Compare este resultado com sua resposta na parte 'a' e comente.

d. Discuta as desvantagens da segmentação de mensagem.

21. Experimente o applet Java de segmentação de mensagem apresentado no site Web deste livro. Os atrasos no applet correspondem aos atrasos obtidos na pergunta anterior? Como os atrasos de propagação no enlace afetam o atraso total fim-a-fim na comutação de pacotes (com segmentação de mensagem) e na comutação de mensagens?

22. Considere o envio de um arquivo grande de F bits do computador A para o computador B. Há dois enlaces (e um comutador) entre eles e os enlaces não estão congestionados (isto é, não há atrasos de fila). O computador A fragmenta o arquivo em segmentos de S bits cada e adiciona 40 bits de cabeçalho a cada segmento, formando pacotes de $L = 40 + S$ bits. Cada enlace tem uma taxa de transmissão de R bps. Qual é o valor de S que minimiza o atraso para levar o arquivo de A para B? Desconsidere o atraso de propagação.

Questões dissertativas

1. Que tipos de serviços de telefone celular sem fio estão disponíveis em sua área?

2. Usando tecnologia de LAN sem fio 802.11, elabore o projeto de uma rede doméstica para sua casa ou para a casa de seus pais. Relacione os modelos de produtos específicos para essa rede doméstica juntamente com seus custos.

3. O que é "PC para telefone"? Descubra alguns sites de empresas que estão no negócio de "PC para telefone".

4. O que é Short Message Service (SMS)? Esse serviço é popular em algum lugar do mundo? Caso seja, diga onde e qual seu grau de popularidade. É possível enviar uma mensagem SMS de um Web site para um telefone portátil?

5. O que é recepção de áudio armazenado? Descreva alguns dos produtos existentes para a recepção de áudio em tempo real pela Internet. Descubra alguns sites de empresas que estão no negócio de recepção de áudio em tempo real pela Internet.

6. O que é videoconferência pela Internet? Descreva alguns dos produtos existentes para esse tipo de videoconferência. Descubra alguns sites de empresas que estão nesse negócio.

7. Descubra cinco empresas que oferecem serviços de compartilhamento de arquivos P2P. Cite os tipos de arquivos (isto é, conteúdo) que cada empresa processa.
8. O que é mensagem instantânea? Existem produtos que permitem acesso a um serviço de mensagem instantânea por meio de um equipamento de mão?
9. Quem inventou o ICQ, o primeiro serviço de mensagem instantânea? Quando foi inventado e que idade tinham seus inventores? Quem inventou o Napster? Quando foi inventado e que idade tinham seus inventores?
10. Quais são as semelhanças e diferenças entre as tecnologias Wi-Fi e 3G de acesso sem fio à Internet? Quais são as taxas de bits dos dois serviços? Quais são os custos? Discuta roaming e acesso de qualquer lugar.
11. Por que o Napster deixou de existir? O que é a RIAA e que providências está tomando para limitar o compartilhamento de arquivos P2P de conteúdo protegido por direitos autorais? Qual é a diferença entre violação de direitos autorais direta e indireta?
12. Você acha que daqui a dez anos redes de computadores ainda compartilharão amplamente arquivos protegidos por direitos autorais? Por que sim ou por que não? Justifique.

Ethereal Lab

"Conte-me e eu esquecerei. Mostre-me e eu lembrarei. Envolva-me e eu entenderei."
Provérbio chinês

A compreensão de protocolos de rede pode ser muito mais profunda se os virmos em ação e interagirmos com eles — observando a seqüência de mensagens trocadas entre duas entidades de protocolo, pesquisando detalhes de sua operação, fazendo com que eles executem determinadas ações e observando essas ações e suas conseqüências. Isso pode ser feito em cenários simulados ou em um ambiente real de rede, tal como a Internet. Os applets Java apresentados (em inglês) no site deste livro adotam a primeira abordagem. Nos Ethereal labs adotaremos a última. Você executará aplicações de rede em vários cenários utilizando seu computador no escritório, em casa ou em um laboratório e observará também os protocolos de rede interagindo e trocando mensagens com entidades de protocolo que estão executando em outros lugares da Internet. Assim, você e seu computador serão partes integrantes desses laboratórios ao vivo e você observará — e aprenderá — fazendo.

A ferramenta básica para observar as mensagens trocadas entre entidades de protocolos em execução é denominada **analisador de pacotes**. Como o nome sugere, um analisador de pacotes recebe passivamente mensagens enviadas e recebidas por seu computador; também exibe o conteúdo dos vários campos de protocolo das mensagens que captura. Uma tela do analisador de pacotes Ethereal é mostrada na Figura 1.22. O Ethereal é um analisador de pacotes gratuito que funciona em computadores com sistemas operacionais Windows, Linux/Unix e Mac. Nos capítulos 1 a 5 você encontrará Ethereal labs que o habilitarão a explorar vários dos protocolos estudados em cada capítulo. Neste primeiro Ethereal lab, você obterá e instalará uma cópia do programa, acessará um Web site e examinará as mensagens de protocolo trocadas entre seu browser e o servidor Web.

Você encontrará detalhes completos, em inglês, sobre este primeiro Ethereal Lab (incluindo instruções sobre como obter e instalar o programa) no site `www.aw.com/kurose_br`.

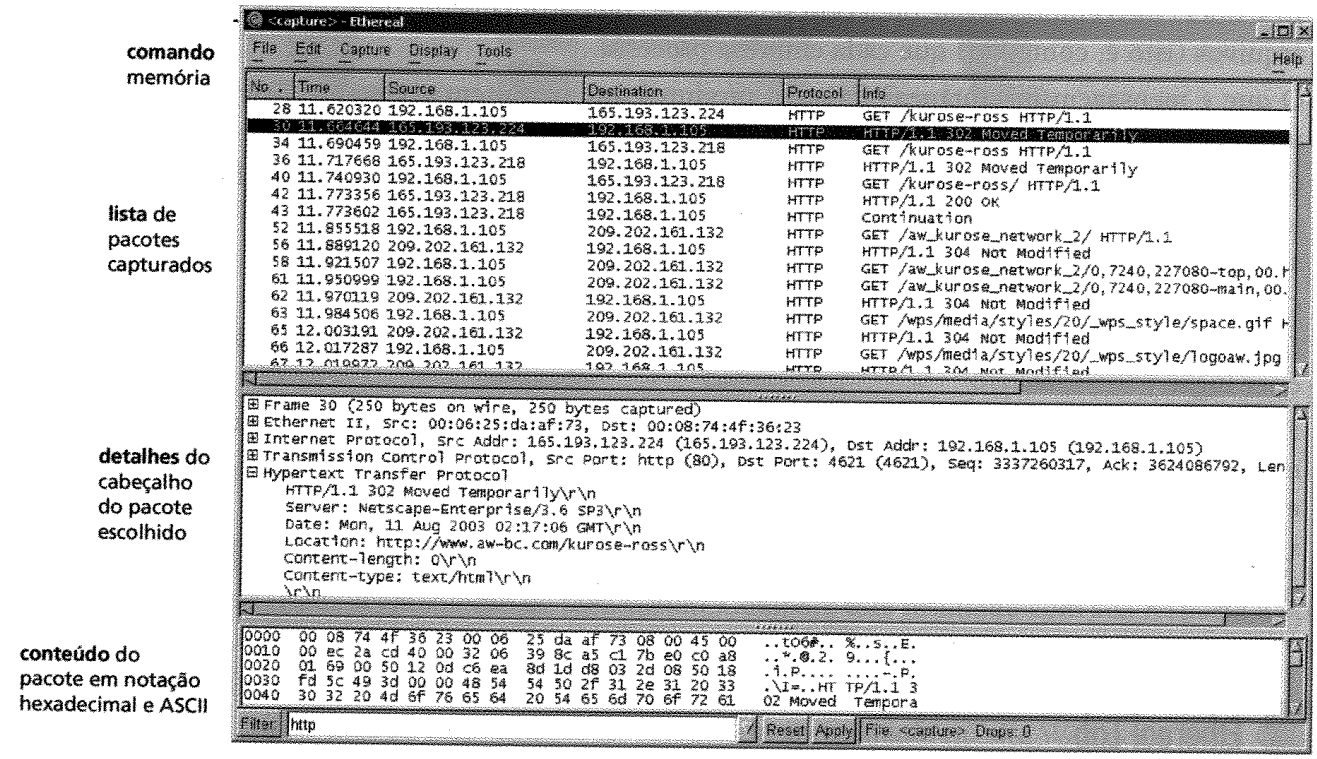


Figura 1.22 Uma amostra de tela do programa Ethereal

Entrevista

Leonard Kleinrock

Leonard Kleinrock é professor de ciência da computação da Universidade da Califórnia em Los Angeles. Em 1969, seu computador na UCLA se tornou o primeiro nó da Internet. Ele criou os princípios da comutação de pacotes em 1961, que se tornou a tecnologia básica da Internet. Leonard também é presidente e fundador da Nomadix, Inc., uma companhia cuja tecnologia oferece maior acessibilidade a serviços de Internet banda larga. Ele é bacharel em engenharia elétrica pela City College of New York (CCNY) e mestre e doutor em engenharia elétrica pelo Instituto de Tecnologia de Massachusetts (MIT).

O que o fez se decidir pela especialização em tecnologia de redes/Internet?

Como doutorando do MIT em 1959, percebi que a maioria dos meus colegas de turma estava fazendo pesquisas nas áreas de teoria da informação e de teoria da codificação. Havia no MIT o grande pesquisador Claude Shannon, que já tinha proposto estudos nessas áreas e resolvido a maior parte dos problemas importantes. Os problemas que restaram para pesquisar eram difíceis e de menor importância. Portanto, decidi propor uma nova área na qual até então ninguém tinha pensado. Lembre-se de que no MIT, eu estava cercado de computadores, e era evidente para mim que brevemente aquelas máquinas teriam de se comunicar umas com as outras. Na época, não havia nenhum meio eficaz de fazer isso; portanto, decidi desenvolver a tecnologia que permitiria a criação de redes de dados eficientes.

Qual foi seu primeiro emprego no setor de computação? O que implicava?

Freqüentei o curso noturno de graduação em engenharia elétrica da CCNY de 1951 a 1957. Durante o dia, trabalhei inicialmente como técnico e depois como engenheiro em uma pequena empresa de eletrônica industrial chamada Photobell. Enquanto trabalhava lá, introduzi tecnologia digital na linha de produtos da empresa. Essencialmente, estávamos usando equipamentos fotoelétricos para detectar a presença de certos itens (caixas, pessoas etc.) e a utilização de um circuito conhecido na época como *multivibrador biestável* era exatamente o tipo de tecnologia de que precisávamos para levar o processamento digital a esse campo da detecção. Acontece que esses circuitos são os blocos de construção básicos dos computadores e vieram a ser conhecidos como *flip-flops* ou *comutadores* na linguagem coloquial de hoje.

O que passou por sua cabeça quando enviou a primeira mensagem computador a computador (da UCLA para o Stanford Research Institute)?

A primeira mensagem computador a computador foi, de certa maneira, um anticlímax. Para mim, o evento mais impressionante aconteceu em 2 de setembro de 1969, quando o primeiro componente de equipamento de rede

(o IMP) foi conectado com o primeiro sistema operacional no mundo exterior (meu computador na UCLA). Foi então que nasceu a Internet. Anteriormente, naquele mesmo ano, fui citado em um comunicado de imprensa da UCLA por ter dito que, logo que a rede estivesse pronta e em funcionamento, seria possível ter acesso a utilidades de outros computadores a partir de nossas casas e escritórios tão facilmente quanto tínhamos acesso à eletricidade e ao telefone. Portanto, a visão que eu tinha da Internet naquela época era que ela seria onipresente, estaria sempre em funcionamento e sempre disponível, que qualquer pessoa que possuísse qualquer equipamento poderia se conectar com ela de qualquer lugar e que ela seria invisível. Mas jamais imaginei que minha mãe, aos 94 anos de idade, estaria ligada à Internet hoje — como ela realmente está.

Em sua opinião, qual é o futuro das redes?

O mais claro para mim é o que se refere à computação nomádica e a espaços inteligentes. A disponibilidade de equipamentos de computação de baixo peso, baratos, de alto desempenho e portáteis, somada à onipresença da Internet, permitiu que nos tornássemos nômades. *Computação nomádica* se refere à tecnologia que habilita usuários finais que viajam de um lugar para outro a ter acesso a serviços da Internet de modo transparente, não importando para onde vão. Contudo, ela é apenas um passo. O passo seguinte vai nos capacitar a sair do mundo misterioso do ciberespaço para o mundo físico dos espaços inteligentes. A tecnologia dará vida a nossos ambientes (mesas, paredes, veículos, relógios e cintos, entre outros) por meio de atuadores, sensores, lógica, processamento, armazenagem, câmeras, microfones, alto-falantes, painéis e comunicação. Essa tecnologia embutida permitirá que nosso ambiente forneça os serviços IP que quisermos. Quando eu entrar em uma sala, ela saberá que eu entrei. Poderei me comunicar com meu ambiente naturalmente, como se estivesse falando o meu idioma nativo; minhas solicitações gerarão respostas apresentadas como páginas Web em painéis de parede, por meus óculos, por voz, por hologramas e assim por diante.

Analisando um panorama mais longínquo, vejo um futuro para as redes que inclui componentes fundamentais que ainda virão. Vejo agentes inteligentes de software distribuídos por toda a rede cuja função é fazer mineração de dados, agir sobre esses dados, observar tendências e adaptar e realizar tarefas dinamicamente. Vejo tráfego de rede consideravelmente maior gerado não tanto por seres humanos, mas por esses equipamentos embutidos e agentes inteligentes de software. Vejo grandes conjuntos de sistemas auto-organizáveis controlando essa rede imensa e veloz. Vejo quantidades enormes de informações zunindo por essa rede instantaneamente e passando por extraordinários processamentos e filtragens. A Internet será, essencialmente, um sistema nervoso de presença global. Vejo tudo isso e mais enquanto entramos de cabeça no século XXI.

Que pessoas o inspiraram profissionalmente?

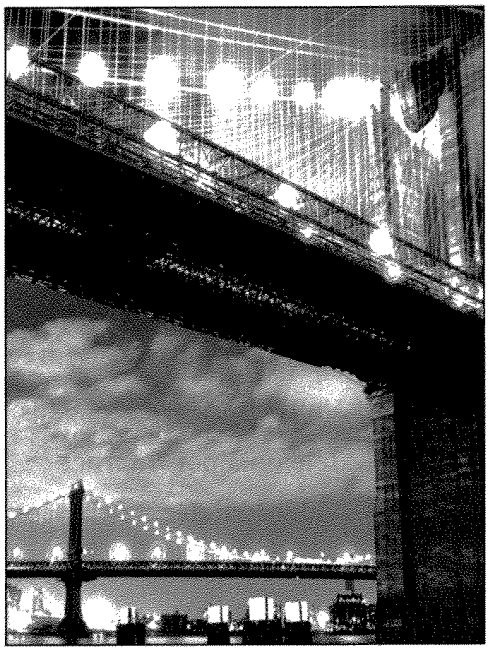
Quem mais me inspirou foi Claude Shannon, do MIT, um brilhante pesquisador que tinha a capacidade de relacionar suas idéias matemáticas com o mundo físico de modo muitíssimo intuitivo. Ele fazia parte da banca examinadora de minha tese de doutorado.

Você pode dar algum conselho aos estudantes que estão ingressando na área de redes/ Internet?

A Internet, e tudo o que ela habilita, é uma vasta fronteira nova, cheia de desafios surpreendentes. Há espaço para grandes inovações. Não fiquem limitados à tecnologia existente hoje. Soltem sua imaginação e pensem no que poderia acontecer e transformem isso em realidade.

Capítulo 2

Camada de Aplicação



Aplicações de rede são a razão de ser de uma rede de computadores. Se não fosse possível inventar aplicações úteis, não haveria necessidade de projetar protocolos de rede para suportá-las. Nos últimos 35 anos, foram criadas numerosas aplicações de rede engenhosas e maravilhosas. Entre elas estão as aplicações clássicas de texto, que se tornaram populares na década de 1980: correio eletrônico, acesso a computadores remotos, transferência de arquivo, grupos de discussão e bate-papo e também uma aplicação que alcançou estrondoso sucesso em meados da década de 1990: a Web. Há ainda muitas aplicações multimídia, como vídeo em tempo real, rádio e telefonia por Internet e videoconferência. Duas aplicações de enorme sucesso também surgiram no final do milênio — mensagem instantânea e compartilhamento não hierárquico de arquivos (*peer-to-peer* — P2P).

Neste capítulo estudamos os aspectos conceituais e de implementação de aplicações de rede. Começamos definindo conceitos fundamentais de camada de aplicação, incluindo protocolos, clientes e servidores, processos, sockets e interfaces de camada de transporte. Então examinamos detalhadamente várias aplicações de rede, entre elas a Web, e-mail, DNS e compartilhamento de arquivos P2P.

Em seguida, abordamos desenvolvimento de aplicação de rede por TCP e também por UDP. Em particular, estudamos o API socket e examinamos algumas aplicações cliente-servidor simples em Java. Por exemplo, exploramos como é possível criar um servidor Web simples em Java. Apresentamos também vários exercícios divertidos e interessantes de programação de aplicações no final do capítulo.

A camada de aplicação é um lugar particularmente bom para iniciarmos o estudo de protocolos. É terreno familiar, pois conhecemos muitas das aplicações que dependem dos protocolos que estudaremos. Ela nos dará uma boa idéia do que são protocolos e nos apresentará muitos assuntos que encontraremos novamente quando estudarmos protocolos de camadas de transporte, de rede e de enlace.

2.1 Princípios de aplicações de rede

Vamos começar relacionando exemplos de aplicações de rede populares:

- Correio eletrônico
- A Web

- Mensagem instantânea
- Login em computador remoto como Telnet e SSH
- Compartilhamento de arquivos P2P
- Transferência de arquivos entre duas contas em dois computadores, como FTP (File Transport Protocol — Protocolo de Transferência de Arquivos)
- Jogos multiusuários em rede
- Videoclipes armazenados
- Telefonia por Internet
- Videoconferência em tempo real

Neste livro discutiremos muitas dessas aplicações com níveis variáveis de detalhamento. Por exemplo, neste capítulo abordaremos e-mail, a Web e compartilhamento de arquivos P2P. No Capítulo 7, que trata de redes multimídia, estudaremos vídeo em tempo real e telefonia por Internet.

Agora, suponha que você tenha uma grande idéia para uma nova aplicação de rede. Essa aplicação será, talvez, um grande serviço para a humanidade, ou agradará a seu professor, ou fará de você um homem (ou mulher) rico(a), ou simplesmente será divertido desenvolvê-la. Seja qual for sua motivação, vamos examinar agora como transformar a idéia em uma aplicação do mundo real.

O cerne do desenvolvimento de aplicação de rede é escrever programas que rodem em sistemas finais diferentes e se comuniquem entre si pela rede. Por exemplo, na aplicação Web há dois programas distintos que se comunicam um com o outro: o programa do browser, que roda na máquina do usuário (computador de mesa, laptop, PDA, telefone celular e assim por diante); e o programa do servidor Web, que roda na máquina do servidor Web. Um outro exemplo é um sistema de compartilhamento de arquivos P2P no qual há um programa em cada máquina que participa da comunidade de compartilhamento de arquivos. Nesse caso, os programas de cada máquina podem ser semelhantes ou idênticos.

Portanto, ao desenvolver sua nova aplicação, você precisará escrever um software que rode em várias máquinas. É importante dizer que você não precisará escrever um software que rode em equipamentos de núcleo de rede, como roteadores ou comutadores Ethernet. E, mesmo que você quisesse escrever software de aplicação para esses equipamentos, não poderia. Como aprendemos no Capítulo 1 e mostramos na Figura 1.18, equipamentos de núcleo de rede não funcionam na camada de aplicação, mas em camadas mais baixas, especificamente na de rede e abaixo dela. Esse projeto básico — a saber, confinar o software de aplicação nos sistemas finais, como mostra a Figura 2.1, facilitou o desenvolvimento e a proliferação rápidos de uma vasta gama de aplicações de Internet.

2.1.1 Arquiteturas de aplicação de rede

Ao construir uma nova aplicação de rede, você primeiramente precisará escolher a arquitetura da aplicação. Realmente, antes de mergulhar na codificação do software, você deverá elaborar um plano geral para a arquitetura da sua aplicação. Tenha sempre em mente que a arquitetura de uma aplicação é distintamente diferente da arquitetura de rede (por exemplo, a arquitetura em cinco camadas da Internet que discutimos no Capítulo 1). Do ponto de vista do profissional que desenvolve a aplicação, a arquitetura de rede é fixa e provê um conjunto específico de serviços às aplicações. Por outro lado, a **arquitetura da aplicação** é projetada pelo desenvolvedor e determina como a aplicação é organizada nos vários sistemas finais. Ao escolher a arquitetura da aplicação, o desenvolvedor provavelmente aproveitará uma das três arquiteturas mais utilizadas em aplicações modernas de rede: a arquitetura cliente-servidor, a arquitetura P2P e uma arquitetura híbrida cliente-servidor/P2P.

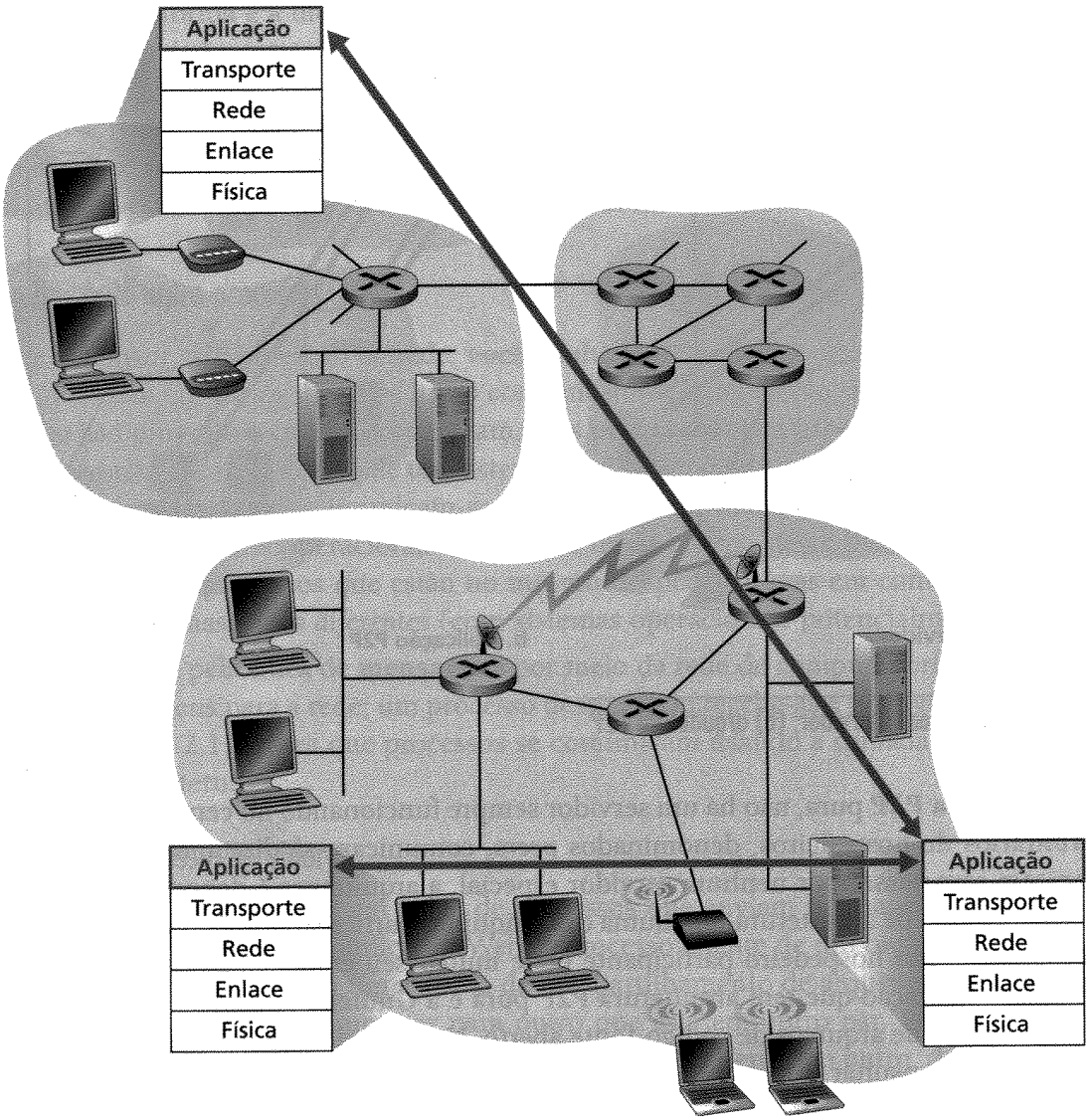
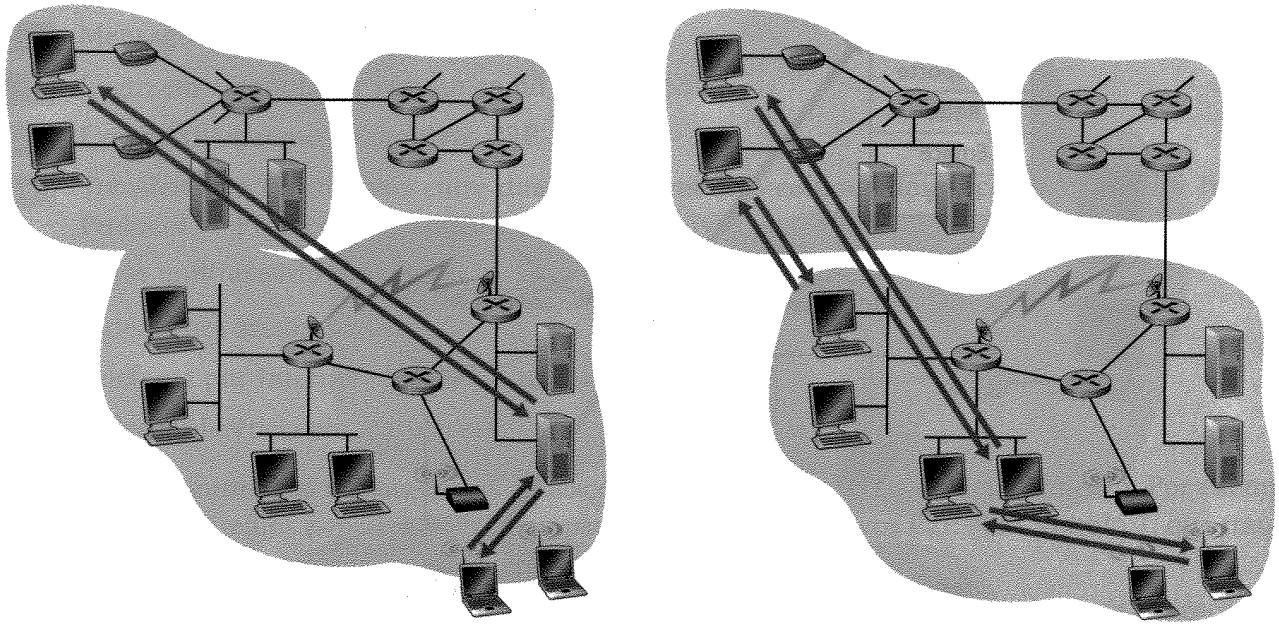


Figura 2.1 A comunicação de uma aplicação de rede ocorre entre sistemas finais na camada de aplicação

Em uma **arquitetura cliente-servidor** há um hospedeiro sempre em funcionamento, denominado *servidor*, que atende a requisições de muitos outros hospedeiros, denominados *clientes*. Estes podem estar em funcionamento às vezes ou sempre. Um exemplo clássico é a aplicação Web na qual um servidor Web que está sempre em funcionamento atende a requisições de browsers de hospedeiros clientes. Quando recebe uma requisição de um objeto de um hospedeiro cliente, um servidor Web responde enviando o objeto requisitado a ele. Observe que, na arquitetura cliente-servidor, os clientes não se comunicam diretamente uns com os outros; por exemplo, na aplicação Web, dois browsers não se comunicam diretamente. Outra característica da arquitetura cliente-servidor é que o servidor tem um endereço fixo, bem conhecido, denominado endereço IP (que discutiremos em breve). Devido a essa característica do servidor e devido ao fato de ele estar sempre em funcionamento, um cliente sempre pode contatá-lo, enviando um pacote ao endereço do servidor. Algumas das aplicações mais conhecidas que empregam a arquitetura cliente-servidor são a Web, a transferência de arquivos, o login remoto e o e-mail. Essa arquitetura cliente-servidor é mostrada na Figura 2.2(a).

Em aplicações cliente-servidor, muitas vezes acontece de um único hospedeiro servidor ser incapaz de atender a todas as requisições de seus clientes. Por exemplo, um site Web de notícias popular pode ficar rapidamente saturado se tiver apenas um servidor para atender a todas as requisições. Por essa razão, muitas vezes são utilizados conjuntos de hospedeiros — às vezes denominados ***server farm*** — para criar um servidor virtual poderoso em arquiteturas cliente-servidor.



a. Aplicação cliente-servidor

b. Aplicação P2P

Figura 2.2 (a) Arquitetura cliente-servidor; (b) arquitetura P2P

Em uma **arquitetura P2P** pura, não há um servidor sempre funcionando no centro da aplicação. Em vez disso, pares arbitrários de hospedeiros, denominados *peers*, comunicam-se diretamente entre si. Como os pares se comunicam sem passar por nenhum servidor especial, a arquitetura é denominada par-a-par (*peer-to-peer* — P2P). Nesse tipo de arquitetura, nenhuma das máquinas participantes precisa estar sempre em funcionamento; além disso, um hospedeiro participante pode mudar seu endereço IP cada vez que for ligado. Um bom exemplo de aplicação que tem arquitetura P2P pura é a Gnutella [Gnutella, 2004], uma aplicação P2P de compartilhamento de arquivos de código-fonte aberto. Na Gnutella, qualquer hospedeiro pode requisitar e enviar arquivos, consultar a localização de um arquivo e responder e transmitir consultas. Examinaremos a Gnutella mais detalhadamente na Seção 2.6. A arquitetura P2P é ilustrada na Figura 2.2(b).

Uma das características mais fortes da arquitetura P2P é sua escalabilidade. Por exemplo, em uma aplicação P2P de compartilhamento de arquivos, milhões de pares podem participar da comunidade de compartilhamento de arquivos, sendo que cada um deles funciona como um servidor e contribui com recursos para a comunidade. Desse modo, conquanto cada par gere carga de trabalho requisitando arquivos, também agrega capacidade de serviço ao sistema respondendo requisições de outros pares. Assim, em princípio, o compartilhamento de arquivos P2P é intrinsecamente escalável — cada par adicional não apenas aumenta a demanda, mas também aumenta a capacidade de serviço. Na Internet de hoje, o tráfego de compartilhamento de arquivos P2P é responsável por uma grande parcela de todo o tráfego [Saroiu, 2002].

Por outro lado, devido à sua natureza altamente distribuída e descentralizada, pode ser difícil gerenciar aplicações P2P. Por exemplo, um par pode ter a única cópia de um arquivo importante e sair da comunidade a qualquer momento. Continua em aberto a questão da possibilidade de criar soluções P2P de grande porte para aplicações empresariais [Adya, 2004].

As duas arquiteturas, cliente-servidor e P2P, são comuns em aplicações de rede. Contudo, muitas aplicações são organizadas segundo arquiteturas **híbridas** cliente-servidor/P2P. Um exemplo disso é a já extinta Napster, a primeira das populares aplicações de compartilhamento de arquivos MP3. A arquitetura da Napster era P2P no sentido de que arquivos MP3 eram trocados diretamente entre pares, sem passar por servidores dedicados, sempre em funcionamento; mas também era cliente-servidor, já que um par consultava um servidor central para determinar quais pares que estavam em funcionamento tinham um arquivo MP3 desejado. (A arquitetura da Napster é estudada mais detalhadamente na Seção 2.6.) Outra aplicação que uti-

liza uma arquitetura híbrida é a mensagem instantânea. Nela, a conversa entre dois usuários é tipicamente P2P, isto é, o texto enviado entre dois usuários não passa por um servidor intermediário, sempre em funcionamento. Entretanto, quando Alice, uma usuária, lança sua aplicação de mensagem instantânea, ela se registra em um servidor central; e quando Bob, um outro usuário, quer conversar com alguém inscrito na sua lista de amigos, seu cliente de mensagem instantânea contata o servidor central para descobrir quais desses seus amigos estão correntemente on-line e disponíveis.

2.1.2 Comunicação entre processos

Antes de construir sua aplicação de rede, você também precisará ter um entendimento básico de como programas que rodam em vários sistemas finais comunicam-se entre si. No jargão de sistemas operacionais, na verdade não são programas que se comunicam, mas **processos**. Um processo pode ser imaginado como um programa que está rodando dentro de um sistema final. Quando os processos comunicantes estão rodando no mesmo sistema final, eles comunicam-se entre si usando comunicação interprocessos, cujas regras são determinadas pelo sistema operacional do sistema final. Porém, neste livro, não estamos interessados em como se comunicam processos que estão no mesmo hospedeiro, mas em como se comunicam processos que rodam em sistemas finais *diferentes* (com sistemas operacionais potencialmente diferentes).

Eles se comunicam pela troca de **mensagens** por meio da rede de computadores. Um processo originador cria e envia mensagens para a rede; um processo destinatário recebe-as e possivelmente responde, devolvendo outras. A Figura 2.1 mostra que processos se comunicam usando a camada de aplicação da pilha de cinco camadas da arquitetura.

Processos clientes e processos servidores

Uma aplicação de rede consiste em pares de processos que enviam mensagens uns para os outros por meio de uma rede. Por exemplo, na aplicação Web, o processo cliente de um browser troca mensagens com o processo de um servidor Web. Em um sistema de compartilhamento de arquivos P2P, um arquivo é transferido de um processo que está em um par para outro que está em outro par. Para cada par de processos comunicantes normalmente rotulamos um dos dois processos de **cliente** e o outro, de **servidor**. Na Web, um browser é um processo cliente e um servidor Web é um processo servidor. No compartilhamento de arquivos P2P, o par que está descarregando o arquivo é rotulado de cliente e o que está recebendo, de servidor.

Talvez você já tenha observado que, em algumas aplicações, tal como compartilhamento de arquivos P2P, um processo pode ser ambos, cliente e servidor. Realmente, um processo em um sistema de compartilhamento de arquivos P2P pode carregar e descarregar arquivos. Mesmo assim, no contexto de qualquer dada sessão entre um par de processos, ainda podemos rotular um processo de cliente e o outro de servidor. Definimos os processos cliente e servidor como segue:

> *No contexto de uma sessão de comunicação entre um par de processos, o processo que inicia a comunicação (isto é, o primeiro a contatar o outro no início da sessão) é rotulado de* ***cliente****. O processo que espera ser contatado para iniciar a sessão é o* ***servidor****.*

Na Web, um processo do browser inicia o contato com um processo do servidor Web; por conseguinte, o processo do browser é o cliente e o processo do servidor Web é o servidor. No compartilhamento de arquivos P2P, quando o Par A solicita ao Par B o envio de um arquivo específico, o Par A é o cliente enquanto o Par B é o servidor no contexto dessa sessão específica de comunicação. Quando não houver possibilidade de confusão, às vezes usaremos também a terminologia "lado cliente e lado servidor de uma aplicação". No final deste capítulo examinaremos passo a passo um código simples para ambos os lados de aplicações de rede: o lado cliente e o lado servidor.

Há um ponto em nossa terminologia que talvez precise ser mais bem esclarecido. Na Seção 2.1.1, classificamos aplicações segundo suas arquiteturas, a saber, cliente-servidor, P2P ou híbrida. Essa classificação

é útil, pois proporciona uma estrutura geral para determinar a arquitetura de aplicações de rede. Todavia, é preciso ter em mente que grande parte das aplicações de rede — incluindo as que utilizam a arquitetura P2P — consiste em vários pares de processos comunicantes; na qualidade de partes de uma sessão de comunicação entre um par de processos, um é rotulado de cliente e o outro, de servidor.

Sockets

Como dissemos anteriormente, a maioria das aplicações consiste em pares de processos comunicantes, sendo que os dois processos de cada par enviam mensagens um para o outro. Qualquer mensagem enviada de um processo para um outro tem de passar pela rede subjacente. Um processo envia mensagens para a rede e recebe mensagens dela através de seu **socket**. Vamos considerar uma analogia que nos auxiliará a entender processos e sockets. Um processo é análogo a uma casa e seu socket, à porta da casa. Quando um processo quer enviar uma mensagem a um outro processo em outro hospedeiro, ele empurra a mensagem porta (socket) afora para dentro da rede. Esse processo emissor admite que exista uma infra-estrutura de transporte do outro lado de sua porta que transportará a mensagem pela rede até a porta do processo destinatário. Ao chegar ao hospedeiro destinatário, a mensagem passa através da porta (socket) do processo receptor, que então executa alguma ação sobre a mensagem.

A Figura 2.3 ilustra a comunicação por socket entre dois processos que se comunicam pela Internet. (A Figura 2.3 admite que o protocolo de transporte subjacente é o TCP, embora o protocolo UDP também possa ser usado na Internet.) Como mostra essa figura, um socket é a interface entre a camada de aplicação e a de transporte dentro de uma máquina. É também denominado **interface de programação da aplicação** (*application programming interface* — API) entre a aplicação e a rede, visto que o socket é a interface de programação pela qual as aplicações de rede são inseridas na Internet. O desenvolvedor da aplicação controla tudo o que existe no lado da camada de aplicação do socket, mas tem pouco controle do lado da camada de transporte do socket. Os únicos controles que o desenvolvedor da aplicação tem do lado da camada de transporte são: (1) a escolha do protocolo de transporte e (2), talvez, a capacidade de determinar alguns parâmetros da camada de transporte, tais como tamanho máximo de buffer e de segmentos. Uma vez escolhido um protocolo de transporte, (se houver escolha) o desenvolvedor constrói a aplicação usando os serviços da camada de transporte oferecidos por esse protocolo. Examinaremos sockets mais detalhadamente nas seções 2.7 e 2.8.

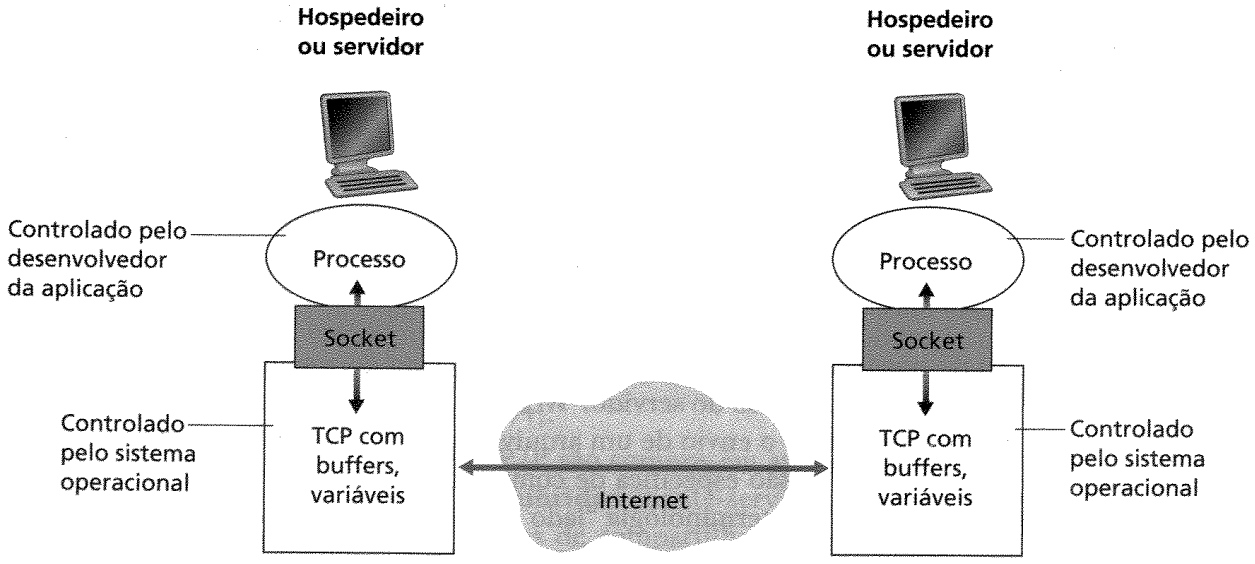


Figura 2.3 Processos de aplicação, sockets e protocolo de transporte subjacente

Endereçamento de processos

Para que um processo em um hospedeiro envie uma mensagem a um processo em outro, o processo originador tem de identificar o processo destinatário. Para isso, é preciso especificar duas informações: (1) o nome ou o endereço da máquina hospedeira e (2) um identificador que especifique o processo destinatário no hospedeiro de destino.

Primeiro, vamos considerar endereços de hospedeiros. Em aplicações da Internet, o hospedeiro de destino é identificado por seu **endereço IP**. Discutiremos endereços IP em detalhes no Capítulo 4. Por enquanto, basta saber que o endereço IP é uma quantidade de 32 bits que identifica *exclusivamente* o sistema final (sendo mais precisos, identifica exclusivamente a interface de rede que liga aquele hospedeiro à Internet). Visto que o endereço IP de qualquer sistema final conectado à Internet pública deve ser *globalmente* exclusivo, a atribuição desses endereços deve ser gerenciada com cuidado, como será discutido no Capítulo 4.

Além de saber o endereço do sistema final ao qual a mensagem se destina, o hospedeiro originador também deve identificar o processo que está rodando no outro hospedeiro. Essa informação é necessária porque, em geral, um hospedeiro poderia estar executando muitas aplicações de rede. Um **número de porta** de destino atende a essa finalidade. Aplicações populares receberam números de porta específicos. Por exemplo, um servidor Web é identificado pelo número de porta 80. Um processo servidor de correio (que usa o protocolo SMTP) é identificado pelo número de porta 25. Uma lista de números bem conhecidos de portas para todos os protocolos padronizados da Internet pode ser encontrada no site http://www.iana.org. Quando um desenvolvedor cria uma nova aplicação de rede, ela deve receber um novo número de porta. Examinaremos números de porta detalhadamente no Capítulo 3.

2.1.3 Protocolos de camada de aplicação

Acabamos de aprender que processos de rede comunicam-se entre si enviando mensagens para dentro de sockets. Mas, como essas mensagens são estruturadas? O que significam os vários campos nas mensagens? Quando os processos enviam as mensagens? Essas perguntas nos transportam para o mundo dos protocolos de camada de aplicação. Um **protocolo de camada de aplicação** define como processos de uma aplicação, que funcionam em sistemas finais diferentes, passam mensagens entre si. Em particular, um protocolo de camada de aplicação define:

- os tipos de mensagens trocadas, por exemplo, de requisição e de resposta;
- a sintaxe dos vários tipos de mensagens, tais como os campos da mensagem e como os campos são delineados;
- a semântica dos campos, isto é, o significado da informação nos campos;
- regras para determinar quando e como um processo envia mensagens e responde a mensagens.

Alguns protocolos de camada de aplicação estão especificados em RFCs e, portanto, são de domínio público. Por exemplo, o protocolo de camada de aplicação da Web, HTTP (HyperText Transfer Protocol [RFC 2616]), está à disposição como um RFC. Se um desenvolvedor de browser seguir as regras do RFC do HTTP, seu browser estará habilitado a extrair páginas de qualquer servidor Web que também tenha seguido as regras do RFC do HTTP. Muitos outros protocolos de camada de aplicação são proprietários e, intencionalmente, não estão disponíveis ao público. Por exemplo, muitos dos sistemas de compartilhamento de arquivos P2P existentes usam protocolos de camada de aplicação proprietários.

É importante distinguir aplicações de rede de protocolos de camada de aplicação. Um protocolo de camada de aplicação é apenas um pedaço (embora grande) de aplicação de rede. Examinemos alguns exemplos. A Web é uma aplicação cliente-servidor que permite aos usuários obter documentos de servidores Web por demanda. A aplicação Web consiste em muitos componentes, entre eles um padrão para formato de documentos (isto é, HTML), browsers Web (por exemplo, Netscape Navigator e Microsoft Internet Explorer), servidores Web (por exemplo, servidores Apache, Microsoft e Netscape) e um protocolo de

camada de aplicação. O protocolo de camada de aplicação da Web, HTTP, define o formato e a seqüência das mensagens que são passadas entre o browser e o servidor Web. Assim, ele é apenas um pedaço da aplicação Web. Como outro exemplo, considere a aplicação correio eletrônico da Internet. O correio eletrônico da Internet também tem muitos componentes, entre eles servidores de correio que armazenam caixas postais de usuários, leitores de correio que permitem aos usuários ler e criar mensagens, um padrão que define como mensagens são passadas entre servidores e entre servidores e leitores de correio e como o conteúdo de certas partes da mensagem de correio (por exemplo, um cabeçalho) deve ser interpretado. O principal protocolo de camada de aplicação para o correio eletrônico é o SMTP — Simple Mail Protocol [RFC 2821]. Assim, o SMTP é apenas um pedaço (embora grande) da aplicação correio eletrônico.

2.1.4 De que serviços uma aplicação necessita?

Lembre-se de que um socket é a interface entre o processo da aplicação e o protocolo de camada de transporte. A aplicação do lado remetente envia mensagens através da porta. Do outro lado da porta, o protocolo de camada de transporte tem a responsabilidade de levar as mensagens pela rede até a porta do processo destinatário. Muitas redes, inclusive a Internet, oferecem mais de um protocolo de camada de transporte. Ao desenvolver uma aplicação, você deve escolher um dos protocolos de camada de transporte disponíveis. Como fazer essa escolha? O mais provável é que você avalie os serviços providos pelos protocolos de camada de transporte disponíveis e escolha o protocolo que melhor atenda às necessidades de sua aplicação. A situação é semelhante a escolher trem ou avião como meio de transporte entre duas cidades. Você tem de escolher um ou outro, e cada modalidade de transporte oferece serviços diferentes. Por exemplo, o trem oferece a facilidade da partida e da chegada no centro da cidade, ao passo que o avião oferece menor tempo de viagem.

Quais são os serviços que uma aplicação necessitaria de um protocolo de transporte? Podemos classificar, de maneira geral, as necessidades de serviço de uma aplicação segundo três dimensões: perda de dados, largura de banda e temporização.

Transferência confiável de dados

Algumas aplicações, como correio eletrônico, mensagem instantânea, transferência de arquivos, acesso a hospedeiros remotos, transferência de documentos Web e aplicações financeiras, exigem transferência de dados totalmente confiável, isto é, não pode haver perda de dados. Em particular, a perda de dados de arquivo ou de uma transação financeira pode ter conseqüências devastadoras (no último caso, tanto para o banco quanto para o cliente). Outras **aplicações tolerantes a perda**, mais notavelmente aplicações de multimídia, como áudio/vídeo em tempo real ou áudio/vídeo armazenados, podem tolerar uma certa perda de dados. Nessas últimas aplicações, dados perdidos podem resultar em uma pequena falha durante a execução do áudio/vídeo — o que não é um prejuízo crucial. Os efeitos dessa perda sobre a qualidade da aplicação e a quantidade real tolerável de perda de pacotes dependerão muito da aplicação e do esquema de codificação usado.

Largura de banda*

Algumas aplicações têm de transmitir dados a uma certa velocidade para serem efetivas. Por exemplo, se uma aplicação de telefonia por Internet codifica voz a 32 kbps, então também deve poder enviar dados para a rede e fazer com que sejam entregues na aplicação receptora a essa mesma taxa. Se essa largura de banda não estiver disponível, a aplicação precisará codificar a uma taxa diferente (e receber largura de banda suficiente para sustentar essa taxa de codificação diferente) ou então desistir, já que receber metade da largura de banda de que precisa de nada adianta para tal **aplicação sensível à largura de banda**. Muitas aplicações de multimídia existentes são sensíveis à largura de banda, mas as futuras poderão usar técnicas adaptativas de codi-

* Apesar de este não ser o termo mais preciso para definir o conceito apresentado, optou-se por respeitar a escolha do autor (*bandwidth*). (N. do R.T.)

ficação para codificar a uma taxa que corresponda à largura de banda disponível na ocasião. Embora aplicações sensíveis à largura de banda exijam uma dada quantidade de largura de banda, **aplicações elásticas** podem fazer uso de qualquer quantidade mínima ou máxima que por acaso esteja disponível. Correio eletrônico, transferência de arquivos e transferências Web são todas aplicações elásticas. Evidentemente, quanto mais largura de banda, melhor. Há um ditado que diz que "dinheiro nunca é demais"; nesse caso, podemos dizer que largura de banda nunca é demais!

Temporização

O requisito final de serviço é a temporização. Aplicações interativas em tempo real, tais como telefonia por Internet, ambientes virtuais, teleconferência e jogos multiusuários, exigem limitações estritas de temporização na entrega de dados para ser efetivas. Por exemplo, muitas dessas aplicações exigem que atrasos fim-a-fim sejam da ordem de poucas centenas de milissegundos ou menos. (Veja Capítulo 7, [Gauthier, 1999; Ramjee, 1994].) Longos atrasos na telefonia por Internet, por exemplo, tendem a resultar em pausas artificiais na conversação; em um jogo multiusuário ou ambiente virtual interativo, um longo atraso entre realizar uma ação e ver a reação do ambiente (por exemplo, a reação de um outro jogador na outra extremidade de uma conexão fim-a-fim) faz com que a aplicação pareça menos realista. Para aplicações que não são em tempo real, é sempre preferível um atraso menor a um maior, mas não há nenhuma limitação estrita aos atrasos fim-a-fim.

A Figura 2.4 resume os requisitos de confiabilidade, largura de banda e temporização para algumas aplicações populares e emergentes da Internet. Ela cita apenas alguns dos principais requisitos das aplicações mais populares da Internet. Nosso objetivo aqui não é dar uma classificação completa, mas simplesmente identificar algumas das diretrizes mais importantes segundo as quais os requisitos de uma aplicação de rede podem ser classificados.

2.1.5 Serviços providos pelos Protocolos de Transporte da Internet

A Internet (e, de maneira mais geral, as redes TCP/IP) disponibiliza dois protocolos de transporte para a aplicação: UDP e TCP. Quando um desenvolvedor cria uma nova aplicação para a Internet, uma das primeiras decisões que deve tomar é se usará UDP ou TCP. Cada um desses protocolos oferece um modelo de serviço diferente para as aplicações solicitantes.

Serviços do TCP

O modelo de serviço TCP inclui um serviço orientado para conexão e um serviço confiável de transferência de dados. Quando uma aplicação solicita o TCP como seu protocolo de transporte, recebe dele ambos os serviços.

Aplicação	Perda de dados	Largura de banda	Sensibilidade ao atraso
Transferência de arquivos	Sem perda	Elástica	Não
E-mail	Sem perda	Elástica	Não
Documentos Web	Sem perda	Elástica (alguns kbps)	Não
Áudio/vídeo em tempo real	Tolerante à perda	Áudio: alguns kbps – 1 Mbps Vídeo: 10 kbps – 5 Mbps	Sim: décimos de segundo
Áudio/vídeo armazenado	Tolerante à perda	Igual acima	Sim: alguns segundos
Jogos interativos	Tolerante à perda	Alguns kbps – 10 Mbps	Sim: décimos de segundo
Mensagem instantânea	Sem perda	Elástica	Sim e não

Figura 2.4 Requisitos de aplicações de rede selecionadas

- *Serviço orientado para conexão*: o TCP faz com que o cliente e o servidor troquem informações de controle de camada de transporte *antes* que as mensagens de camada de aplicação comecem a fluir. Esse procedimento de apresentação, por assim dizer, alerta o cliente e o servidor, permitindo que eles se preparem para uma enxurrada de pacotes. Após a fase de apresentação, dizemos que existe uma **conexão TCP** entre as portas dos dois processos. A conexão é *full-duplex* (simultânea), visto que os dois processos podem enviar mensagens um ao outro pela conexão ao mesmo tempo. Quando termina de enviar mensagens, a aplicação deve interromper a conexão. Esse serviço é chamado de serviço "orientado para conexão", e não serviço "de conexão", porque os dois processos estão conectados de um modo muito solto. No Capítulo 3, discutiremos detalhadamente serviço orientado para conexão e examinaremos como ele é implementado.
- *Serviço confiável de transporte*: os processos comunicantes podem confiar no TCP para a entrega de todos os dados enviados sem erro e na ordem correta. Quando um lado da aplicação passa uma cadeia de bytes para dentro de um socket, pode contar com o TCP para entregar a mesma cadeia de dados ao socket receptor, sem falta de bytes nem bytes duplicados.

O TCP também inclui um mecanismo de controle de congestionamento, um serviço voltado ao bem-estar geral da Internet e não ao benefício direto dos processos comunicantes. O mecanismo de controle de congestionamento do TCP limita a capacidade de transmissão de um processo (cliente ou servidor) quando a rede está congestionada entre cliente e servidor. Em particular, como veremos no Capítulo 3, o controle de congestionamento do TCP tenta limitar cada conexão do TCP à sua justa porção de largura de banda de rede.

A limitação da velocidade de transmissão pode ter um efeito muito prejudicial sobre aplicações de áudio e vídeo em tempo real que imponham uma limitação de largura de banda mínima. Além disso, aplicações em tempo real são tolerantes à perda e não necessitam de um serviço de transporte totalmente confiável. Por essas razões, desenvolvedores de aplicações em tempo real usualmente executam suas aplicações em UDP, e não em TCP.

Delineados os serviços providos pelo TCP, vamos falar um pouco dos serviços que o TCP *não* oferece. Em primeiro lugar, o TCP não garante uma taxa de transmissão mínima. Em particular, não é permitido a um processo originador transmitir com a taxa que bem entender; em vez disso, ela é regulada pelo controle de congestionamento do TCP, que pode forçar o remetente a enviar dados a uma taxa média baixa. Em segundo lugar, o TCP não oferece nenhuma garantia quanto a atrasos. Em particular, quando um processo originador passa dados para dentro de um socket, os dados cedo ou tarde chegarão ao processo receptor, mas o TCP não garante absolutamente nenhum limite de tempo para que eles cheguem lá. Como já aconteceu com muitos de nós, no caso da "espera mundial", às vezes ficamos esperando por dezenas de segundos ou até minutos para que o TCP entregue uma mensagem (contendo, por exemplo, um arquivo HTML) do servidor ao cliente Web. Em resumo, o TCP garante a entrega de todos os dados, mas não dá nenhuma garantia quanto à velocidade de entrega ou aos atrasos experimentados.

Serviços do UDP

O UDP é um protocolo de transporte simplificado, leve, com um modelo de serviço minimalista. É um serviço não orientado para conexão; portanto, não há apresentação antes que os dois processos comecem a se comunicar. O UDP provê um serviço não confiável de transferência de dados — isto é, quando um processo envia uma mensagem para dentro de um socket UDP, o protocolo não oferece *nenhuma* garantia de que a mensagem chegará ao processo receptor. Além do mais, mensagens que realmente chegam ao processo receptor podem chegar fora de ordem.

O UDP não inclui um mecanismo de controle de congestionamento; portanto, um processo originador pode bombear dados para dentro de um socket UDP à taxa que quiser (embora nem todos os dados consigam alcançar o socket receptor). Como aplicações em tempo real usualmente podem tolerar uma certa perda de dados, mas exigem uma taxa mínima, desenvolvedores desse tipo de aplicações freqüentemente preferem executá-las por UDP, evitando, assim, o controle de congestionamento e os cabeçalhos de pacotes do TCP. Similarmente ao TCP, o UDP não oferece nenhuma garantia quanto a atrasos.

Aplicações	Protocolo de camada de aplicação	Protocolo de transporte subjacente
Correio eletrônico	SMTP (RFC 2821)	TCP
Acesso a terminal remoto	Telnet (RFC 854)	TCP
Web	HTTP (RFC 2616)	TCP
Transferência de arquivos	FTP (RFC 959)	TCP
Servidor remoto de arquivos	NFS (McKusik, 1996)	UDP ou TCP
Multimídia em tempo real	Quase sempre proprietária (por exemplo, Real Networks)	UDP ou TCP
Telefonia por Internet	Quase sempre proprietária (por exemplo, Net2phone)	Tipicamente UDP

Figura 2.5 Aplicações populares da Internet, seus protocolos de camada de aplicação e seus protocolos de transporte subjacentes

A Figura 2.5 mostra os protocolos de transporte usados por algumas aplicações populares da Internet. Vemos que e-mail, a Web, acesso a terminais remotos e transferência de arquivos usam o TCP. Essas aplicações escolheram o TCP primordialmente porque ele oferece um serviço confiável de transferência de dados, garantindo que todos eles mais cedo ou mais tarde cheguem a seu destino. Vemos também que a aplicação de telefonia por Internet normalmente funciona em UDP. Cada lado de uma aplicação de telefone por Internet precisa enviar dados pela rede a uma taxa mínima (veja a Figura 2.4), o que é mais provavelmente possível com UDP do que com TCP. E, também, aplicações de telefone por Internet são tolerantes às perdas, de modo que não necessitam do serviço confiável de transferência de dados provido pelo TCP.

Como já observamos anteriormente, nem o TCP nem o UDP oferecem garantia de temporização. Isso significa que aplicações sensíveis a atrasos não podem ser executadas na Internet de hoje? A resposta é claramente "sim, elas podem" — a Internet abriga aplicações desse tipo há muitos anos. Essas aplicações em geral funcionam razoavelmente bem porque foram projetadas para enfrentar, até onde for possível, essa falta de garantia. Examinaremos diversos desses truques de projeto no Capítulo 7. Porém, mesmo o projeto inteligente tem suas limitações quando o atraso é excessivo, como freqüentemente é o caso da Internet pública. Em resumo, hoje, a Internet pode prover serviço satisfatório a aplicações sensíveis ao atraso, mas não pode oferecer nenhuma garantia de temporização ou de largura de banda. No Capítulo 7, discutiremos também modelos emergentes de serviços de Internet que oferecem novos serviços, inclusive serviço de garantia contra atrasos para aplicações em tempo real.

2.1.6 Aplicações de rede abordadas neste livro

Novas aplicações de Internet de domínio público e proprietárias são desenvolvidas todos os dias. Em vez de tratarmos de um grande número dessas aplicações de maneira enciclopédica, preferimos focalizar um pequeno número de aplicações ao mesmo tempo importantes e populares. Neste capítulo, discutiremos cinco aplicações populares: a Web, a transferência de arquivos, o correio eletrônico, o serviço de diretório e o compartilhamento de arquivos P2P. Discutiremos primeiramente a Web não somente porque ela é uma aplicação imensamente popular, mas também porque seu protocolo de camada de aplicação, HTTP, é direto e fácil de entender. Após examinarmos a Web, examinaremos brevemente o FTP porque este protocolo oferece um ótimo contraste com o HTTP. Em seguida, discutiremos o correio eletrônico, a primeira aplicação de enorme sucesso da Internet. O correio eletrônico é mais complexo do que a Web, no sentido de que usa não somente um, mas vários protocolos de camada de aplicação. Após o e-mail, estudaremos o DNS, que provê um serviço de diretório para a Internet. A maioria dos usuários não interage diretamente com o DNS; em vez disso, eles o chamam indiretamente por meio de outras aplicações (inclusive a Web, a transferência de arquivos e o correio eletrônico). O DNS ilustra primorosamente como um componente de funcionalidade de núcleo de rede (tradução de nome de rede para endereço de rede) pode ser implementado na camada de aplicação da Internet. Finalmente, discutiremos o compartilhamento de arquivos P2P que, segundo algumas medições (por exemplo, tráfego de rede), é a classe de aplicações mais popular da Internet de hoje.

2.2 A Web e o HTTP

Até a década de 1990, a Internet era usada primordialmente por pesquisadores, acadêmicos e estudantes universitários para se interligar com hospedeiros remotos, transferir arquivos de hospedeiros locais para hospedeiros remotos e vice-versa, enviar e receber notícias e enviar e receber correio eletrônico. Embora essas aplicações fossem (e continuem a ser) extremamente úteis, a Internet não era conhecida fora das comunidades acadêmicas e de pesquisa. Então, no início da década de 1990, entrou em cena uma nova aplicação importantíssima — a World Wide Web [Berners-Lee, 1994]. A Web é a aplicação da Internet que chamou a atenção do público em geral. Ela transformou drasticamente a maneira como pessoas interagem dentro e fora de seus ambientes de trabalho. Alçou a Internet de apenas mais uma entre muitas redes de dados (inclusive redes on-line como Prodigy, America Online e Compuserve, redes de dados de alcance nacional como Minitel/Transpac da França, a rede privada X.25 e redes frame relay) para, essencialmente, a única rede de dados.

A história está repleta de exemplos de lançamentos de tecnologias de comunicação eletrônica que provocaram enormes impactos sociais. A primeira dessas tecnologias foi o telefone, inventado na década de 1870. O telefone permitiu que duas pessoas se comunicassem oralmente em tempo real, sem que estivessem na mesma localização física. Outra tecnologia de comunicação eletrônica significativa, que provocou enorme impacto social, foi a transmissão de grande alcance de rádio/televisão, que surgiu no período 1920–1930. A transmissão de rádio/televisão permitiu que o público recebesse vastas quantidades de informação de áudio e vídeo. A Internet é mais uma tecnologia de comunicação eletrônica que causou um grande impacto social. O correio eletrônico e a Web, duas das aplicações mais populares, revolucionaram o modo como vivemos e trabalhamos. Discutiremos o e-mail na Seção 2.4. A Web e seu protocolo de camada de aplicação serão discutidos nesta seção.

Talvez o que mais atraia a maioria dos usuários da Web é que ela funciona *por demanda*. Usuários recebem o que querem, quando querem, o que é diferente da transmissão de rádio e de televisão, que força o usuário a sintonizar quando o provedor disponibiliza o conteúdo. Além de funcionar por demanda, a Web tem muitas outras características maravilhosas que as pessoas adoram. É muitíssimo fácil para qualquer indivíduo fazer com que informações fiquem disponíveis na Web; todo mundo pode se transformar em editor a um custo extremamente baixo. Hiperlinks e buscadores nos ajudam a navegar pelo oceano dos sites Web. Dispositivos gráficos estimulam nossos sentidos. Formulários, applets Java e muitos outros dispositivos nos habilitam a interagir com páginas e sites. E, cada vez mais, a Web oferece um menu de interfaces para vastas quantidades de material de vídeo e áudio armazenadas na Internet — áudio e vídeo que podem ser acessados por demanda.

2.2.1 Descrição geral do HTTP

O **HTTP — Protocolo de Transferência de Hipertexto** (HyperText Transfer Protocol) — protocolo de camada de aplicação da Web, está no coração da Web e é definido no [RFC 1945] e no [RFC 2616]. O HTTP é implementado em dois programas: um programa cliente e outro servidor. Os dois programas, executados em sistemas finais diferentes, conversam um com o outro por meio da troca de mensagens HTTP. O HTTP define a estrutura dessas mensagens e o modo como o cliente e o servidor as trocam. Antes de explicarmos detalhadamente o HTTP, devemos revisar a terminologia da Web.

Uma **página Web** (também denominada documento) é constituída de objetos. Um **objeto** é simplesmente um arquivo — tal como um arquivo HTML, uma imagem JPEG, uma imagem GIF, um applet Java, um clipe de áudio e assim por diante — que se pode acessar com um único URL. A maioria das páginas Web é constituída de um **arquivo-base HTML** e diversos objetos referenciados. Por exemplo, se uma página Web contiver um texto HTML e cinco imagens JPEG, então ela terá seis objetos: o arquivo-base HTML e mais as cinco imagens. O arquivo-base HTML referencia os outros objetos na página com os URLs dos

objetos. Cada URL tem dois componentes: o nome do hospedeiro do servidor que abriga o objeto e o nome do caminho do objeto. Por exemplo, no URL

```
http://www.someSchool.edu/someDepartment/picture.gif
```

`www.someSchool.edu` é o nome do hospedeiro e `/someDepartment/picture.gif` é o nome do caminho. Um **browser** é um agente de usuário para a Web; apresenta a página requisitada ao usuário e fornece numerosas características de navegação e de configuração. Como browsers Web também implementam o lado cliente do HTTP, podemos usar as palavras *browser* e *cliente* indiferentemente no contexto da Web. São browsers populares o Netscape Communicator e o Microsoft Internet Explorer. Um **servidor Web** abriga objetos Web, cada um endereçado por um URL. Os servidores Web também implementam o lado servidor do HTTP. Dentre os servidores populares da Web estão o Apache e o Microsoft Internet Information Server. (A Netcraft fornece um interessante levantamento sobre a aceitação dos servidores Web [Netcraft, 2000].)

O HTTP define como clientes Web (por exemplo, browsers) requisitam páginas Web aos servidores e como eles as transferem a clientes. Discutiremos detalhadamente a interação entre cliente e servidor mais adiante, mas a idéia geral está ilustrada na Figura 2.6. Quando um usuário requisita uma página Web (por exemplo, clica sobre um hiperlink), o browser envia ao servidor mensagens de requisição HTTP para os objetos da página. O servidor recebe as requisições e responde com mensagens de resposta HTTP que contêm os objetos. Até 1997, essencialmente todos os browsers e servidores Web implementavam a versão HTTP/1.0, que é definida no RFC 1945. A partir de 1998, servidores e browsers Web começaram a implementar a versão HTTP/1.1, definida no RFC 2616. O HTTP/1.1 é compatível com o HTTP/1.0; um servidor Web que executa a versão 1.1 pode 'conversar' com um browser que executa a versão 1.0, e um browser que executa a versão 1.1 pode 'conversar' com um servidor que executa a versão 1.0. Como, agora, o HTTP/1.1 é o protocolo dominante, daqui em diante quando nos referirmos a HTTP estaremos nos referindo a essa versão.

O HTTP usa o TCP como seu protocolo de transporte subjacente (em vez de rodar em UDP). O cliente HTTP primeiramente inicia uma conexão TCP com o servidor. Uma vez estabelecida a conexão, os processos do browser e do servidor acessam o TCP por meio de suas interfaces sockets. Como descrito na Seção 2.1, no lado cliente a interface socket é a porta entre o processo cliente e a conexão TCP; no lado servidor, ela é a porta entre o processo servidor e a conexão TCP.

O cliente envia mensagens de requisição HTTP para sua interface socket e recebe mensagens de resposta HTTP de sua interface socket. De maneira semelhante, o servidor HTTP recebe mensagens de requisição de sua interface socket e envia mensagens de resposta para sua interface socket. Assim que o cliente envia

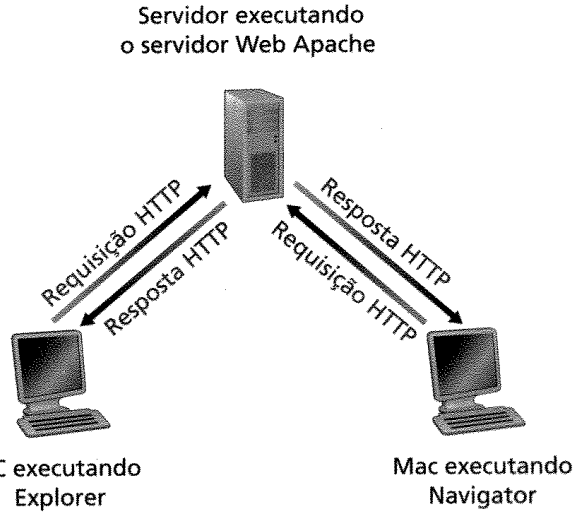


Figura 2.6 Comportamento de requisição-resposta do HTTP

uma mensagem para sua interface socket, a mensagem sai de suas mãos e "passa para as mãos do TCP". Lembre-se de que na Seção 2.1 dissemos que o TCP provê ao HTTP um serviço confiável de transferência de dados, o que implica que toda mensagem de requisição HTTP emitida por um processo cliente chegará intacta ao servidor. De maneira semelhante, toda mensagem de resposta HTTP emitida pelo processo servidor chegará intacta ao cliente. Percebemos, nesse ponto, uma das grandes vantagens de uma arquitetura de camadas — o HTTP não precisa se preocupar com dados perdidos ou com detalhes de como o TCP recupera a perda de dados ou os reordena dentro da rede. Esta é a tarefa do TCP e dos protocolos das camadas mais inferiores da pilha de protocolos.

É importante notar que o servidor envia ao cliente os arquivos solicitados sem armazenar nenhuma informação de estado sobre este. Se um determinado cliente solicita o mesmo objeto duas vezes em um período de poucos segundos, o servidor não responde dizendo que acabou de enviar o objeto ao cliente; em vez disso, envia novamente o objeto, pois já esqueceu completamente o que fez antes. Como o servidor HTTP não mantém nenhuma informação sobre clientes, o HTTP é denominado um **protocolo sem estado**.

Salientamos também que a Web usa a arquitetura de aplicação cliente-servidor, como descrito na Seção 2.1. Um servidor Web está sempre em funcionamento, tem um endereço IP fixo e atende requisições de potencialmente milhões de browsers diferentes.

2.2.2 Conexões persistentes e não persistentes

O HTTP pode usar conexões persistentes e não persistentes. Embora ele use conexões persistentes em seu modo default, os clientes e servidores também podem ser configurados para utilizar conexões não persistentes. Nesta subseção, examinaremos ambos os tipos de conexões.

Conexões não persistentes

Vamos percorrer as etapas da transferência de uma página Web de um servidor para um cliente para o caso de conexões não persistentes. Suponha que uma página consista em um arquivo-base HTML e em dez imagens JPEG e que todos esses 11 objetos residam no mesmo servidor. Suponha também que o URL para o arquivo-base HTTP seja

```
http://www.someSchool.edu/someDepartment/home.index
```

Eis o que acontece:

1. O processo cliente HTTP inicia uma conexão TCP para o servidor `www.someSchool.edu` na porta número 80, que é o número de porta default para o HTTP.
2. O cliente HTTP envia uma mensagem de requisição HTTP ao servidor através de seu socket associado com a conexão TCP. Essa mensagem inclui o nome de caminho `/someDepartment/home.index`. (Discutiremos mensagens HTTP mais detalhadamente logo adiante.)
3. O processo servidor HTTP recebe a mensagem de requisição através de seu socket associado à conexão, extrai o objeto `/someDepartment/home.index` de seu armazenamento (RAM ou disco), encapsula o objeto em uma mensagem de resposta HTTP e a envia ao cliente através de seu socket.
4. O processo servidor HTTP ordena ao TCP que encerre a conexão TCP. (Mas, na realidade, o TCP só encerrará a conexão quando tiver certeza de que o cliente recebeu a mensagem de resposta intacta.)
5. O cliente HTTP recebe a mensagem de resposta e a conexão TCP é encerrada. A mensagem indica que o objeto encapsulado é um arquivo HTML. O cliente extrai o arquivo da mensagem de resposta, analisa o arquivo HTML e encontra referências aos dez objetos JPEG.
6. As primeiras quatro etapas são repetidas para cada um dos objetos JPEG referenciados.

À medida que recebe a página Web, o browser a apresenta ao usuário. Dois browsers diferentes podem interpretar (isto é, exibir ao usuário) uma página Web de modos ligeiramente diferentes. O HTTP não tem nada a ver com o modo como uma página Web é interpretada por um cliente. As especificações do HTTP [RFC 1945] e [RFC 2616] definem apenas o protocolo de comunicação entre o programa cliente HTTP e o programa servidor HTTP.

As etapas apresentadas ilustram a utilização de conexões não persistentes, nas quais cada conexão TCP é encerrada após o servidor enviar o objeto — a conexão não persiste para outros objetos. Note que cada conexão TCP transporta exatamente uma mensagem de requisição e uma mensagem de resposta. Assim, nesse exemplo, quando um usuário solicita a página Web, são geradas 11 conexões TCP.

Nas etapas descritas, fomos intencionalmente vagos sobre se os clientes obtêm as dez JPEGs por meio de dez conexões TCP em série ou se algumas delas são recebidas por conexões TCP paralelas. Na verdade, usuários podem configurar browsers modernos para controlar o grau de paralelismo. Nos modos default, a maioria dos browsers abre de cinco a dez conexões TCP paralelas e cada uma delas manipula uma transação requisição/resposta. Se o usuário preferir, o número máximo de conexões paralelas poderá ser fixado em um, caso em que as dez conexões são estabelecidas em série. Como veremos no próximo capítulo, a utilização de conexões paralelas reduz o tempo de resposta.

Antes de continuarmos, vamos fazer um cálculo simples para estimar o tempo que transcorre entre a requisição e o recebimento de um arquivo-base HTTP por um cliente. Para essa finalidade, definimos o **tempo de viagem de ida e volta** (*round-trip time* — RTT), ou seja, o tempo que leva para um pequeno pacote viajar do cliente ao servidor e de volta ao cliente. O RTT inclui atrasos de propagação de pacotes, de fila de pacotes em roteadores e comutadores intermediários e de processamento de pacotes. (Esses atrasos foram discutidos na Seção 1.6.) Considere, agora, o que acontece quando um usuário clica sobre um hiperlink. Como ilustrado na Figura 2.7, isso faz com que o browser inicie uma conexão TCP entre ele e o servidor Web, o que envolve uma 'apresentação de três vias' — o cliente envia um pequeno segmento TCP ao servidor, o servidor o reconhece e responde com um pequeno segmento ao cliente que, por fim, o reconhece novamente para o servidor. As duas primeiras partes da apresentação de três vias representam um RTT. Após

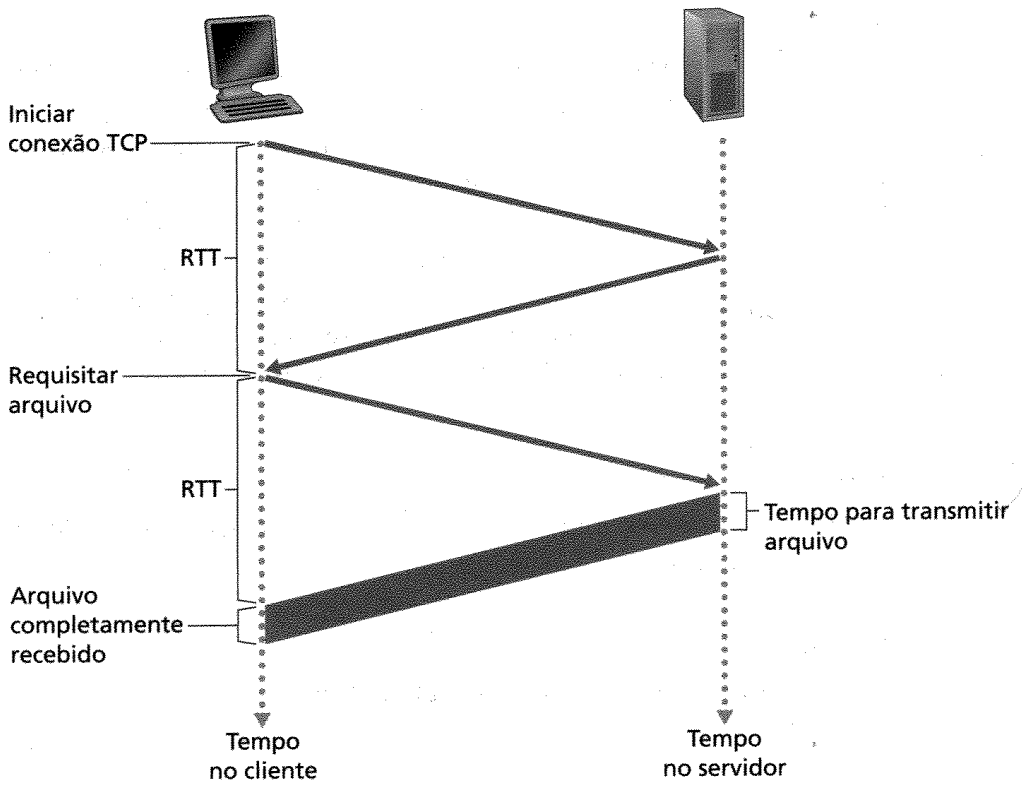


Figura 2.7 Cálculo simples para a requisição de um arquivo HTML

concluir essas partes, o cliente envia a mensagem de requisição HTTP combinada com a terceira parte da apresentação de três vias (o reconhecimento) para dentro da conexão TCP. Assim que a mensagem de requisição chega ao servidor, este envia o arquivo HTML por meio da conexão TCP. Essa requisição/resposta HTTP causa mais um RTT. Assim, aproximadamente, o tempo total de resposta são dois RTTs mais o tempo de transmissão do arquivo HTML no servidor.

Conexões persistentes

Conexões não persistentes têm algumas desvantagens. Em primeiro lugar, uma nova conexão deve ser estabelecida e mantida para *cada objeto solicitado*. Para cada uma dessas conexões, devem ser alocados buffers TCP e conservadas variáveis TCP tanto no cliente quanto no servidor. Isso pode sobrecarregar seriamente o servidor Web, que poderá estar processando requisições de centenas de diferentes clientes ao mesmo tempo. Em segundo lugar, como acabamos de descrever, cada objeto sofre dois RTTs — um RTT para estabelecer a conexão TCP e um RTT para solicitar e receber um objeto.

Em conexões persistentes, o servidor deixa a conexão TCP aberta após enviar resposta. Requisições e respostas subseqüentes entre os mesmos cliente e servidor podem ser enviadas por meio da mesma conexão. Em particular, uma página Web inteira (no exemplo anterior, o arquivo-base HTML e as dez imagens) pode ser enviada mediante uma única conexão TCP persistente. Além disso, várias páginas Web que residem no mesmo servidor podem ser enviadas por meio de uma única conexão TCP persistente. Normalmente, o servidor HTTP fecha uma conexão quando ela não é usada durante um certo tempo (um intervalo de pausa configurável).

Há duas versões de conexões persistentes: **sem paralelismo** e **com paralelismo**. Na versão sem paralelismo, o cliente emite uma nova requisição somente quando a resposta anterior foi recebida. Nesse caso, o cliente sofre um RTT para requisitar e receber cada um dos objetos referenciados (as dez imagens do exemplo acima). Embora isso seja uma melhoria em relação aos dois RTTs por objeto da conexão não persistente, o atraso de RTT pode ser reduzido ainda mais com paralelismo. Outra desvantagem de não se ter paralelismo é que, após o servidor enviar um objeto por meio de uma conexão TCP persistente, a conexão fica ociosa — não faz nada — enquanto espera a chegada de outra requisição. Essa ociosidade causa desperdício de recursos do servidor.

O modo default do HTTP usa conexões persistentes com paralelismo. Nesse caso, o cliente HTTP emite uma requisição logo que encontra uma referência. Assim, pode fazer requisições seqüenciais para os objetos relacionados, isto é, pode fazer uma nova requisição antes de receber uma resposta a uma requisição anterior. Quando o servidor recebe as requisições seqüenciais, envia os objetos seqüencialmente. Com paralelismo é possível gastar somente um RTT para todos os objetos referenciados (em vez de um RTT por objeto referenciado, como acontece quando não se utiliza paralelismo). E mais, a conexão TCP com paralelismo fica ociosa durante uma fração menor de tempo. Faremos uma comparação quantitativa entre os desempenhos de conexões persistentes e não persistentes nos exercícios de fixação dos capítulos 2 e 3. Aconselhamos o leitor interessado a consultar [Heidemann, 1997; Nielsen, 1997].

2.2.3 Formato da mensagem HTTP

As especificações do HTTP [RFC 2616] definem os formatos das mensagens HTTP. Há dois tipos de mensagens HTTP: de requisição e de resposta. Ambas serão discutidas a seguir.

Mensagem de requisição HTTP

Apresentamos a seguir uma mensagem de requisição HTTP típica:

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
```

```
Connection: close
User-agent: Mozilla/4.0
Accept-language: fr
```

Podemos aprender bastante examinando essa simples mensagem de requisição. Em primeiro lugar, vemos que ela está escrita em texto ASCII comum, de modo que pode ser lida por qualquer um que conheça computadores. Em segundo lugar, vemos que ela é constituída de cinco linhas, cada uma seguida de um 'carriage return' e 'line feed' (fim de linha) para o início de uma nova linha. A última linha é seguida de um comando adicional de 'carriage return' e 'line feed'. Embora essa mensagem de requisição específica tenha cinco linhas, uma mensagem de requisição pode ter muito mais do que isso e também menos do que isso, podendo conter até mesmo uma única linha. A primeira linha de uma mensagem de requisição HTTP é denominada **linha de requisição**; as subseqüentes são denominadas **linhas de cabeçalho**. A linha de requisição tem três campos: o campo do método, o do URL e o da versão do HTTP. O campo do método pode assumir vários valores diferentes, entre eles `GET`, `POST` e `HEAD`. A grande maioria das mensagens de requisição HTTP emprega o método `GET`, o qual é usado quando o browser requisita um objeto e este é identificado no campo do URL. Nesse exemplo, o browser está requisitando o objeto `/somedir/page.html`. A versão é auto-explicativa. Nesse exemplo, o browser implementa a versão HTTP/1.1.

Vamos agora examinar as linhas de cabeçalho do exemplo. A linha de cabeçalho `Host: www.someschool.edu` especifica o hospedeiro no qual o objeto reside. Talvez você ache que ela é desnecessária, pois já existe uma conexão TCP para o hospedeiro. Mas, como veremos na Seção 2.2.6, a informação fornecida pela linha de cabeçalho do hospedeiro é requerida por armazenadores intermediários da Web. Ao incluir a linha de cabeçalho `Connection: close`, o browser está dizendo ao servidor que não quer usar conexões persistentes; quer que o servidor feche a conexão após o envio do objeto requisitado. A linha de cabeçalho `User-agent:` especifica o agente de usuário, isto é, o tipo de browser que está fazendo a requisição ao servidor. No caso em questão, o agente de usuário é o Mozilla/4.0, um browser da Netscape. Essa linha de cabeçalho é útil porque, na verdade, o servidor pode enviar versões diferentes do mesmo objeto a tipos diferentes de agentes de usuário. (Cada uma das versões é endereçada pelo mesmo URL.) Por fim, o cabeçalho `Accept-language:` mostra que o usuário prefere receber uma versão em francês do objeto se esse objeto existir no servidor; se não existir, o servidor deve enviar a versão default. O cabeçalho `Accept-language:` é apenas um dos muitos cabeçalhos de negociação de conteúdo disponíveis no HTTP.

Acabamos de examinar um exemplo. Vamos agora analisar o formato geral de uma mensagem de requisição, ilustrado na Figura 2.8.

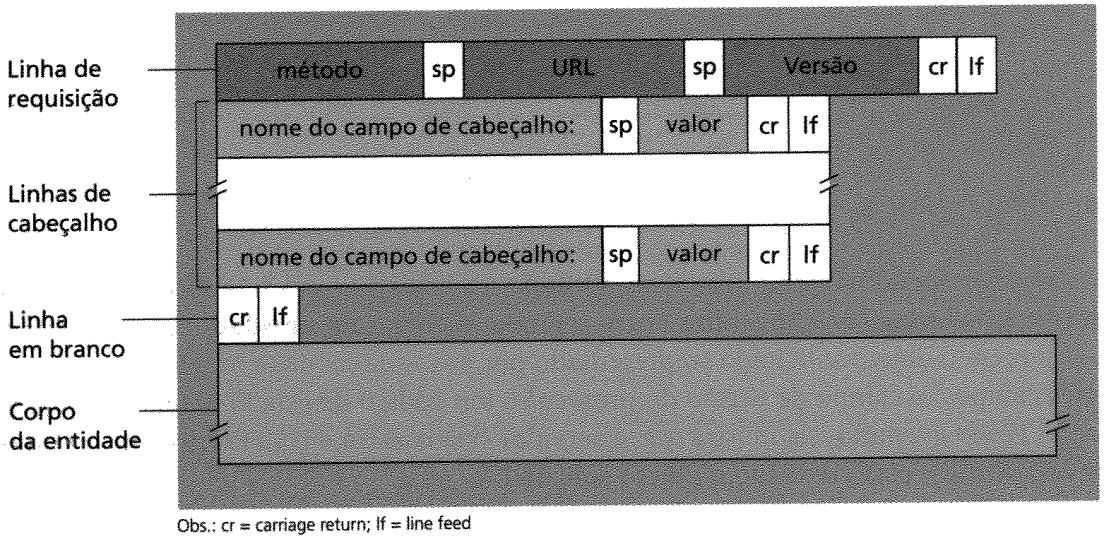


Figura 2.8 Formato geral de uma mensagem de requisição

Vemos que o formato geral de uma mensagem de requisição é muito parecido com nosso exemplo anterior. Contudo, você provavelmente notou que, após as linhas de cabeçalho (e após a linha adicional com 'carriage return' e 'line feed'), há um 'corpo de entidade'. O corpo de entidade fica vazio com o método GET, mas é utilizado com o método POST. Um cliente HTTP normalmente usa o método POST quando o usuário preenche um formulário — por exemplo, quando fornece palavras de busca a um site buscador. Com uma mensagem POST, o usuário continua solicitando uma página Web ao servidor, mas o conteúdo específico dela depende do que o usuário escreveu nos campos do formulário. Se o valor do campo de método for POST, então o corpo de entidade conterá o que o usuário digitou nos campos do formulário.

Seríamos omissos se não mencionássemos que uma requisição gerada com um formulário não utiliza necessariamente o método POST. Ao contrário, formulários HTML freqüentemente utilizam o método GET e incluem os dados digitados (nos campos do formulário) no URL requisitado. Por exemplo, se um formulário usar o método GET, tiver dois campos e as entradas desses dois campos forem monkeys e bananas, então a estrutura do URL será www.somesite.com/animalsearch?monkeys&bananas. Ao navegar normalmente pela Web, você provavelmente já notou URLs extensos como esse.

O método HEAD é semelhante ao GET. Quando um servidor recebe uma requisição com o método HEAD, responde com uma mensagem HTTP, mas deixa de fora o objeto requisitado. Esse método é freqüentemente usado pelos desenvolvedores de servidores HTTP para depuração.

A especificação HTTP/1.0 permite somente três tipos de métodos: GET, POST e HEAD. Além desses três métodos, a especificação HTTP/1.1 permite vários métodos adicionais, entre eles PUT e DELETE. O método PUT é freqüentemente usado juntamente com ferramentas de edição da Web. Permite que um usuário carregue um objeto para um caminho específico (diretório) em um servidor Web específico. O método PUT também é usado por aplicações que precisam carregar objetos para servidores Web. O método DELETE permite que um usuário, ou uma aplicação, elimine um objeto em um servidor Web.

Mensagem de resposta HTTP

Apresentamos a seguir uma mensagem de resposta HTTP típica. Essa mensagem poderia ser a resposta ao exemplo de mensagem de requisição que acabamos de discutir.

```
HTTP/1.1 200 OK
Connection: close
Date: Thu, 03 Jul 2003 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Sun, 5 May 2003 09:23:24 GMT
Content-Length: 6821
Content-Type: text/html

(data data data data data ...)
```

Vamos examinar cuidadosamente essa mensagem de resposta. Ela tem três seções: uma linha inicial, a **linha de estado**, seis **linhas de cabeçalho** e, em seguida, o **corpo da entidade**, que é o 'filé mignon' da mensagem — contém o objeto solicitado (representado por data data data data data ...). A linha de estado tem três campos: o campo de versão do protocolo, um código de estado e uma mensagem de estado correspondente. Nesse exemplo, ela mostra que o servidor está usando o HTTP/1.1 e que está tudo OK (isto é, o servidor encontrou e está enviando o objeto solicitado).

Agora, vamos examinar as linhas de cabeçalho. O servidor usa a linha de cabeçalho Connection: close para informar ao cliente que fechará a conexão TCP após enviar a mensagem. A linha de cabeçalho

`Date:` indica a hora e a data em que a resposta HTTP foi criada e enviada pelo servidor. Note que esse não é o horário em que o objeto foi criado nem o de sua modificação mais recente; é a hora em que o servidor extraiu o objeto de seu sistema de arquivos, inseriu-o na mensagem de resposta e enviou-a. A linha de cabeçalho `Server:` mostra que a mensagem foi gerada por um servidor Web Apache, análoga à linha de cabeçalho `User-agent:` na mensagem de requisição HTTP. A linha de cabeçalho `Last-Modified:` indica a hora e a data em que o objeto foi criado ou sofreu a última modificação. Este cabeçalho, que logo estudaremos mais detalhadamente, é fundamental para fazer cache do objeto, tanto no cliente local quanto em servidores de cache da rede (também conhecidos como servidores proxy). A linha de cabeçalho `Content-Length:` indica o número de bytes do objeto que está sendo enviado e a linha `Content-Type:` mostra que o objeto presente no corpo da mensagem é um texto HTML. (O tipo do objeto é oficialmente indicado pelo cabeçalho `Content-Type:`, e não pela extensão do arquivo.)

Acabamos de ver um exemplo. Vamos agora examinar o formato geral de uma mensagem de resposta, ilustrado na Figura 2.9.

Esse formato geral de mensagem de resposta condiz com o exemplo anterior. Mas falemos um pouco mais sobre códigos de estado e suas frases, que indicam o resultado da requisição. Eis alguns códigos de estado e frases associadas comuns:

- `200 OK:` requisição bem-sucedida e a informação é entregue com a resposta.
- `301 Moved Permanently:` objeto requisitado foi removido permanentemente; novo URL é especificado no cabeçalho `Location:` da mensagem de resposta. O software do cliente recuperará automaticamente o novo URL.
- `400 Bad Request:` código genérico de erro que indica que a requisição não pôde ser entendida pelo servidor.
- `404 Not Found:` o documento requisitado não existe no servidor.
- `505 HTTP Version Not Supported:` a versão do protocolo HTTP requisitada não é suportada pelo servidor.

Você gostaria de ver uma mensagem de resposta HTTP real? É muito recomendável e muito fácil! Primeiramente, dê um comando Telnet em seu servidor Web favorito. Em seguida, digite uma mensagem de requisição de uma linha solicitando algum objeto abrigado no servidor. Por exemplo, se você tem acesso a um prompt de comando, digite:

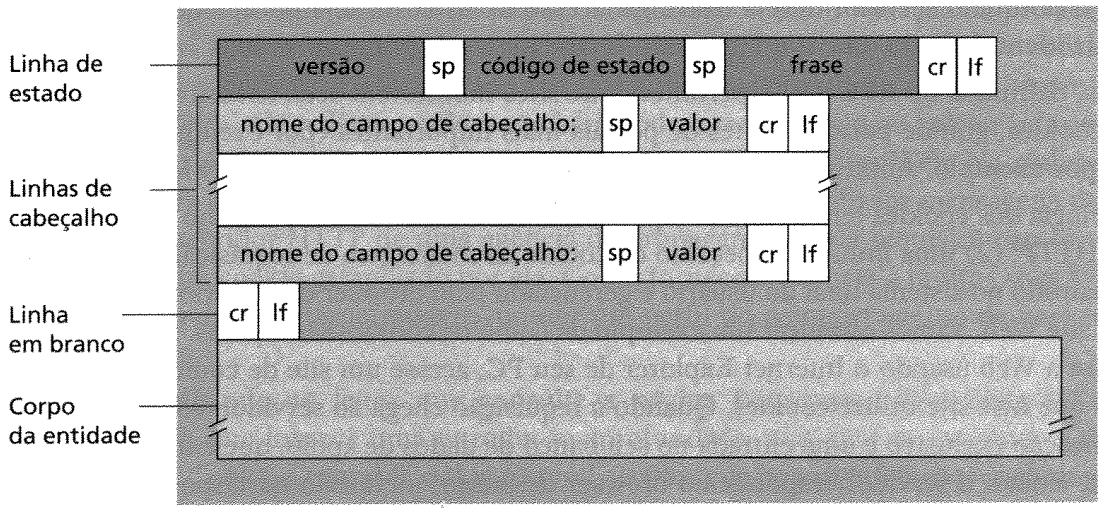


Figura 2.9 Formato geral de uma mensagem de resposta

```
telnet cis.poly.edu 80

GET /~ross/ HTTP/1.1
Host: cis.poly.edu
```

(Clique duas vezes a tecla 'enter' após digitar a última linha.) Essa seqüência de comandos abre uma conexão TCP para a porta número 80 do hospedeiro `cis.poly.edu` e, em seguida, envia a mensagem de requisição HTTP. Deverá aparecer uma mensagem de resposta que inclui o arquivo-base HTML da homepage do professor Ross. Se você preferir apenas ver as linhas da mensagem HTTP e não receber o objeto em si, substitua `GET` por `HEAD`. Finalmente, substitua `/~ross/` por `/~banana/` e veja que tipo de mensagem você obtém.

Nesta seção, discutimos várias linhas de cabeçalho que podem ser usadas em mensagens de requisição e de resposta HTTP. A especificação do HTTP define muitas outras linhas de cabeçalho que podem ser inseridas por browsers, servidores Web e servidores cache de redes. Vimos apenas um pouco do total de linhas de cabeçalho. Examinaremos mais algumas a seguir e mais um pouco quando discutirmos armazenagem Web na Seção 2.2.6. Uma discussão muito abrangente e fácil de ler sobre o protocolo HTTP, seus cabeçalhos e códigos de estado, pode ser encontrada em [Krishnamurty, 2001]; veja também [Luotonen, 1998] se estiver interessado no ponto de vista do desenvolvedor. Uma excelente introdução à Web pode ser encontrada em [Yeager, 1996].

Como um browser decide quais linhas de cabeçalho serão incluídas em uma mensagem de requisição? Como um servidor Web decide quais linhas de cabeçalho serão incluídas em uma mensagem de resposta? Um browser vai gerar linhas de cabeçalho em função de seu tipo e versão (por exemplo, um browser HTTP/1.0 não vai gerar nenhuma linha de cabeçalho 1.1), da configuração do usuário para o browser (por exemplo, idioma preferido) e de o browser ter uma versão do objeto possivelmente ultrapassada em sua memória. Servidores Web se comportam de maneira semelhante: há diferentes produtos, versões e configurações, e todos influenciam as linhas de cabeçalho que são incluídas nas mensagens de resposta.

2.2.4 Interação usuário-servidor: cookies

Mencionamos anteriormente que um servidor HTTP não tem estado, o que simplifica o projeto do servidor e vem permitindo que engenheiros desenvolvam servidores Web de alto desempenho que podem manipular milhares de conexões TCP simultâneas. No entanto, é sempre bom que um site Web identifique usuários, seja porque o servidor deseja restringir acesso de usuário, seja porque quer apresentar conteúdo em função da identidade do usuário. Para essas finalidades, o HTTP usa cookies.

Cookies, definidos no RFC 2109, permitem que sites monitorem seus usuários. Embora nem todos os sites usem cookies, grande parte dos sites de portais mais importantes (por exemplo, Yahoo), de comércio eletrônico (por exemplo, Amazon) e de propaganda (por exemplo, DoubleClick) faz uso intensivo deles.

A tecnologia dos cookies tem quatro componentes: (1) uma linha de cabeçalho de cookie na mensagem de resposta HTTP; (2) uma linha de cabeçalho de cookie na mensagem de requisição HTTP; (3) um arquivo de cookie mantido no sistema final do usuário e gerenciado pelo browser do usuário; (4) um banco de dados de apoio no site Web. Vamos esmiuçar um exemplo de como os cookies são usados. Suponha que Susan, que sempre acessa a Web usando o Internet Explorer de seu PC, acesse um site de comércio eletrônico pela primeira vez, e que esse site utilize cookies. Quando a requisição chega ao servidor Web, o site cria um número de identificação exclusivo e uma entrada no seu banco de dados de apoio, que é indexado pelo número de identificação. Então, o servidor responde ao browser de Susan, incluindo na resposta HTTP um cabeçalho `Set-cookie:` que contém o número de identificação. Por exemplo, a linha de cabeçalho poderia ser:

```
Set-cookie: 1678453
```

Quando recebe a mensagem de resposta HTTP, o browser de Susan vê o cabeçalho `Set-cookie:` e, então, anexa uma linha ao arquivo especial de cookies que ele gerencia. Essa linha inclui o nome de hospedeiro do servidor e seu número de identificação nesse cabeçalho. Toda vez que Susan requisita uma página Web enquanto navega pelo site de comércio eletrônico em questão, seu browser consulta seu arquivo de cookies, extrai seu número de identificação para esse site e insere na requisição HTTP uma linha de cabeçalho de cookie que inclui o número de identificação. Especificamente, cada uma de suas requisições HTTP ao servidor do site de comércio eletrônico inclui a linha de cabeçalho:

```
Cookie: 1678453
```

Dessa maneira, o site Web pode monitorar a atividade de Susan e, embora não saiba necessariamente que o nome dela é Susan, sabe exatamente quais páginas o usuário 1678453 visitou, em que ordem e em que horários! Então, pode utilizar cookies para oferecer um serviço de carrinho de compra — durante uma sessão particular, o site pode manter uma lista de todas as compras de Susan, de modo que ela possa pagar por todas elas ao mesmo tempo, no final da sessão.

Se Susan voltar ao site, digamos, uma semana depois, seu browser continuará a inserir a linha de cabeçalho `Cookie: 1678453` nas mensagens de requisição. O site de comércio eletrônico pode recomendar produtos a Susan com base nas páginas Web que ela visitou anteriormente. Se ela também se registrar no site — fornecendo seu nome completo, endereço de e-mail, endereço postal e informações de cartão de crédito — o site pode incluir essas informações em seu banco de dados e, assim, associar o nome de Susan com seu número de identificação (e com todas as páginas que ela consultou em suas visitas anteriores). É assim que alguns sites de comércio eletrônico oferecem "compras com um só clique" — quando quiser comprar um item durante uma visita subseqüente, Susan não precisará digitar novamente seu nome, número de seu cartão de crédito, nem endereço.

Essa discussão nos mostrou que cookies podem ser usados para identificar um usuário. Quando visitar um site pela primeira vez, um usuário pode fornecer dados de identificação (possivelmente seu nome). Então, o browser passa um cabeçalho de cookie ao servidor durante todas as visitas subseqüentes ao site, identificando, desse modo, o usuário ao servidor. A discussão também deixou claro que cookies podem ser usados para criar uma camada de sessão de usuário sobre HTTP sem estado. Por exemplo, quando um usuário acessa uma aplicação de e-mail baseada na Web, o browser envia informações de cookie ao servidor que, por sua vez, identifica o usuário por meio da sessão do usuário com a aplicação.

Embora cookies freqüentemente simplifiquem a experiência de compra pela Internet, continuam provocando muita controvérsia porque também podem ser considerados como violação da privacidade de um usuário. Como acabamos de ver, usando uma combinação de cookies e informações de conta fornecidas pelo usuário, um site Web pode ficar sabendo muita coisa sobre esse usuário e, potencialmente, vender o que sabe para algum terceiro. Além disso, cookies também podem ser usados para coletar informações sobre o comportamento de um determinado usuário em um grande número de sites Web. Na realidade, páginas Web que exibem banners de propaganda (os quais são GIFs ou PEGs) utilizam mensagens de requisição HTTP para obtê-los do servidor HTTP de uma agência de propaganda. Cada requisição ao servidor HTTP da agência pode conter um cookie gerenciado por ela. Como fornecem banners a muitos sites Web, as grandes agências de propaganda da Internet podem montar um perfil do comportamento de navegação de um indivíduo por vários sites.

Concluímos aconselhando ao leitor que consulte a página Persistent Client State HTTP Cookies [Netscape Cookie, 1999], que fornece uma introdução aprofundada, mas fácil de entender, sobre cookies. Recomendamos também Cookie Central [Cookie Central, 2004], que possui informações abrangentes sobre a controvérsia dos cookies.

2.2.5 Conteúdo HTTP

Por todo este capítulo, admitimos tacitamente que os dados carregados em mensagens de resposta HTTP são objetos de páginas Web, isto é, arquivos HTML, GIFs, JPEGs, applets Java e assim por diante. Apresentamos o HTTP no contexto da Web de modo a dar um exemplo utilizando uma aplicação concreta e familiar — a saber, a navegação pela Web. Porém, seríamos omissos se não mencionássemos que o HTTP é usado freqüentemente para transferir outros tipos de arquivos.

Atualmente, por exemplo, é muito usado em aplicações de comércio eletrônico para transferir arquivos XML de uma máquina para outra, sem que nenhuma dessas máquinas envolva um browser ou um usuário. Bancos freqüentemente usam XML para estruturar informações (como informações de contas de clientes) e máquinas bancárias utilizam freqüentemente HTTP para trocar entre si essas informações estruturadas em XML. (Discutir XML está fora do escopo deste livro. Diremos apenas que um documento XML típico provê dados estruturados e uma indicação do significado dos dados; não provê indicações de formatação, como o HTML.). O HTTP também é utilizado para transferir VoiceXML, WML (a linguagem de markup WAP) e outros tipos de documentos XML. Além do mais, como veremos no final deste capítulo, HTTP é freqüentemente usado como o protocolo de transferência de arquivos no compartilhamento de arquivos P2P. Veremos no Capítulo 7 que o HTTP é muito usado para recepção de conteúdo de áudio e vídeo armazenados.

2.2.6 Caches Web

Um **cache Web** — também denominado **servidor proxy** — é uma entidade da rede que atende requisições HTTP em nome de um servidor Web de origem. O cache Web tem seu próprio disco de armazenagem e mantém, dentro dele, cópias de objetos recentemente requisitados. Como ilustrado na Figura 2.10, o browser de um usuário pode ser configurado de modo que todas as suas requisições HTTP sejam dirigidas primeiramente ao cache Web. Uma vez que esteja configurado um browser, cada uma das requisições de um objeto que o browser faz é primeiramente dirigida ao cache Web. Como exemplo, suponha que um browser esteja requisitando o objeto `http://www.someschool.edu/campus.gif`. Eis o que acontece:

1. O browser estabelece uma conexão TCP com o cache Web e envia a ele uma requisição HTTP para um objeto.
2. O cache Web verifica se tem uma cópia do objeto armazenada localmente. Se tiver, envia o objeto ao browser do cliente, dentro de uma mensagem de resposta HTTP.
3. Se não tiver o objeto, o cache Web abre uma conexão TCP com o servidor de origem, isto é, com `www.someschool.edu`. Então, envia uma requisição HTTP do objeto para a conexão TCP. Após receber essa requisição, o servidor de origem envia o objeto ao cache Web, dentro de uma resposta HTTP.

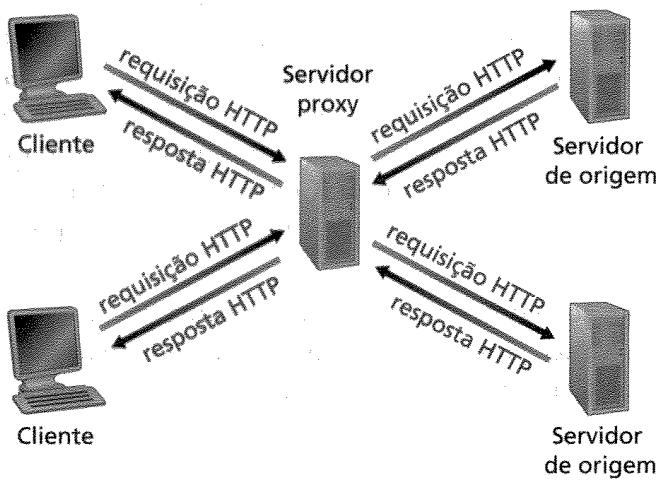


Figura 2.10 Clientes requisitando objetos por meio de um cache Web

4. Quando recebe o objeto, o cache Web guarda uma cópia em seu armazenamento local e envia outra, dentro de uma mensagem de resposta, ao browser do cliente (pela conexão TCP existente entre o browser do cliente e o cache Web).

Note que um cache é, ao mesmo tempo, um servidor e um cliente. Quando recebe requisições de um browser e lhe envia respostas, é um servidor. Quando envia requisições para um servidor de origem e recebe respostas dele, é um cliente.

Em geral, é um ISP que compra e instala um cache Web. Por exemplo, uma universidade poderia instalar um cache na rede de seu *campus* e configurar todos os browsers apontando para esse cache. Ou um importante ISP residencial (como a AOL) poderia instalar um ou mais caches em sua rede e configurar antecipadamente os browsers que fornece apontando para os caches instalados.

O cache na Web tem sido utilizado amplamente na Internet por duas razões. Em primeiro lugar, um cache Web pode reduzir substancialmente o tempo de resposta para a requisição de um cliente, em particular se o gargalo da largura de banda entre o cliente e o servidor de origem for muito menor do que aquele entre o cliente e o cache. Se houver uma conexão de alta velocidade entre o cliente e o cache, como em geral é o caso, e se este tiver o objeto requisitado, então ele poderá entregar rapidamente o objeto ao cliente. Em segundo lugar, como logo ilustraremos com um exemplo, caches Web podem reduzir substancialmente o tráfego no enlace de acesso de uma instituição qualquer à Internet. Com a redução do tráfego, a instituição (por exemplo, uma empresa ou uma universidade) não precisa ampliar sua largura de banda tão rapidamente, o que diminui os custos. Além disso, caches Web podem reduzir substancialmente o tráfego na Internet como um todo, melhorando, assim, o desempenho para todas as aplicações.

Para entender melhor os benefícios dos caches, vamos considerar um exemplo no contexto da Figura 2.11. Essa figura mostra duas redes: uma rede institucional e a Internet pública. A rede institucional é uma LAN de alta velocidade. Um roteador da rede institucional e um roteador da Internet estão ligados por um enlace de 1,5 Mbps. Os servidores de origem estão todos ligados à Internet, mas localizados pelo mundo todo. Suponha que o tamanho médio do objeto seja 100 kbits e que a taxa média de requisição dos browsers da instituição até os servidores de origem seja de 15 requisições por segundo. Imagine também que o tamanho das mensagens de requisição HTTP seja insignificante e, portanto, elas não criem tráfego nas redes ou no enlace de acesso (do roteador da instituição até o da Internet). Suponha ainda que o tempo entre o envio de uma requisição HTTP (dentro de um datagrama IP) pelo roteador do lado da Internet do enlace de acesso mostrado na Figura 2.11 e o recebimento da resposta (normalmente, dentro de muitos datagramas IPs) seja de 2 segundos em média. Esse último atraso é denominado informalmente "atraso da Internet".

O tempo de resposta total — isto é, aquele transcorrido entre a requisição de um objeto feita pelo browser e o recebimento dele — é a soma do atraso da LAN, do atraso de acesso (isto é, o atraso entre os dois roteadores) e do atraso da Internet. Vamos fazer agora um cálculo bastante rudimentar para estimar esse atraso. A intensidade de tráfego na LAN (veja a Seção 1.6) é

$$(15 \text{ requisições/segundo}) \cdot (100 \text{ kbits/requisição})/(10 \text{ Mbps}) = 0{,}15$$

ao passo que a intensidade de tráfego no enlace de acesso (do roteador da Internet ao da instituição) é

$$(15 \text{ requisições/segundo}) \cdot (100 \text{ kbits/requisição})/(1{,}5 \text{ Mbps}) = 1$$

Uma intensidade de tráfego de 0,15 em uma LAN resulta em, no máximo, dezenas de milissegundos de atraso; conseqüentemente, podemos desprezar o atraso da LAN. Contudo, como discutimos na Seção 1.6, à medida que a intensidade de tráfego se aproxima de 1 (como é o caso do enlace de acesso da Figura 2.11), o atraso em um enlace se torna muito grande e cresce sem limites. Assim, o tempo médio de resposta para atender requisições será da ordem de minutos, se não for maior, o que é inaceitável para os usuários da instituição. Evidentemente, algo precisa ser feito.

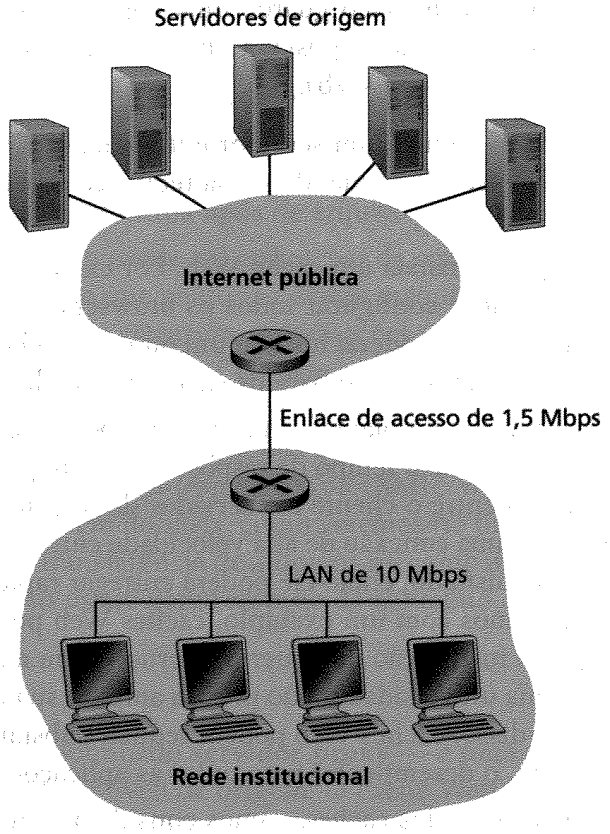


Figura 2.11 Gargalo entre uma rede institucional e a Internet

Uma possível solução seria aumentar a velocidade de acesso de 1,5 Mbps para, digamos, 10 Mbps. Isso reduziria a intensidade de tráfego no enlace de acesso a 0,15, o que se traduziria em atrasos desprezíveis entre os dois roteadores. Nesse caso, o tempo total de resposta seria aproximadamente 2 segundos, isto é, o atraso da Internet. Mas essa solução também significa que a instituição tem de atualizar seu enlace de acesso de 1,5 Mbps para 10 Mbps, o que pode ser muito dispendioso.

Considere agora a solução alternativa de não atualizar o enlace de acesso, mas, em vez disso, instalar um cache Web na rede institucional. Essa solução é ilustrada na Figura 2.12. As taxas de resposta local — a fração de requisições atendidas por um cache — em geral ficam na faixa de 0,2 a 0,7 na prática. Para demonstrarmos isso, vamos supor que a taxa de resposta local do cache dessa instituição seja 0,4. Como os clientes e o cache estão conectados à mesma LAN de alta velocidade, 40 por cento das requisições serão atendidas quase imediatamente, digamos, em 10 milissegundos. Mesmo assim, os demais 60 por cento das requisições ainda precisam ser atendidos pelos servidores de origem. Mas, com apenas 60 por cento dos objetos requisitados passando pelo enlace de acesso, a intensidade de tráfego neste diminui de 1,0 para 0,6. Em geral, uma intensidade de tráfego menor do que 0,8 corresponde a um atraso pequeno, digamos, dezenas de milissegundos, no caso de um enlace de 1,5 Mbps. Esse atraso é desprezível se comparado aos 2 segundos do atraso da Internet. Dadas essas considerações, o atraso médio é, por conseguinte,

$$0{,}4 \cdot (0{,}01 \text{ segundo}) + 0{,}6 \cdot (2{,}01 \text{ segundos}),$$

que é apenas ligeiramente maior do que 1,2 segundo. Assim, essa segunda solução resulta em tempo de resposta até menor do que o que se conseguiria com a troca do enlace de acesso e não requer que a instituição atualize seu enlace com a Internet. Evidentemente, a instituição terá de comprar e instalar um cache Web. Mas esse custo é baixo — muitos caches usam softwares de domínio público que rodam em PCs baratos.

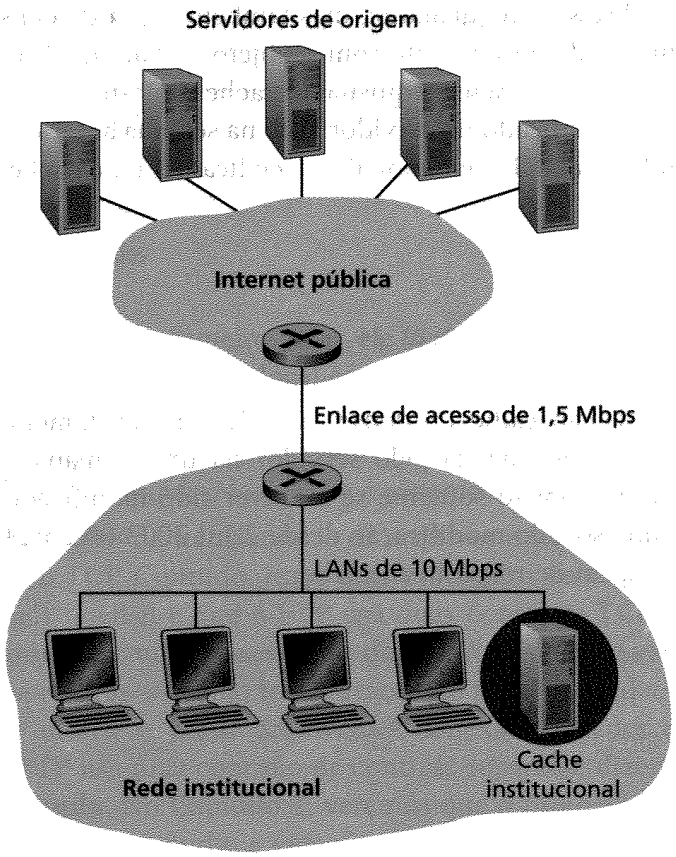


Figura 2.12 Adição de um cache à rede institucional

2.2.7 GET condicional

Embora possa reduzir os tempos de resposta do ponto de vista do usuário, fazer cache introduz um novo problema — a cópia de um objeto existente no cache pode estar desatualizada. Em outras palavras, o objeto abrigado no servidor Web pode ter sido modificado desde a data em que a cópia entrou no cache no cliente. Felizmente, o HTTP tem um mecanismo que permite que um cache verifique se seus objetos estão atualizados. Esse mecanismo é denominado **GET condicional** (*conditional GET*). Uma mensagem de requisição HTTP é denominada uma mensagem GET condicional se (1) usar o método `GET` e (2) possuir uma linha de cabeçalho `If-Modified-Since:`

Para ilustrar como o GET condicional funciona, vamos examinar um exemplo. Em primeiro lugar, um cache proxy envia uma mensagem de requisição a um servidor Web em nome de um browser requisitante:

```
GET /fruit/kiwi.gif HTTP/1.1
Host: www.exotiquecuisine.com
```

Em segundo lugar, o servidor Web envia ao cache uma mensagem de resposta com o objeto requisitado:

```
HTTP/1.1 200 OK
Date: Mon, 7 Jul 2003 15:39:29
Server: Apache/1.3.0 (Unix)
Last-Modified: Wed, 2 Jul 2003 09:23:24
Content-Type: image/gif

(data data data data data ...)
```

O cache passa o objeto ao browser requisitante, mas também o guarda em sua memória cache local. O importante é que ele também guarda, juntamente com o objeto, a data da última modificação. Em terceiro lugar, uma semana depois, um outro browser requisita ao cache o mesmo objeto, que ainda está no cache. Como esse objeto pode ter sido modificado no servidor Web na semana anterior, o browser realiza uma verificação de atualização emitindo um GET condicional. Especificamente, o cache envia:

```
GET /fruit/kiwi.gif HTTP/1.1
Host: www.exotiquecuisine.com
If-modified-since: Wed, 2 Jul 2003 09:23:24
```

Note que o valor da linha de cabeçalho `If-modified-since:` é exatamente igual ao valor da linha de cabeçalho `Last-Modified:`, que foi enviada pelo servidor há uma semana. Esse GET condicional está dizendo ao servidor para enviar o objeto somente se ele tiver sido modificado desde a data especificada. Suponha que o objeto não tenha sofrido modificação desde 2 Jul 2003 09:23:24. Então, em quarto lugar, o servidor Web envia uma mensagem de resposta ao cache:

```
HTTP/1.1 304 Not Modified
Date: Mon, 14 Jul 2003 15:39:29
Server: Apache/1.3.0 (Unix)

(corpo de mensagem vazio)
```

Vemos que, em resposta ao GET condicional, o servidor Web ainda envia uma mensagem de resposta, mas não inclui nela o objeto requisitado, o que apenas desperdiçaria largura de banda e aumentaria o tempo de resposta do ponto de vista do usuário, particularmente se o objeto fosse grande. Note que, na linha de estado dessa última mensagem de resposta está inserido `304 Not Modified`, que informa ao cache que ele pode seguir adiante e transmitir ao browser requisitante a cópia do objeto que está contida nele.

2.3 Transferência de arquivo: FTP

Em uma sessão FTP típica, o usuário, sentado à frente de um hospedeiro (o local), quer transferir arquivos de ou para um hospedeiro remoto. Para acessar a conta remota, o usuário deve fornecer uma identificação e uma senha. Após fornecer essas informações de autorização, pode transferir arquivos do sistema local de arquivos para o sistema remoto e vice-versa. Como mostra a Figura 2.13, o usuário interage com o FTP por

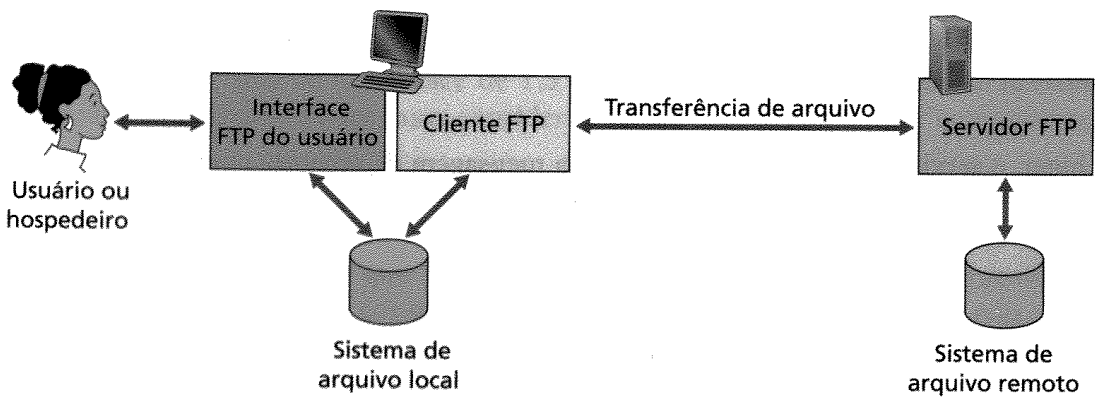


Figura 2.13 FTP transporta arquivos entre sistemas de arquivo local e remoto

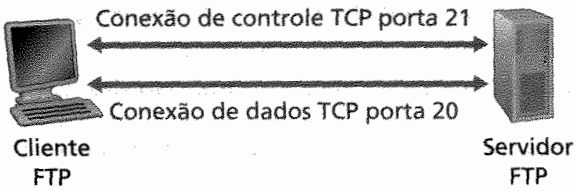


Figura 2.14 Conexões de controle e de dados

meio de um agente de usuário FTP. Em primeiro lugar, ele fornece o nome do hospedeiro remoto, o que faz com que o processo cliente FTP do hospedeiro local estabeleça uma conexão TCP com o processo servidor FTP do hospedeiro remoto. O usuário então fornece sua identificação e senha, que são enviadas pela conexão TCP como parte dos comandos FTP. Assim que autorizado pelo servidor, o usuário copia um ou mais arquivos armazenados no sistema de arquivo local para o sistema de arquivo remoto (ou vice-versa).

O HTTP e o FTP são protocolos de transferência de arquivos e têm muitas características em comum; por exemplo, ambos utilizam o TCP. Contudo, esses dois protocolos de camada de aplicação têm algumas diferenças importantes. A mais notável é que o FTP usa duas conexões TCP paralelas para transferir um arquivo: uma **conexão de controle** e uma **conexão de dados**. A primeira é usada para enviar informações de controle entre os dois hospedeiros — como identificação de usuário, senha, comandos para trocar diretório remoto e comandos de 'inserir' e 'pegar' arquivos. A conexão de dados é a usada para efetivamente enviar um arquivo. Como o FTP usa uma conexão de controle separada, dizemos que ele envia suas informações de controle **fora da banda**. No Capítulo 7, veremos que o protocolo RTSP, usado para controlar a transferência de meios contínuos como áudio e vídeo, também envia suas informações de controle fora da banda. O HTTP, como você recorda, envia linhas de cabeçalho de requisição e de resposta pela mesma conexão TCP que carrega o próprio arquivo transferido. Por essa razão, dizemos que o HTTP envia suas informações de controle **na banda**. Na próxima seção, veremos que o SMTP, o principal protocolo para correio eletrônico, também envia suas informações de controle na banda. As conexões de controle e de dados do FTP estão ilustradas na Figura 2.14.

Quando um usuário inicia uma sessão FTP com um hospedeiro remoto, o lado cliente do FTP (usuário) inicia primeiramente uma conexão TCP de controle com o lado servidor (hospedeiro remoto) na porta número 21 do servidor e envia por essa conexão de controle a identificação e a senha do usuário, além de comandos para mudar o diretório remoto. Quando o lado servidor recebe, pela conexão de controle, um comando para uma transferência de arquivo (de ou para o hospedeiro remoto), abre uma conexão TCP de dados para o lado cliente. O FTP envia exatamente um arquivo pela conexão de dados e em seguida fecha-a. Se, durante a mesma sessão, o usuário quiser transferir outro arquivo, o FTP abrirá outra conexão de dados. Assim, com FTP, a conexão de controle permanece aberta durante toda a sessão do usuário, mas uma nova conexão de dados é criada para cada arquivo transferido dentro de uma sessão (ou seja, a conexão de dados é não persistente).

Durante uma sessão, o servidor FTP deve manter informações de **estado** sobre o usuário. Em particular, o servidor deve associar a conexão de controle com uma conta de usuário específica e também deve monitorar o diretório corrente do usuário enquanto este passeia pela árvore do diretório remoto. Monitorar essas informações de estado para cada sessão de usuário em curso limita significativamente o número total de sessões que o FTP pode manter simultaneamente. Lembre-se de que o HTTP, por outro lado, é sem estado — não tem de monitorar o estado de nenhum usuário.

2.3.1 Comandos e respostas FTP

Encerraremos esta seção com uma breve discussão sobre alguns dos comandos mais comuns do FTP. Os comandos, do cliente para o servidor, e as respostas, do servidor para o cliente, são enviados por meio da conexão de controle no formato ASCII de 7 bits. Assim, tal como comandos HTTP, comandos FTP tam-

bém podem ser lidos pelas pessoas. Para separar comandos sucessivos, um 'carriage return' e um 'line feed' encerram cada um deles. Cada comando é constituído de quatro caracteres ASCII maiúsculos, alguns com argumentos opcionais. Alguns dos comandos mais comuns são descritos a seguir:

- `USER username`: usado para enviar identificação do usuário ao servidor.
- `PASS password`: usado para enviar a senha do usuário ao servidor.
- `LIST`: usado para pedir ao servidor que envie uma lista com todos os arquivos existentes no atual diretório remoto. A lista de arquivos é enviada por meio de uma conexão de dados (nova e não persistente), e não pela conexão TCP de controle.
- `RETR filename`: usado para extrair (isto é, obter) um arquivo do diretório atual do hospedeiro remoto. Ativa o hospedeiro remoto para que abra uma conexão de dados e envie o arquivo requisitado por essa conexão.
- `STOR filename`: usado para armazenar (isto é, inserir) um arquivo no diretório atual do hospedeiro remoto.

Há, particularmente, uma correspondência unívoca entre o comando que o usuário gera e o comando FTP enviado pela conexão de controle. Cada comando é seguido de uma resposta, que é enviada do servidor ao cliente. As respostas são números de três dígitos com uma mensagem opcional após o número. Elas se assemelham, em estrutura, à codificação de estado e à frase da linha de estado da mensagem de resposta HTTP. Os inventores do HTTP incluíram intencionalmente essa similaridade nas mensagens de resposta HTTP. Algumas respostas típicas, junto com suas possíveis mensagens, são as seguintes:

- `331 Nome de usuário OK, senha requisitada`
- `125 Conexão de dados já aberta; iniciando transferência`
- `425 Não é possível abrir a conexão de dados`
- `452 Erro ao escrever o arquivo`

Para saber mais sobre outros comandos e respostas FTP, o leitor interessado pode consultar o RFC 959.

2.4 Correio eletrônico na Internet

O correio eletrônico existe desde o início da Internet. Era uma das aplicações mais populares quando ela ainda estava na infância [Segaller, 1998], ficou mais e mais elaborado e poderoso ao longo dos anos e continua em evolução. Até agora, é uma das aplicações mais importantes e de maior sucesso da Internet.

Tal como o correio normal, o e-mail é um meio de comunicação assíncrono — as pessoas enviam e recebem mensagens quando for conveniente para elas, sem ter de estar coordenadas com o horário das outras pessoas. Ao contrário do correio normal, que anda a passos lentos, o correio eletrônico é rápido, fácil de distribuir e barato. O correio eletrônico moderno tem muitas características poderosas. Utilizando listas de mala direta, é possível enviar mensagens de e-mail, desejadas e indesejadas, a milhares de destinatários ao mesmo tempo. As mensagens do correio eletrônico moderno muitas vezes incluem anexos, hiperlinks, textos formatados em HTML e fotos. A grande maioria dessas mensagens é de texto, mas o correio eletrônico também pode ser usado como uma plataforma para transmissão assíncrona de voz e de mensagens por vídeo [Ross, 2003].

Nesta seção, examinaremos os protocolos de camada de aplicação que estão no coração do correio eletrônico da Internet. Mas, antes de entrarmos nessa discussão, vamos tomar uma visão geral do sistema de correio da Internet e de seus componentes principais.

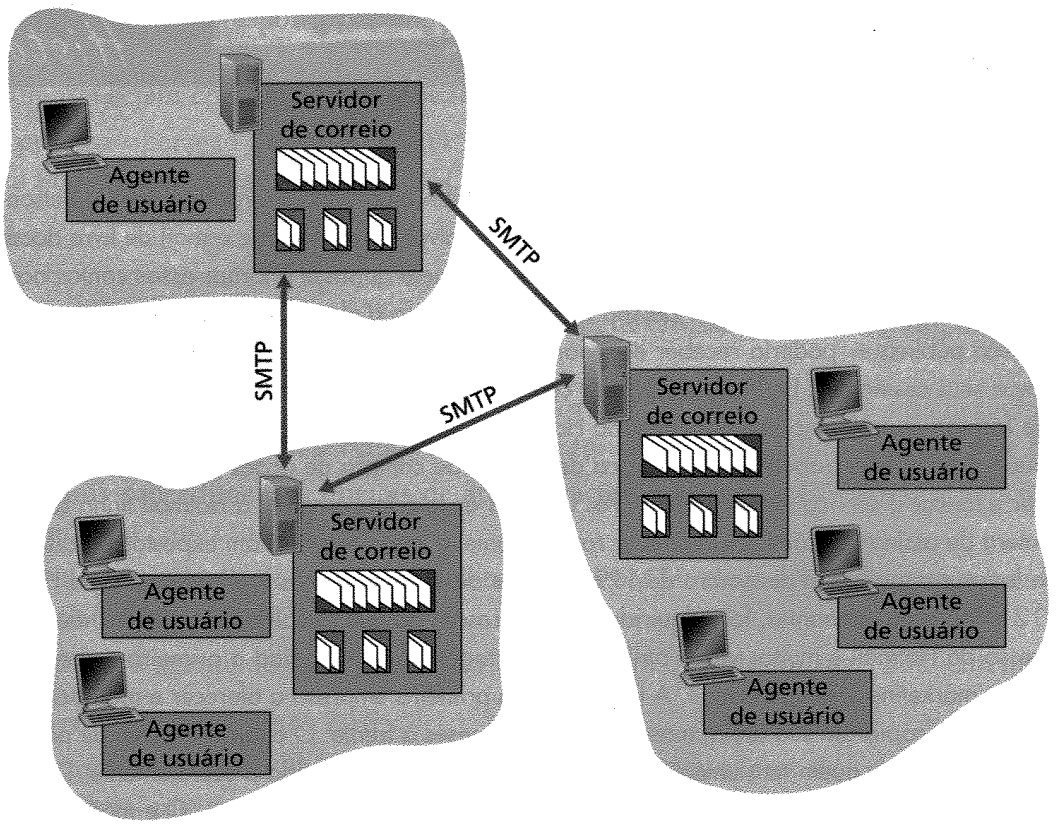


Figura 2.15 Uma visão do sistema de e-mail da Internet em analogia com a correspondência por correio

A Figura 2.15 apresenta uma visão do sistema de correio da Internet utilizando uma analogia com a correspondência por correio. Vemos, por esse diagrama, que há três componentes principais: **agentes de usuário**, **servidores de correio** e o **SMTP**. Descreveremos agora cada um desses componentes partindo do seguinte contexto: um remetente, Alice, está enviando uma mensagem de e-mail para um destinatário, Bob. Agentes de usuários permitem que usuários leiam, respondam, retransmitam, salvem e componham mensagens. (Às vezes, agentes de usuário de correio eletrônico são denominados *leitores de correio*, mas, neste livro, evitaremos essa expressão.) Quando Alice termina de compor sua mensagem, seu agente de usuário a envia a seu servidor de correio, onde ela é colocada na fila de saída de mensagens desse servidor. Quando Bob quer ler uma mensagem, seu agente de usuário a extrai da caixa de correio em seu servidor de correio. No final da década de 1990, agentes de usuário com interfaces gráficas de usuário (GUI) se tornaram populares, pois permitiam que usuários vissem e compusessem mensagens multimídia. Atualmente, o Eudora, o Outlook da Microsoft e o Messenger da Netscape são alguns dos agentes de usuário com interface gráfica populares para e-mail. Também há muitos agentes de usuário de texto de domínio público, entre eles 'mail', 'pine' e 'elm'.

Servidores de correio formam o núcleo da infra-estrutura do e-mail. Cada destinatário, como Bob, tem uma **caixa postal** localizada em um dos servidores de correio. A de Bob administra e guarda as mensagens que foram enviadas a ele. Uma mensagem típica inicia sua jornada no agente de usuário do remetente, vai até o servidor de correio dele e viaja até o servidor de correio do destinatário, onde é depositada na caixa postal. Quando Bob quer acessar as mensagens de sua caixa postal, o servidor de correio que contém sua caixa postal o autentica (com nome de usuário e senha). O servidor de correio de Alice também deve cuidar das falhas no servidor de correio de Bob. Se o servidor de correio dela não puder entregar a correspondência ao servidor dele, manterá a mensagem em uma **fila de mensagens** e tentará transferi-la mais tarde. Em geral, novas tentativas serão feitas a cada 30 minutos aproximadamente; se não obtiver sucesso após alguns dias, o servidor removerá a mensagem e notificará o remetente por meio de uma mensagem de correio.

História

HOTMAIL

Em dezembro de 1995, Sabeer Bhatia e Jack Smith fizeram uma visita a Draper Fisher Jurvetson, um investidor em empreendimentos de Internet, e propuseram desenvolver um sistema de e-mail de livre acesso baseado na Web. A idéia era oferecer uma conta de e-mail grátis a quem quisesse e tornar essas contas acessíveis pela Web. Com o e-mail pela Web, qualquer pessoa que tivesse acesso a ela — por exemplo, em uma escola ou em uma biblioteca pública — poderia receber e enviar e-mails. Além disso, o e-mail pela Web ofereceria grande mobilidade a seus assinantes. Em troca de 15 por cento da empresa, Draper Fisher Jurvetson financiou Bhatia e Smith, que formaram uma empresa denominada Hotmail. Com três funcionários em tempo integral e mais 12 a 14 em tempo parcial, que trabalhavam em troca de opções de compra de ações da empresa, eles conseguiram desenvolver e lançar o serviço em julho de 1996. Um mês após o lançamento, a Hotmail tinha cem mil assinantes. O número de assinantes começou a crescer rapidamente, e todos eles ficavam expostos a banners de propaganda ao ler seus e-mails. Em dezembro de 1997, menos de 18 meses após seu lançamento, a Hotmail, já com mais de 12 milhões de assinantes, foi adquirida pela Microsoft, ao que se saiba, por 400 milhões de dólares.

O sucesso da Hotmail é muitas vezes atribuído à vantagem de ela ter sido a primeira a entrar no mercado e ao inerente 'marketing viral' do e-mail. A Hotmail foi a primeira empresa a oferecer e-mail pela Web. Outras empresas, é claro, copiaram sua idéia, mas ela já estava meses à frente. O cobiçado posto de pioneiro no mercado é conquistado por quem tem uma idéia original e a desenvolve rápida e secretamente. Dizemos que um serviço ou produto tem 'marketing viral' quando ele mesmo promove seu marketing. O e-mail é o exemplo clássico de um serviço com marketing viral — o remetente envia uma mensagem a um ou mais destinatários, e com isso, estes passam a conhecer o serviço. A Hotmail demonstrou que a combinação da vantagem de ser o primeiro a chegar ao mercado com o marketing viral pode produzir uma aplicação absolutamente vitoriosa. Quem sabe alguns dos estudantes que estão lendo este livro estarão entre os novos empreendedores que conceberão e desenvolverão serviços de Internet pioneiros no mercado e com marketing viral?

O SMTP é o principal protocolo de camada de aplicação do correio eletrônico da Internet. Usa o serviço confiável de transferência de dados do TCP para transferir mensagens do servidor de correio do remetente para o do destinatário. Como acontece com a maioria dos protocolos de camada de aplicação, o SMTP tem dois lados: um lado cliente, que funciona no servidor de correio do remetente, e um lado servidor, que funciona no servidor de correio do destinatário. Ambos, o lado cliente e o lado servidor do SMTP, funcionam em todos os servidores de correio. Quando um servidor de correio envia correspondência para outros, age como um cliente SMTP. Quando o servidor de correio recebe correspondência de outros, age como um servidor SMTP.

2.4.1 SMTP

O SMTP, definido no RFC 2821, está no coração do correio eletrônico da Internet. Como mencionamos antes, esse protocolo transfere mensagens de servidores de correio remetentes para servidores de correio destinatários. O SMTP é muito mais antigo que o HTTP. (O RFC original do SMTP data de 1982, e ele já existia muito antes disso.) Embora tenha inúmeras qualidades maravilhosas, como evidencia sua ubiqüidade na Internet, o SMTP é uma tecnologia antiga que possui certas características arcaicas. Por exemplo, restringe o corpo (e não apenas o cabeçalho) de todas as mensagens de correio ao simples formato ASCII de 7 bits. Essa restrição tinha sentido no começo da década de 1980, quando a capacidade de transmissão era escassa e ninguém enviava correio eletrônico com anexos volumosos nem arquivos grandes com imagens, áudio ou vídeo. Mas, hoje em dia, na era da multimídia, a restrição do ASCII de 7 bits é um tanto incômoda — exige que os dados binários de multimídia sejam codificados em ASCII antes de ser enviados pelo

SMTP e que a mensagem correspondente em ASCII seja decodificada novamente para o sistema binário depois do transporte pelo SMTP. Lembre-se da Seção 2.2, na qual dissemos que o HTTP não exige que os dados de multimídia sejam codificados em ASCII antes da transferência.

Para ilustrar essa operação básica do SMTP, vamos percorrer um cenário comum. Suponha que Alice queira enviar a Bob uma simples mensagem ASCII:

1. Alice chama seu agente de usuário para e-mail, fornece o endereço de Bob (por exemplo, `bob@someschool.edu`), compõe uma mensagem e instrui o agente de usuário a enviar a mensagem.
2. O agente de usuário de Alice envia a mensagem para seu servidor de correio, onde ela é colocada em uma fila de mensagens.
3. O lado cliente do SMTP, que funciona no servidor de correio de Alice, vê a mensagem na fila e abre uma conexão TCP para um servidor SMTP, que funciona no servidor de correio de Bob.
4. Após alguns procedimentos iniciais de apresentação, o cliente SMTP envia a mensagem de Alice para dentro da conexão TCP.
5. No servidor de correio de Bob, o lado servidor do SMTP recebe a mensagem e a coloca na caixa postal dele.
6. Bob chama seu agente de usuário para ler a mensagem quando for mais conveniente para ele.

Esse cenário está resumido na Figura 2.16.

É importante observar que o SMTP normalmente não usa servidores de correio intermediários para enviar correspondência, mesmo quando os dois servidores estão localizados em lados opostos do mundo. Se o servidor de Alice está em Hong Kong, e o de Bob, em St. Louis, a conexão TCP é uma conexão direta entre os servidores em Hong Kong e St. Louis. Em particular, se o servidor de correio de Bob não estiver em funcionamento, a mensagem permanece no servidor de correio de Alice esperando por uma nova tentativa — a mensagem não é colocada em nenhum servidor de correio intermediário.

Vamos agora examinar mais de perto como o SMTP transfere uma mensagem de um servidor de correio remetente para um servidor de correio destinatário. Veremos que o protocolo SMTP tem muitas semelhanças com protocolos usados na interação humana cara a cara. Primeiramente, o cliente SMTP (que funciona no hospedeiro do servidor de correio remetente) faz com que o TCP estabeleça uma conexão na porta 25 com o servidor SMTP (que funciona no hospedeiro do servidor de correio destinatário). Se o servidor não estiver em funcionamento, o cliente tenta novamente mais tarde. Uma vez estabelecida a conexão, o servidor e o cliente trocam alguns procedimentos de apresentação de camada de aplicação — exatamente como os seres humanos, que freqüentemente se apresentam antes de transferir informações, clientes e servidores SMTP também se apresentam antes de transferir informações. Durante essa fase, o cliente SMTP indica os endereços de e-mail do remetente (a pessoa que gerou a mensagem) e do destinatário. Assim que o cliente e o servidor SMTP terminam de se apresentar, o cliente envia a mensagem. O SMTP pode

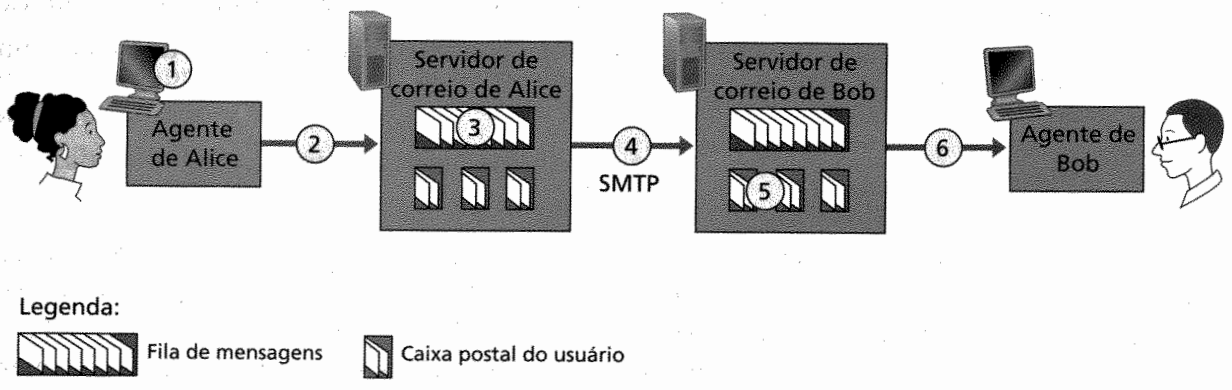


Figura 2.16 Alice envia uma mensagem a Bob

contar com o serviço confiável de transferência de dados do TCP para entregar a mensagem ao servidor sem erros. Então, o cliente repetirá esse processo, na mesma conexão TCP, se houver outras mensagens a enviar ao servidor; caso contrário, dará uma instrução ao TCP para encerrar a conexão.

Vamos examinar um exemplo de troca de mensagens entre um cliente (C) e um servidor SMTP (S). O nome do hospedeiro do cliente é `crepes.fr` e o nome do hospedeiro do servidor é `hamburger.edu`. As linhas de texto ASCII iniciadas com `C:` são exatamente as linhas que o cliente envia para dentro de seu socket TCP e as iniciadas com `S:` são exatamente as linhas que o servidor envia para dentro de seu socket TCP. A transcrição a seguir começa assim que a conexão TCP é estabelecida:

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250 Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr ... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

Nesse exemplo, o cliente enviou uma mensagem ("`Do you like ketchup? How about pickles?`") do servidor de correio `crepes.fr` ao servidor de correio `hamburger.edu`. Como parte do diálogo, o cliente emitiu cinco comandos: `HELO` (uma abreviação de HELLO), `MAIL FROM`, `RCPT TO`, `DATA` e `QUIT`. Esses comandos são auto-explicativos. O cliente também enviou uma linha consistindo em um único ponto final, que indica o final da mensagem para o servidor. (No jargão ASCII, cada mensagem termina com `CRLF.CRLF`, onde `CR` significa 'carriage return' e `LF` significa 'line feed'). O servidor emite respostas a cada comando, e cada resposta tem uma codificação de resposta e algumas explicações (opcionais) em inglês. Mencionamos aqui que o SMTP usa conexões persistentes: se o servidor de correio remetente tiver diversas mensagens para enviar ao mesmo servidor de correio destinatário, poderá enviar todas pela mesma conexão TCP. Para cada mensagem, o cliente inicia o processo com um novo `MAIL FROM: crepes.fr`, indica o final da mensagem com um ponto final isolado e emite QUIT somente após todas as mensagens terem sido enviadas.

Recomendamos veementemente que você utilize o Telnet para executar um diálogo direto com um servidor SMTP. Para fazer isso digite

```
telnet serverName 25
```

em que `serverName` é o nome de um servidor de correio local. Ao fazer isso, você está simplesmente estabelecendo uma conexão TCP entre seu hospedeiro local e o servidor de correio. Após digitar essa linha, você deverá receber imediatamente do servidor a resposta 220. Digite, então, os comandos `HELO`, `MAIL FROM`, `RCPT TO`, `DATA`, `CRLF.CRLF` e `QUIT` nos momentos apropriados. Também recomendamos veementemente que você faça a Tarefa de Programação 2 no final deste capítulo. Nessa tarefa, você construirá um agente de

usuário simples que implementa o lado cliente do SMTP. Esse agente permitirá que você envie uma mensagem de e-mail a um destinatário qualquer, por meio de um servidor de correio local.

2.4.2 Comparação com o HTTP

Agora, vamos fazer uma breve comparação entre o SMTP e o HTTP. Ambos os protocolos são usados para transferir arquivos de um hospedeiro para outro. O HTTP transfere arquivos (também denominados objetos) de um servidor Web para um cliente Web (normalmente um browser). O SMTP transfere arquivos (isto é, mensagens de e-mail) de um servidor de correio para outro. Ao transferir os arquivos, o HTTP persistente e o SMTP usam conexões persistentes. Assim, os dois protocolos têm características em comum. Existem, todavia, diferenças importantes. A primeira é que o HTTP é, principalmente, um **protocolo de recuperação de informações** (*pull protocol*) — alguém carrega informações em um servidor Web e os usuários utilizam o HTTP para recuperá-las do servidor quando quiserem. Em particular, a conexão TCP é ativada pela máquina que quer receber o arquivo. O SMTP, por sua vez, é, primordialmente, um **protocolo de envio de informações** (*push protocol*) — o servidor de correio remetente envia o arquivo para o servidor de correio destinatário. Em particular, a conexão TCP é ativada pela máquina que quer enviar o arquivo.

A segunda diferença, à qual aludimos anteriormente, é que o SMTP exige que cada mensagem, inclusive o corpo, esteja no formato ASCII de 7 bits. Se a mensagem contiver caracteres que não estejam nesse formato (por exemplo, caracteres em francês, com acento) ou dados binários (como um arquivo de imagem), terá de ser codificada em ASCII de 7 bits. Dados HTTP não impõem esta restrição.

A terceira diferença importante refere-se ao modo como um documento que contém texto e imagem (juntamente com outros tipos possíveis de mídia) é manipulado. Como vimos na Seção 2.2, o HTTP encapsula cada objeto em sua própria mensagem HTTP. O correio pela Internet, como discutiremos com maiores detalhes mais adiante, coloca todos os objetos de mensagem em um única mensagem.

2.4.3 Formatos de mensagem de correio e MIME

Quando Alice escreve uma carta a Bob e a envia pelo correio normal, ela pode incluir todos os tipos de informações periféricas no cabeçalho da carta, como seu próprio endereço, o endereço de Bob e a data. De modo semelhante, quando uma mensagem de e-mail é enviada, um cabeçalho contendo informações periféricas antecede o corpo da mensagem em si. Essas informações periféricas estão contidas em uma série de linhas de cabeçalho definidas no RFC 822. As linhas de cabeçalho e o corpo da mensagem são separados por uma linha em branco (isto é, por `CRLF`). O RFC 822 especifica o formato exato das linhas de cabeçalho das mensagens, bem como suas interpretações semânticas. Como acontece com o HTTP, cada linha de cabeçalho contém um texto legível, consistindo em uma palavra-chave seguida de dois-pontos e de um valor. Algumas palavras-chave são obrigatórias e outras, opcionais. Cada cabeçalho deve ter uma linha de cabeçalho `From:` e uma `To:` e pode incluir também uma `Subject:`, bem como outras opcionais. É importante notar que essas linhas de cabeçalho são *diferentes* dos comandos SMTP que estudamos na Seção 2.4.1 (ainda que contenham algumas palavras em comum, como '*from*' e '*to*'). Os comandos daquela seção faziam parte do protocolo de apresentação SMTP; as linhas de cabeçalho examinadas nesta seção fazem parte da própria mensagem de correio.

Um cabeçalho de mensagem típico é semelhante a:

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Searching for the meaning of life.
```

Após o cabeçalho da mensagem, vem uma linha em branco e, em seguida, o corpo da mensagem (em ASCII). Você pode usar o Telnet para enviar a um servidor de correio uma mensagem que contenha algumas linhas de cabeçalho, inclusive `Subject:`. Para tal, utilize o comando `telnet serverName 25`.

A extensão MIME para dados que não seguem o padrão ASCII

Conquanto os cabeçalhos descritos no RFC 822 sejam satisfatórios para enviar textos comuns em padrão ASCII, não são suficientemente ricos para mensagens multimídia (por exemplo, mensagens com imagens, áudio e vídeo) ou para transmitir textos em formatos que não seguem o padrão ASCII (por exemplo, caracteres usados em outros idiomas que não o inglês). Para enviar conteúdo que não seja texto ASCII, o agente de usuário remetente deve incluir cabeçalhos adicionais na mensagem. Esses cabeçalhos extras são definidos no RFC 2045 e no RFC 2046, que são as extensões do RFC 822 referentes ao MIME (*Multipurpose Internet Mail Extensions* — extensões multiuso do correio da Internet).

Os dois cabeçalhos MIME fundamentais para suporte de multimídia são `Content-Type:` e `Content-Transfer-Encoding:`. O cabeçalho `Content-Type:` permite que o agente de usuário destinatário realize uma ação adequada sobre a mensagem. Por exemplo, ao indicar que o corpo da mensagem contém uma imagem JPEG, o agente de usuário destinatário pode dirigir o corpo da mensagem para uma rotina de descompressão de JPEG. Para entender a necessidade do cabeçalho `Content-Transfer-Encoding:`, lembre-se de que textos de mensagem que não seguem o padrão ASCII devem ser codificados em um formato ASCII que não confundirá o SMTP. O cabeçalho `Content-Transfer-Encoding:` alerta o agente de usuário destinatário que o corpo da mensagem foi codificado em ASCII e informa o tipo de codificação usada. Assim, quando um agente de usuário recebe uma mensagem com esses dois cabeçalhos, primeiramente usa o valor do cabeçalho `Content-Transfer-Encoding:` para converter o corpo da mensagem à sua forma original, diferente de ASCII e, em seguida, usa o cabeçalho `Content-Type:` para determinar que ações deve realizar no corpo da mensagem.

Vamos examinar um exemplo concreto. Suponha que Alice queira enviar a Bob uma imagem JPEG. Para fazer isso, ela chama seu agente de usuário de e-mail, especifica o endereço de e-mail de Bob e o assunto da mensagem e insere a imagem JPEG no campo denominado corpo da mensagem. (Dependendo do agente de usuário que usa, Alice pode inserir a imagem na mensagem como um anexo.) Quando termina de compor sua mensagem, ela clica no comando 'Send' ('Enviar'). O agente de usuário de Alice então gera uma mensagem MIME, que se parece com:

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Transfer-Encoding: base64.
Content-Type: image/jpeg

(base64 encoded data .....
.....
.....base64 encoded data)
```

Observamos nessa mensagem MIME que o agente de usuário de Alice codificou a imagem JPEG usando codificação base64. Essa é uma das diversas técnicas de codificação padronizadas no MIME [RFC 2045] para conversão para um formato ASCII de 7 bits aceitável. Outra técnica popular é a codificação de transferência de conteúdo 'quoted-printable', que normalmente é usada para converter uma mensagem ASCII de 8 bits (possivelmente contendo caracteres em outro idioma que não o inglês) para ASCII de 7 bits.

Quando Bob ler sua correspondência com seu agente de usuário, este vai agir sobre a mesma mensagem MIME e, quando observar a linha de cabeçalho `Content-Transfer-Encoding: base64`, procederá à decodificação do corpo da mensagem codificado em base64. A mensagem também inclui uma linha de cabeçalho `Content-Type: image/jpeg`, que indica ao agente de usuário de Bob que o corpo da mensagem deve ser tratado por um decodificador JPEG. Por fim, inclui o cabeçalho `MIME-Version:`, que indica, é claro, a versão de MIME que está sendo usada. Note que, exceto por isso, a mensagem segue o formato padrão RFC 822/SMTP. Em particular, após o cabeçalho da mensagem, há uma linha em branco e, em seguida, o corpo da mensagem.

O cabeçalho Received

Seríamos omissos se não mencionássemos uma outra classe de linhas de cabeçalho que são inseridas pelo servidor SMTP *destinatário*. Ao receber a mensagem com os cabeçalhos RFC 822 e MIME, o servidor destinatário anexa uma linha de cabeçalho `Received:` ao topo da mensagem. Essa linha de cabeçalho especifica o nome do servidor SMTP que enviou a mensagem ('from'), o do que recebeu a mensagem ('by') e o horário em que o servidor destinatário recebeu a mensagem. Assim, vista pelo usuário destinatário, a mensagem toma a seguinte forma:

```
Received: from crepes.fr by hamburger.edu; 12 Oct 98 15:27:39 GMT
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data .....
.....
.....base64 encoded data
```

Quase todos os que usaram correio eletrônico já viram a linha de cabeçalho `Received:` (juntamente com outras linhas de cabeçalho) que precede mensagens de e-mail. (Essa linha muitas vezes é vista diretamente na tela ou quando a mensagem é enviada a uma impressora.) Talvez você tenha notado que, às vezes, uma única mensagem tem várias linhas de cabeçalho `Received:` e uma linha `Return-Path:` (trajeto de volta) mais complexa. Isso acontece porque uma mensagem pode ser transmitida para mais de um servidor SMTP no caminho entre o remetente e o destinatário. Por exemplo, se Bob deu uma instrução a seu servidor de e-mail `hamburger.edu` para transmitir todas as suas mensagens para `sushi.jp`, então a mensagem lida pelo agente de usuário de Bob começaria com algo parecido com:

```
Received: from hamburger.edu by sushi.jp; 3 Jul 01 15:30:01 GMT
Received: from crepes.fr by hamburger.edu; 3 Jul 01 15:17:39 GMT
```

Essas linhas de cabeçalho dão ao agente de usuário destinatário um diagnóstico dos servidores SMTP visitados, bem como os horários em que as visitas ocorreram.

2.4.4 Protocolos de acesso ao correio

Quando o SMTP entrega a mensagem do servidor de correio de Alice ao servidor de correio de Bob, ela é colocada na caixa postal de Bob. Durante toda essa discussão, ficou tacitamente subentendido que Bob lê sua correspondência ao entrar no hospedeiro servidor e, em seguida, executa o leitor de correio que roda naquela máquina. Até o início da década de 1990, este era o modo padronizado de acessar o correio. Mas hoje o acesso ao correio usa uma arquitetura cliente–servidor — o usuário típico lê e-mails com um cliente que funciona em seu sistema final, por exemplo, um PC no escritório, um laptop ou um PDA. Quando executam um cliente de correio em PC local, usuários desfrutam de uma série de propriedades, inclusive a capacidade de ver mensagens e anexos multimídia.

Dado que Bob (o destinatário) executa seu agente de usuário em seu PC local, é natural que ele considere a instalação de um servidor de correio nessa máquina. Adotando essa abordagem, o servidor de correio de Alice dialogaria diretamente com o PC de Bob. Porém, há um problema com essa abordagem. Lembre-se de que um servidor de correio gerencia caixas postais e executa os lados cliente e servidor do SMTP. Se o servidor de correio de Bob residisse em seu PC local, este teria de ficar sempre em funcionamento e ligado na Internet para poder receber novas correspondências que poderiam chegar a qualquer hora, o que não é prático para muitos usuários da Internet. Em vez disso, um usuário típico executa um agente de usuário no PC local, mas acessa sua caixa postal armazenada em um servidor de correio compartilhado que está sempre em funcionamento. Esse servidor de correio é compartilhado com outros usuários e, em geral, é mantido pelo ISP do usuário (por exemplo, universidade ou empresa).

Agora, vamos considerar o caminho que uma mensagem percorre quando é enviada de Alice para Bob. Acabamos de aprender que, em algum ponto ao longo do caminho, a mensagem de e-mail precisa ser depositada no servidor de correio de Bob. Essa tarefa poderia ser realizada simplesmente fazendo com que o agente de usuário de Alice enviasse a mensagem diretamente ao servidor de correio de Bob, o que pode ser feito com o SMTP — realmente, o SMTP foi projetado para enviar e-mail de um hospedeiro para outro. Contudo, normalmente o agente de usuário do remetente não dialoga diretamente com o servidor de correio do destinatário. Em vez disso, como mostra a Figura 2.17, o agente de usuário de Alice usa SMTP para enviar a mensagem de e-mail a seu servidor de correio. Em seguida, este servidor usa SMTP (como um cliente SMTP) para retransmitir a mensagem de e-mail para o servidor de correio de Bob. Por que esse procedimento em duas etapas? Primordialmente porque, sem a retransmissão através do servidor de correio de Alice, o agente de usuário dela não dispõe de nenhum recurso para um servidor de correio de destinatário que não pode ser alcançado. Fazendo com que Alice primeiramente deposite o e-mail em seu próprio servidor de correio, este pode tentar, várias vezes, enviar a mensagem ao servidor de correio de Bob, digamos, a cada 30 minutos, até que esse servidor entre em operação. (E, se o servidor de correio de Alice não estiver funcionando, ela terá o recurso de se queixar ao administrador do seu sistema!) O RFC do SMTP define como os comandos do SMTP podem ser usados para retransmitir uma mensagem por vários servidores SMTP.

Mas ainda falta uma peça do quebra-cabeça! De que forma um destinatário como Bob, que executa um agente de usuário em seu PC local, obtém suas mensagens que estão em um servidor de correio dentro do seu ISP? Note que o agente de usuário de Bob não pode usar SMTP para obter as mensagens porque essa operação é de recuperação (*pull*), e o SMTP é um protocolo de envio (*push*). O quebra-cabeça é concluído com a introdução de um protocolo especial de acesso ao correio que transfere mensagens do servidor de correio de Bob para seu PC local. Atualmente, há vários protocolos populares de acesso a correio, entre eles **POP3** (Post Office Protocol versão 3), **IMAP** (Internet Mail Access Protocol) e **HTTP**.

A Figura 2.17 apresenta um resumo dos protocolos usados no correio da Internet: o SMTP é utilizado para transferir correspondência do servidor de correio remetente para o servidor de correio destinatário; também é usado para transferir correspondência do agente de usuário remetente para o servidor de correio remetente. Um protocolo de acesso de correio, como o POP3, é usado para transferir correspondência do servidor de correio destinatário para o agente de usuário destinatário.

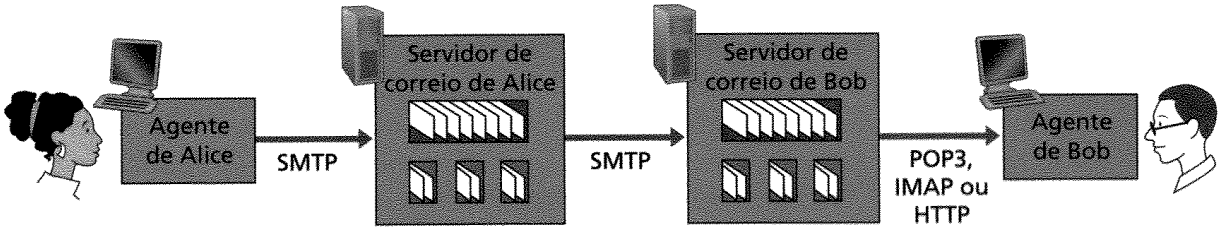


Figura 2.17 Protocolos de e-mail e suas entidades comunicantes

POP3

O POP3 é um protocolo de acesso de correio extremamente simples. É definido no RFC 1939, que é curto e bem fácil de ler. Por esse protocolo ser tão simples, sua funcionalidade é bastante limitada. O POP3 começa quando o agente de usuário (o cliente) abre uma conexão TCP com o servidor de correio (o servidor) na porta 110. Com a conexão TCP ativada, o protocolo passa por três fases: autorização, transação e atualização. Durante a primeira fase, autorização, o agente de usuário envia um nome de usuário e uma senha (às claras) para autenticar o usuário. Na segunda fase, transação, recupera mensagens; é também nessa fase que o agente de usuário pode marcar mensagens que devem ser apagadas, remover essas marcas e obter estatísticas de correio. A terceira fase, atualização, ocorre após o cliente ter dado o comando `quit` que encerra a sessão POP3. Nesse momento, o servidor de correio apaga as mensagens que foram marcadas.

Em uma transação POP3, o agente de usuário emite comandos e o servidor, uma resposta para cada um deles. Há duas respostas possíveis: `+OK` (às vezes seguida de dados do servidor para o cliente), usada pelo servidor para indicar que correu tudo bem com o comando anterior e `-ERR`, que o servidor usa para informar que houve algo errado com o comando anterior.

A fase de autorização tem dois comandos principais: `user <user name>` e `pass <password>`. Para ilustrar esses dois comandos, sugerimos que você realize uma sessão Telnet diretamente com um servidor POP3, usando a porta 110, e emita os dois comandos. Suponha que `mailServer` seja o nome de seu servidor de correio. O que você verá será algo parecido com:

```
telnet mailServer 110
+OK POP3 server ready
user bob
+OK
pass hungry
+OK user successfully logged on
```

Se você escrever um comando errado, o POP3 responderá com uma mensagem `-ERR`.

Agora, vamos examinar a fase de transação. Um agente de usuário que utiliza POP3 freqüentemente pode ser configurado (pelo usuário) para 'ler-e-apagar' ou para 'ler-e-guardar'. A seqüência de comandos emitida por um agente de usuário POP3 depende do modo em que o agente de usuário estiver operando. No modo ler-e-apagar, o agente de usuário emite os comandos `list`, `retr` e `dele`. Como exemplo, suponha que o usuário tenha duas mensagens em sua caixa postal. No diálogo abaixo, `C:` (que representa o cliente) é o agente de usuário e `S:` (que representa o servidor), o servidor de correio. A transação será mais ou menos assim:

```
C: list
S: 1 498
S: 2 912
```

```
S: .
C: retr 1
S: (blah blah ...
S: ..................
S: ..........blah)
S: .
C: dele 1
C: retr 2
S: (blah blah ...
S: ..................
S: ..........blah)
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

O agente de usuário primeiramente pede ao servidor de correio que apresente o tamanho de cada uma das mensagens armazenadas. Então, ele recupera e apaga cada mensagem do servidor. Note que, após a fase de autorização, o agente de usuário empregou apenas quatro comandos: `list`, `retr`, `dele` e `quit`. A sintaxe para esses comandos é definida no RFC 1939. Depois de processar o comando de saída (*quit*), o servidor POP3 entra na fase de atualização e remove as mensagens 1 e 2 da caixa postal.

Há um problema com o modo ler-e-apagar: o destinatário, Bob, pode ser nômade e querer acessar seu correio de muitas máquinas, por exemplo, do PC de seu escritório, do PC de sua casa e de seu computador portátil. O modo ler-e-apagar reparte as mensagens de correio de Bob entre essas três máquinas; em particular, se ele ler primeiramente uma mensagem no PC de seu escritório, não poderá lê-la novamente mais tarde em seu computador portátil. No modo ler-e-guardar, o agente de usuário deixa as mensagens no servidor de correio após descarregá-las. Nesse caso, Bob pode reler mensagens em máquinas diferentes; pode acessar uma mensagem em seu local de trabalho e, uma semana depois, acessá-la novamente em casa.

Durante uma sessão POP3 entre um agente de usuário e o servidor de correio, o servidor POP3 mantém alguma informação de estado; em particular, monitora as mensagens do usuário marcadas para apagar. Contudo, não mantém informação de estado entre sessões POP3. Essa falta de informação simplifica a implementação de um servidor POP3.

IMAP

Usando POP3, assim que baixar suas mensagens na máquina local, Bob pode criar pastas de correspondência e transferir as mensagens baixadas para elas. Em seguida, pode apagar as mensagens, mudá-las de pastas e procurar mensagens (por nome de remetente ou assunto). Mas esse paradigma — pastas e mensagens na máquina local — apresenta um problema para o usuário nômade que gostaria de manter uma hierarquia de pastas em um servidor remoto que possa ser acessado de qualquer computador: com o POP3, isso não é possível. Esse protocolo não provê nenhum meio para um usuário criar pastas remotas e designar mensagens a pastas.

Para resolver esse e outros problemas, foi inventado o protocolo IMAP, definido no RFC 2060. Como o POP3, o IMAP é um protocolo de acesso a correio, porém com mais recursos, mas é também significativamente mais complexo. (E, portanto, também as implementações dos lados cliente e servidor são significativamente mais complexas.)

Um servidor IMAP associa cada mensagem a uma pasta. Quando uma mensagem chega a um servidor pela primeira vez, é associada com a pasta INBOX do destinatário, que, então, pode transferir a mensagem para uma nova pasta criada por ele, lê-la, apagá-la e assim por diante. O protocolo IMAP provê comandos que permitem que os usuários criem pastas e transfiram mensagens de uma para outra. O protocolo também provê comandos que os usuários podem usar para pesquisar pastas remotas em busca de mensagens que obedeçam a critérios específicos. Note que, diferentemente do POP3, um servidor IMAP mantém informação de estado de usuário entre sessões IMAP — por exemplo, os nomes das pastas e quais mensagens estão associadas a elas.

Outra característica importante do IMAP é que ele tem comandos que permitem que um agente de usuário obtenha componentes de mensagens. Por exemplo, um agente de usuário pode obter apenas o cabeçalho ou somente uma das partes de uma mensagem MIME multiparte. Essa característica é útil quando há uma conexão de largura de banda estreita entre o agente de usuário e seu servidor de correio, como, por exemplo, uma conexão de modem sem fio ou de baixa velocidade. Com uma conexão de pequena largura de banda, o usuário pode decidir não baixar todas as mensagens de sua caixa postal, evitando, em particular, mensagens longas que possam conter, por exemplo, um clipe de áudio ou de vídeo. Se quiser ler tudo sobre o IMAP, consulte o site oficial [IMAP, 2004].

E-mail pela Web

Hoje, um número cada vez maior de usuários está enviando e acessando e-mails por meio de seus browsers Web. O Hotmail lançou o e-mail com acesso pela Web em meados da década de 1990; agora, esse tipo de acesso é provido por praticamente todos os sites ISP, bem como universidades e empresas importantes. Com esse serviço, o agente de usuário é um browser Web comum e o usuário se comunica com sua caixa postal remota via HTTP. Quando um destinatário, por exemplo, Bob, quer acessar uma mensagem em sua caixa postal, ela é enviada do servidor de correio para o browser de Bob usando o protocolo HTTP, e não os protocolos POP3 ou IMAP. Quando um remetente, por exemplo, Alice, quer enviar uma mensagem de e-mail, esta é enviada do browser de Alice para seu servidor de correio por HTTP, e não por SMTP. O servidor de correio de Alice, contudo, ainda envia mensagens para outros servidores de correio e recebe mensagens de outros servidores de correio usando o SMTP.

Essa solução para o acesso ao correio é extremamente conveniente quando o usuário está em trânsito. Para enviar e receber mensagens, basta que ele possa acessar um browser. Esse browser pode estar em um cibercafé, na casa de um amigo, em um PDA, em um quarto de hotel que tenha Web TV e assim por diante. Como ocorre com o IMAP, usuários podem organizar suas mensagens em uma hierarquia de pastas no servidor remoto. Na verdade, muitas implementações de e-mail pela Web utilizam um servidor IMAP para prover a funcionalidade de pastas. Nesse caso, o acesso às pastas e mensagens é provido por scripts que rodam em um servidor HTTP e usam o protocolo IMAP para se comunicar com um servidor IMAP.

2.5 DNS: o serviço de diretório da Internet

Nós, seres humanos, podemos ser identificados por diversas maneiras. Por exemplo, podemos ser identificados pelo nome que aparece em nossa certidão de nascimento, pelo número do RG ou da carteira de motorista. Embora cada um desses números possa ser usado para identificar pessoas, em um dado contexto um pode ser mais adequado que outro. Por exemplo, os computadores da Receita Federal preferem usar o número do CPF (de tamanho fixo) ao nome que consta em nossa certidão de nascimento. Por outro lado, pessoas comuns preferem nosso nome de batismo, mais fácil de lembrar, ao número do CPF. (Realmente, você pode se imaginar dizendo: "Oi, meu nome é 360.039.012-68. Este é meu marido, 086.154.758-69"?)

Assim como seres humanos podem ser identificados de muitas maneiras, exatamente o mesmo acontece com hospedeiros da Internet. Um identificador é seu nome de **hospedeiro** (*hostname*). Nomes de hospedeiro — como `cnn.com`, `www.yahoo.com`, `gaia.cs.umass.edu` e `cis.poly.edu` — são fáceis de lembrar e,

portanto, apreciados pelos seres humanos. Todavia, eles fornecem pouca, se é que alguma, informação sobre a localização de um hospedeiro na Internet. (Um nome como `www.eurecom.fr`, que termina com o código do país `.fr`, nos informa que o hospedeiro provavelmente está na França, mas não diz muito mais do que isso.) Além disso, como nomes de hospedeiros podem consistir em caracteres alfanuméricos de comprimento variável, seriam difíceis de ser processados por roteadores. Por essas razões, hospedeiros também são identificados pelo que denominamos **endereços IP**. Discutiremos endereços IP mais detalhadamente no Capítulo 4, mas é importante falar um pouco sobre eles agora. Um endereço IP é constituído de 4 bytes e sua estrutura hierárquica é rígida. Ele é semelhante a `121.7.106.83`, no qual cada ponto separa um dos bytes expressos em notação decimal de 0 a 255. Um endereço IP é hierárquico porque, ao examiná-lo da esquerda para a direita, obtemos gradativamente mais informações específicas sobre onde o hospedeiro está localizado na Internet (isto é, em qual rede da rede de redes). De maneira semelhante, quando examinamos um endereço postal de cima para baixo, obtemos informações cada vez mais específicas sobre a localização do destinatário.

2.5.1 Serviços fornecidos pelo DNS

Acabamos de ver que há duas maneiras de identificar um hospedeiro — por um nome de hospedeiro e por um endereço IP. As pessoas preferem o identificador nome de hospedeiro por ser mais fácil de lembrar, ao passo que roteadores preferem endereços IP de comprimento fixo e estruturados hierarquicamente. Para conciliar essas preferências, é necessário um serviço de diretório que traduza nomes de hospedeiro para endereços IP. Esta é a tarefa principal do `DNS` (*domain name system* — **sistema de nomes de domínio**) da Internet. O DNS é (1) um banco de dados distribuído implementado em uma hierarquia de `servidores de nome` (**servidores DNS**), e (2) um protocolo de camada de aplicação que permite que hospedeiros consultem o banco de dados distribuído. Os servidores de nome são freqüentemente máquinas UNIX que executam o software BIND (Berkeley Internet Name Domain) [BIND, 2004]. O protocolo DNS utiliza UDP e usa a porta 53.

O DNS é comumente empregado por outras entidades da camada de aplicação — inclusive HTTP, SMTP e FTP — para traduzir nomes de hospedeiros fornecidos por usuários para endereços IP. Como exemplo, considere o que acontece quando um browser (isto é, um cliente HTTP), que roda na máquina de algum usuário, requisita o URL `www.someschool.edu/index.html`. Para que a máquina do usuário possa enviar uma mensagem de requisição HTTP ao servidor Web `www.someschool.edu`, ela precisa primeiramente obter o endereço IP de `www.someschool.edu`. Isso é feito da seguinte maneira:

1. A própria máquina do usuário executa o lado cliente da aplicação DNS.
2. O browser extrai o nome de hospedeiro, `www.someschool.edu`, do URL e passa o nome para o lado cliente da aplicação DNS.
3. O cliente DNS envia uma consulta contendo o nome do hospedeiro para um servidor DNS.
4. O cliente DNS finalmente recebe uma resposta, que inclui o endereço IP para o nome de hospedeiro.
5. Tão logo o browser receba o endereço do DNS, pode abrir uma conexão TCP com o processo servidor HTTP localizado naquele endereço IP.

Vemos, por esse exemplo, que o DNS adiciona mais um atraso — às vezes substancial — às aplicações de Internet que o usam. Felizmente, como discutiremos mais adiante, o endereço IP procurado freqüentemente está no cache de um servidor DNS 'próximo', o que ajuda a reduzir o tráfego DNS na rede, bem como o atraso médio do DNS.

Princípios na prática

DNS: funções decisivas de rede via paradigma cliente–servidor

Como os protocolos HTTP, FTP e SMTP, o DNS é um protocolo da camada de aplicação, já que (1) roda entre sistemas finais comunicantes usando o paradigma cliente–servidor e (2) depende de um protocolo de transporte fim-a-fim subjacente para transferir mensagens DNS entre sistemas finais comunicantes. Em outro sentido, contudo, o papel do DNS é bastante diferente das aplicações Web, da transferência de arquivo e do e-mail. Diferentemente dessas aplicações, o DNS não é uma aplicação com a qual o usuário interage diretamente. Em vez disso, fornece uma função interna da Internet — a saber, a tradução de nomes de hospedeiro para seus endereços IP subjacentes, para aplicações de usuário e outros softwares da Internet. Notamos, na Seção 1.2, que muito da complexidade da arquitetura da Internet está localizada na 'periferia' da rede. O DNS, que implementa o processo crucial de tradução de nome para endereço usando clientes e servidores localizados nas bordas da rede, é mais um exemplo dessa filosofia de projeto.

O DNS provê alguns outros serviços importantes além da tradução de nomes de hospedeiro para endereços IP:

- **Apelidos de hospedeiro.** Um hospedeiro com nome complicado pode ter um ou mais apelidos. Um nome como `relay1.west-coast.enterprise.com` pode ter, por exemplo, dois apelidos, como `enterprise.com` e `www.enterprise.com`. Nesse caso, o nome de hospedeiro `relay1.west-coast.enterprise.com` é denominado **nome canônico**. Apelidos, quando existem, são em geral mais fáceis de lembrar do que o nome canônico. O DNS pode ser chamado por uma aplicação para obter o nome canônico correspondente a um apelido fornecido, bem como para obter o endereço IP do hospedeiro.
- **Apelidos de servidor de correio.** Por razões óbvias, é adequado que endereços de e-mail sejam fáceis de lembrar. Por exemplo, se Bob tem uma conta no Hotmail, seu endereço de e-mail pode ser simplesmente `bob@hotmail.com`. Contudo, o nome de hospedeiro do servidor do Hotmail é mais complicado e muito mais difícil de lembrar do que simplesmente `hotmail.com` (por exemplo, o nome canônico pode ser algo parecido com `relay1.west-coast.hotmail.com`). O DNS pode ser chamado por uma aplicação de correio para obter o nome canônico a partir de um apelido fornecido, bem como o endereço IP do hospedeiro. Na verdade, o registro MX (veja adiante) permite que o servidor de correio e o servidor Web de uma empresa tenham nomes (apelidos) idênticos; por exemplo, o servidor Web e o servidor de correio de uma empresa podem ambos ser denominados `enterprise.com`.
- **Distribuição de carga.** O DNS também é usado para realizar distribuição de carga entre servidores replicados, tais como os servidores Web replicados. Sites movimentados como `cnn.com` são replicados em vários servidores, sendo que cada servidor roda em um sistema final diferente e tem um endereço IP diferente. Assim, no caso de servidores Web replicados, um *conjunto* de endereços IP fica associado a um único nome canônico e contido no banco de dados do DNS. Quando clientes consultam um nome mapeado para um conjunto de endereços, o DNS responde com o conjunto inteiro de endereços IP, mas faz um rodízio da ordem dos endereços dentro de cada resposta. Como um cliente normalmente envia sua mensagem de requisição HTTP ao endereço IP que ocupa o primeiro lugar no conjunto, o rodízio de DNS distribui o tráfego entre os servidores replicados. O rodízio de DNS também é usado para e-mail, de modo que vários servidores de correio podem ter o mesmo apelido. Recentemente, empresas distribuidoras de conteúdo como a Akamai [Akamai, 2004] passaram a usar o DNS de maneira mais sofisticada para prover distribuição de conteúdo na Web (veja Capítulo 7).

O DNS está especificado no RFC 1034 e no RFC 1035 e atualizado em diversos RFCs adicionais. É um sistema complexo e, neste livro, apenas mencionamos os aspectos fundamentais de sua operação. O leitor interessado pode consultar os RFCs citados, o livro escrito por Abitz e Liu [Abitz, 1993] e também o artigo de [Mockapetris, 1988], que apresenta uma retrospectiva e uma ótima descrição do que e do porquê do DNS.

2.5.2 Visão geral do modo de funcionamento do DNS

Apresentaremos, agora, uma visão panorâmica do modo de funcionamento do DNS. Nossa discussão focalizará o serviço de tradução de nome de hospedeiro para endereço IP.

Suponha que uma certa aplicação (como um browser Web ou um leitor de correio), que executa na máquina de um usuário, precise traduzir um nome de hospedeiro para um endereço IP. A aplicação chamará o lado cliente do DNS, especificando o nome de hospedeiro que precisa ser traduzido. (Em muitas máquinas UNIX, `gethostbyname( )` é a chamada de função que uma aplicação invoca para realizar a tradução. Na Seção 2.7, mostraremos como uma aplicação Java pode chamar o DNS). A partir daí, o DNS do hospedeiro do usuário assume o controle, enviando uma mensagem de consulta para dentro da rede. Todas as mensagens de consulta e de resposta do DNS são enviadas dentro de datagramas UDP à porta 53. Após um atraso na faixa de milissegundos a segundos, o DNS no hospedeiro do usuário recebe uma mensagem de resposta DNS fornecendo o mapeamento desejado, que é, então, passado para a aplicação que está interessada. Portanto, do ponto de vista dessa aplicação, que está na máquina do cliente, o DNS é uma caixa-preta que provê um serviço de tradução simples e direto. Mas, na realidade, a caixa-preta que implementa o serviço é complexa, consistindo em um grande número de servidores de nomes distribuídos ao redor do mundo, bem como em um protocolo de camada de aplicação que especifica como se comunicam os servidores de nomes e os hospedeiros que fazem a consulta.

Um arranjo simples para DNS seria ter um servidor de nomes contendo todos os mapeamentos. Nesse projeto centralizado, os clientes simplesmente dirigiriam todas as consultas a esse único servidor de nomes, que responderia diretamente aos clientes que estão fazendo a consulta. Embora a simplicidade desse arranjo seja atraente, ele não é adequado para a Internet de hoje com seu vasto (e crescente) número de hospedeiros. Dentre os problemas de um arranjo centralizado, estão:

- **Um único ponto de falha.** Se o servidor de nomes quebrar, a Internet inteira quebrará!
- **Volume de tráfego.** Um único servidor de nomes teria de manipular todas as consultas DNS (para todas as requisições HTTP e mensagens de e-mail geradas por centenas de milhões de hospedeiros).
- **Banco de dados centralizado distante.** Um único servidor de nomes nunca poderia estar 'próximo' de todos os clientes que fazem consultas. Se colocarmos o único servidor de nomes na cidade de Nova York, todas as buscas provenientes da Austrália terão de viajar até o outro lado do globo, talvez por linhas lentas e congestionadas, o que pode resultar em atrasos significativos.
- **Manutenção.** O único servidor de nomes teria de manter registros de todos os hospedeiros da Internet. Esse banco de dados não somente seria enorme, mas também teria de ser atualizado freqüentemente para atender a todos os novos hospedeiros.

Em resumo, um banco de dados centralizado em um único servidor DNS simplesmente não é escalável. Conseqüentemente, o DNS é distribuído por conceito de projeto. Na verdade, ele é um ótimo exemplo de como um banco de dados distribuído pode ser implementado na Internet.

Um banco de dados distribuído, hierárquico

Para tratar da questão da escala, o DNS usa um grande número de servidores, organizados de maneira hierárquica e distribuídos por todo o mundo. Nenhum servidor de nomes isolado tem todos os mapeamentos

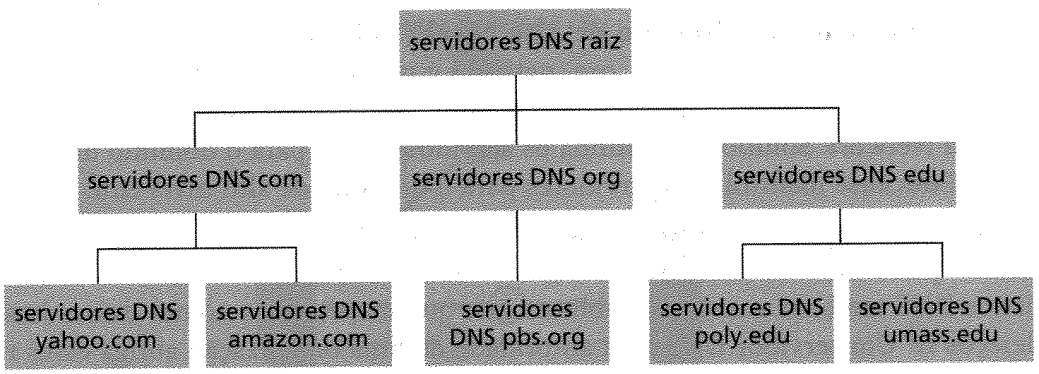


Figura 2.18 Parte da hierarquia de servidores DNS

para todos os hospedeiros da Internet. Em vez disso, os mapeamentos são distribuídos pelos servidores de nomes. Como uma primeira aproximação, há três classes de servidores de nomes: servidores de nomes raiz, servidores DNS de domínio de alto nível (*top-level domain* — TLD) e servidores DNS com autoridade — organizados em uma hierarquia, como mostra a Figura 2.18.

Para entender como essas três classes de servidores interagem, suponha que um cliente DNS queira determinar o endereço IP para o nome de hospedeiro `www.amazon.com`. Como uma primeira aproximação, ocorrerão os seguintes eventos. Em primeiro lugar, o cliente contatará um dos servidores raiz, que retornará endereços IP dos servidores TLD para o domínio de alto nível `com`. Então, o cliente contatará um desses servidores TLD, que retornará o endereço IP de um servidor com autoridade para `amazon.com`. Finalmente, o cliente contatará um dos servidores com autoridade para amazon.com, que retornará o endereço IP para o nome de hospedeiro `www.amazon.com`. Mais adiante, analisaremos mais detalhadamente esse processo de consulta DNS. Mas, em primeiro lugar, vamos examinar mais de perto as três classes de servidores DNS.

Servidores de nomes raiz. Na Internet há 13 servidores de nomes raiz (denominados de A a M) e a maior parte deles está localizada na América do Norte. Um mapa de servidores de nomes raiz de fevereiro de 2004 é mostrado na Figura 2.19; uma lista dos servidores de nomes raiz existentes hoje está disponível em [Root-servers, 2004]. Embora tenhamos nos referido a cada um dos 13 servidores de nomes raiz como se fossem um servidor único, na realidade, cada um é um conglomerado de servidores replicados, para fins de segurança e confiabilidade.

Figura 2.19 Servidores DNS raiz em 2004 (nome, organização, localização)

- **Servidores de nomes de Domínio de Alto Nível (TLD).** Esses servidores são responsáveis por domínios de alto nível como com, org, net, edu e gov, e por todos os domínios de alto nível de países, tais como uk, fr, ca e jp. Na época em que escrevíamos este livro (2º trimestre de 2004), a empresa Network Solutions mantinha os servidores TLD para o domínio de alto nível com e a empresa Educause mantinha os servidores para o domínio de alto nível edu.
- **Servidores de nomes com autoridade.** Toda organização que tiver hospedeiros que possam ser acessados publicamente na Internet (como servidores Web e servidores de correio) deve fornecer registros DNS também acessíveis publicamente que mapeiem os nomes desses hospedeiros para endereços IP. Um servidor DNS com autoridade de uma organização abriga esses registros. Uma organização pode preferir implementar seu próprio servidor DNS com autoridade para abrigar esses registros ou, como alternativa, pode pagar para armazená-los em um servidor DNS com autoridade de algum provedor de serviço. A maioria das universidades e empresas de grande porte implementam e mantêm seus próprios servidores DNS primário e secundário (backup) com autoridade.

Os servidores de nomes raiz, TLD e com autoridade pertencem à hierarquia de servidores DNS, como mostra a Figura 2.18. Há um outro tipo importante de DNS, denominado **servidor DNS local**, que não pertence, estritamente, à hierarquia de servidores, mas, mesmo assim, é central para a arquitetura DNS.

Cada ISP — como o de uma universidade, de um departamento acadêmico, de uma empresa ou de uma residência — tem um servidor de nomes local (também denominado servidor de nomes default). Quando um hospedeiro se conecta com um ISP, o ISP fornece ao hospedeiro os endereços IP de um ou mais de seus servidores de nomes locais (normalmente por DHCP, que será discutido na Seção 5.4.3). Determinar o endereço IP do seu servidor de nomes local é fácil: basta acessar a janela de estado da rede no Windows ou UNIX. O servidor de nomes local de um hospedeiro normalmente está 'próximo' dele. No caso de um ISP institucional, pode estar na mesma LAN do hospedeiro; já no caso de um ISP residencial, em geral o servidor de nomes está separado do hospedeiro por não mais do que alguns roteadores. Quando um hospedeiro faz uma consulta ao DNS, ela é enviada ao servidor de nomes local, que age como proxy e a retransmite para a hierarquia do servidor DNS, como discutiremos mais detalhadamente a seguir.

Vamos examinar um exemplo simples. Suponha que o hospedeiro `cis.poly.edu` deseje o endereço IP de `gaia.cs.umass.edu`. Suponha também que o servidor de nomes local da Polytechnic seja denominado `dns.poly.edu` e que um servidor de nomes com autoridade para `gaia.cs.umass.edu` seja denominado `dns.umass.edu`. Como mostra a Figura 2.20, o hospedeiro `cis.poly.edu` primeiramente envia uma mensagem de consulta DNS a seu servidor de nomes local `dns.poly.edu`. Essa mensagem contém o nome de hospedeiro a ser traduzido, isto é, `gaia.cs.umass.edu`. O servidor de nomes local transmite a mensagem de consulta a um servidor de nomes raiz, que percebe o sufixo edu e retorna ao servidor de nomes local uma lista de endereços IP contendo servidores TLD responsáveis por `edu`. Então, o servidor de nomes local retransmite a mensagem de consulta a um desses servidores TLD que, por sua vez, percebe o sufixo `umass.edu` e responde com o endereço IP do servidor de nomes com autoridade para a University of Massachusetts, a saber, `dns.umass.edu`. Finalmente, o servidor de nomes local reenvia a mensagem de consulta diretamente a `dns.umass.edu`, que responde com o endereço IP de `gaia.cs.umass.edu`. Note que, nesse exemplo, para poder obter o mapeamento para um único nome de hospedeiro, foram enviadas oito mensagens DNS: quatro mensagens de consulta e quatro mensagens de resposta! Em breve veremos como o cache de DNS reduz esse tráfego de consultas.

Nosso exemplo anterior afirmou que o servidor TLD conhece o servidor de nomes com autoridade para o nome de hospedeiro, o que nem sempre é verdade. Ele pode conhecer apenas um servidor de nomes intermediário que, por sua vez, conhece o servidor de nomes com autoridade para o nome de hospedeiro. Por exemplo, suponha novamente que a Universidade de Massachusetts tenha um servidor de nomes para a universidade denominado `dns.umass.edu`. Imagine também que cada um dos departamentos da universidade tenha seu próprio servidor de nomes e que cada servidor de nomes departamental seja um servidor de

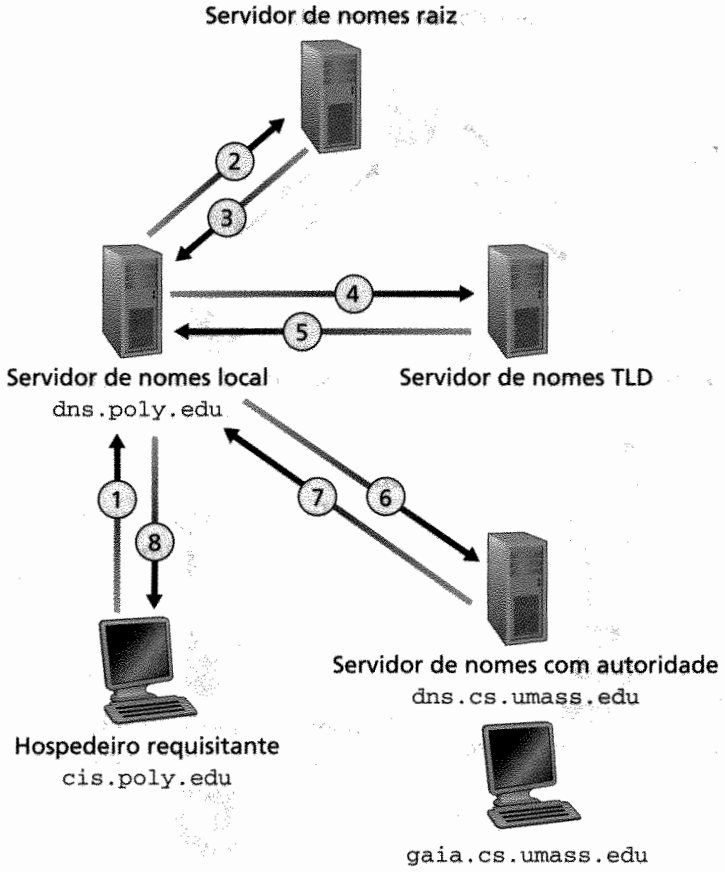


Figura 2.20 Interação de vários servidores DNS

nomes com autoridade para todos os hospedeiros do departamento. Nesse caso, quando o servidor de nomes intermediário dns.umass.edu receber uma consulta para um hospedeiro cujo nome termina com `cs.umass.edu`, ele retornará a dns.poly.edu o endereço IP de `dns.cs.umass.edu`, que tem autoridade para todos os nomes de hospedeiro que terminam com `cs.umass.edu`. Então, o servidor de nomes local dns.poly.edu enviará a consulta ao servidor de nomes com autoridade, que retornará o mapeamento desejado para o servidor de nomes local e que, por sua vez, o repassará ao hospedeiro requisitante. Nesse caso, serão enviadas dez mensagens DNS no total!

O exemplo mostrado na Figura 2.20 usa **consultas recursivas** e **consultas iterativas.** A consulta enviada de `cis.poly.edu` para `dns.poly.edu` é recursiva, visto que pede a `dns.poly.edu` que obtenha o mapeamento em seu nome. Mas as três consultas subseqüentes são iterativas, visto que todas as respostas são retornadas diretamente a `dns.poly.edu`. Teoricamente, qualquer consulta DNS pode ser iterativa ou recursiva. Por exemplo, a Figura 2.21 mostra uma cadeia de consultas DNS na qual todas elas são recursivas. Na prática, as consultas normalmente seguem o padrão mostrado na Figura 2.20: a consulta do hospedeiro requisitante ao servidor de nomes local é recursiva e todas as outras são iterativas.

Cache DNS

Até aqui, nossa discussão ignorou o **cache DNS,** uma característica muito importante do sistema DNS. Na realidade, o DNS explora extensivamente o cache para melhorar o desempenho quanto ao atraso e reduzir o número de mensagens DNS que ricocheteia pela Internet. A idéia que está por trás do cache DNS é muito simples. Em uma cadeia de consultas, quando um servidor de nomes recebe uma resposta DNS (contendo, por exemplo, o mapeamento de um nome de hospedeiro para um endereço IP), ele pode fazer cache das informações da resposta em sua memória local. Por exemplo, na Figura 2.20, toda vez que o servidor de nomes local `dns.poly.edu` recebe uma resposta de algum servidor DNS, pode fazer cache de qualquer

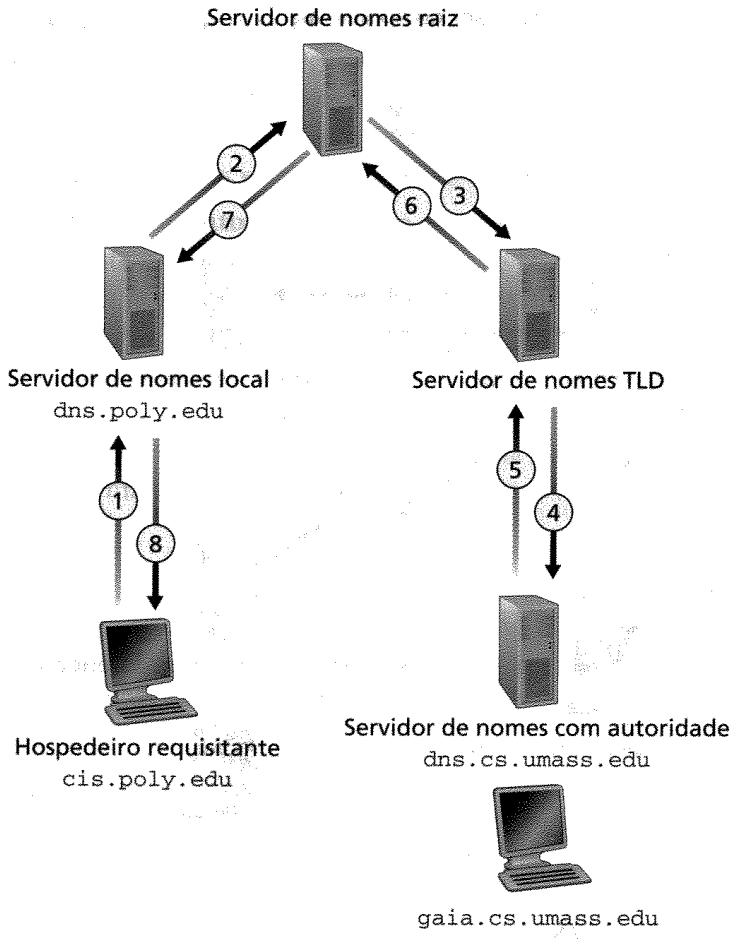


Figura 2.21 Consultas recursivas em DNS

informação contida na resposta. Se um par nome de hospedeiro/endereço IP estiver no cache de um servidor DNS e outra consulta chegar ao mesmo servidor para o mesmo nome de máquina, o servidor de nomes poderá fornecer o endereço IP desejado, mesmo que não tenha autoridade para esse nome. Como hospedeiros e mapeamentos entre hospedeiros e endereços IP não são, de modo algum, permanentes, após um período de tempo (freqüentemente dois dias), os servidores DNS descartam as informações armazenadas em seus caches.

Como exemplo, imagine que um hospedeiro `apricot.poly.edu` consulte `dns.poly.edu` para o endereço IP da máquina `cnn.com`. Além disso, suponha que algumas horas mais tarde outra máquina da Polytechnic University, digamos, `kiwi.poly.fr` também consulte `dns.poly.edu` para o mesmo nome de hospedeiro. Por causa do cache, o servidor local poderá imediatamente retornar o endereço IP de `cnn.com` a esse segundo hospedeiro requisitante, sem ter de consultar quaisquer outros servidores DNS. Um servidor de nomes local também pode fazer cache de endereços IP de servidores TLD, permitindo, assim, que servidores de nomes locais evitem os servidores de nomes raiz em uma cadeia de consultas (isso acontece freqüentemente).

2.5.3 Registros e mensagens DNS

Os servidores de nomes que juntos implementam o banco de dados distribuído do DNS armazenam registros de recursos (RR) que fornecem mapeamentos de nomes de hospedeiros para endereços IP. Cada mensagem de resposta DNS carrega um ou mais registros de recursos. Nesta seção e na subseqüente, apresentaremos uma breve visão geral dos registros de recursos e mensagens DNS. Para mais detalhes, consulte [Abitz, 1993] ou [RFC 1034; RFC 1035].

Um registro de recurso é uma tupla de quatro elementos que contém os seguintes campos:

```
(Name, Value, Type, TTL)
```

TTL é o tempo de vida útil do registro de recurso; determina quando um recurso deve ser removido de um cache. Nos exemplos de registros dados a seguir, ignoramos o campo TTL. Os significados de Name e Value dependem de Type:

- Se Type=A, então Name é um nome de hospedeiro e Value é o endereço IP para o nome de hospedeiro. Assim, um registro Type A fornece o mapeamento padrão de nomes hospedeiros para endereços IP. Como exemplo, (relay1.bar.foo.com, 145.37.93.126, A) é um registro com Type igual a A.
- Se Type=NS, então Name é um domínio (como foo.com) e Value é o nome de um servidor de nomes com autoridade que sabe como obter os endereços IP para hospedeiros do domínio. Esse registro é usado para encaminhar consultas DNS ao longo da cadeia de consultas. Como exemplo, (foo.com, dns.foo.com, NS) é um registro com Type igual a NS.
- Se Type=CNAME, então Value é um nome canônico de hospedeiro para o apelido de hospedeiro contido em Name. Esse registro pode fornecer aos hospedeiros consultantes o nome canônico correspondente a um nome de hospedeiro. Como exemplo, (foo.com, relay1.bar.foo.com, CNAME) é um registro CNAME.
- Se Type=MX, então Value é o nome canônico de um servidor de correio cujo apelido de hospedeiro está contido em Name. Como exemplo, (foo.com. mail.bar.foo.com, MX) é um registro MX. Registros MX permitem que os nomes de hospedeiros de servidores de correio tenham apelidos simples. Note que usando o registro MX, uma empresa pode ter o mesmo apelido para seu servidor de arquivo e para um de seus outros servidores (tal como seu servidor Web). Para obter o nome canônico do servidor de correio, um cliente DNS consultaria um registro MX; para obter o nome canônico do outro servidor, o cliente DNS consultaria o registro CNAME.

Se um servidor de nomes tiver autoridade para um determinado nome de hospedeiro, então conterá um registro Type A para o nome de hospedeiro. (Mesmo que não tenha autoridade, o servidor de nomes pode conter um registro Type A em seu cache.) Se um servidor não tiver autoridade para um nome de hospedeiro, conterá um registro Type NS para o domínio que inclui o nome e um registro Type A que fornece o endereço IP do servidor de nomes no campo Value do registro NS. Como exemplo, suponha que um servidor TLD edu não tenha autoridade para o hospedeiro gaia.cs.umass.edu. Nesse caso, esse servidor conterá um registro para um domínio que inclui o hospedeiro cs.umass.edu, por exemplo (umass.edu, dns.umass.edu, NS). O servidor TLD edu conterá também um registro Type A, que mapeia o servidor de nome dns.umass.edu para um endereço IP, por exemplo (dns.umass.edu, 128.119.40.111, A).

Mensagens DNS

Abordamos anteriormente nesta seção mensagens de consulta e de resposta DNS, que são as duas únicas espécies de mensagem DNS. Além disso, tanto as mensagens de consulta como as de resposta têm o mesmo formato, como mostra a Figura 2.22. A semântica dos vários campos de uma mensagem DNS é a seguinte:

- Os primeiros 12 bytes formam a *seção de cabeçalho*, que tem vários campos. O primeiro campo é um número de 16 bits que identifica a consulta. Esse identificador é copiado para a mensagem de resposta a uma consulta, permitindo que o cliente combine respostas recebidas com consultas enviadas. Há várias flags no campo de flag. Uma flag de consulta/resposta de 1 bit indica se a mensagem é uma consulta (0) ou uma resposta (1). Uma flag de autoridade de 1 bit é marcada em uma mensagem de resposta quando o servidor de nomes é um servidor com autoridade para um

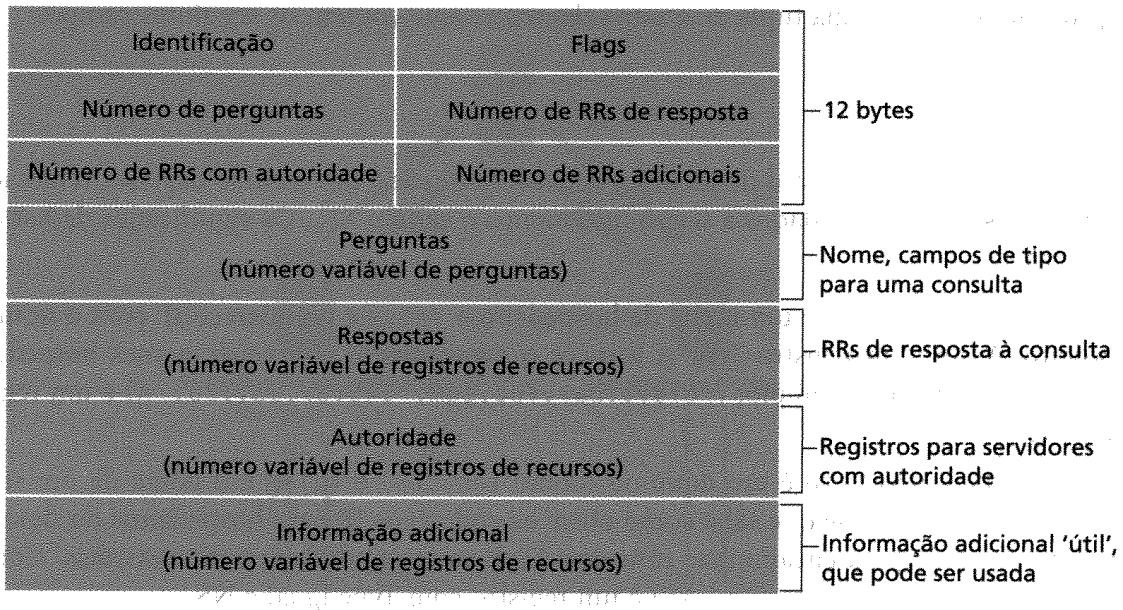


Figura 2.22 Formato da mensagem DNS

nome consultado. Uma flag de recursão desejada de 1 bit é estabelecida quando um cliente (hospedeiro ou servidor de nomes) quer que um servidor de nomes proceda recursivamente sempre que não tem o registro. Um campo de recursão disponível de 1 bit é marcado em uma resposta se o servidor de nomes suporta buscas recursivas. No cabeçalho, há também quatro campos de 'número de'. Esses campos indicam o número de ocorrências dos quatro tipos de seção de dados que se seguem ao cabeçalho.

- A *seção de pergunta* contém informações sobre a consulta que está sendo feita. Essa seção inclui (1) um campo de nome que contém o nome que está sendo consultado e (2) um campo de tipo que indica o tipo de pergunta que está sendo feito sobre o nome — por exemplo, um endereço de hospedeiro associado a um nome (Type A) ou o servidor de correio para um nome (Type MX).
- Em uma resposta de um servidor de nomes, a *seção de resposta* contém os registros de recursos para o nome que foi consultado originalmente. Lembre-se de que em cada registro de recurso há o `Type` (por exemplo, A, NS, CSNAME e MX), o `Value` e o `TTL`. Uma resposta pode retornar vários RRs, já que um nome de hospedeiro pode ter diversos endereços IP (por exemplo, para servidores Web replicados, como já discutimos anteriormente nesta seção).
- A *seção de autoridade* contém registros de outros servidores com autoridade.
- A *seção adicional* contém outros registros úteis. Por exemplo, o campo resposta em uma resposta a uma consulta MX conterá um registro de recurso que informa o nome canônico de um servidor de correio. A seção adicional conterá um registro Type A que fornece o endereço IP para o nome canônico do servidor de correio.

Você gostaria de enviar uma mensagem de consulta DNS diretamente de sua máquina a algum servidor DNS? Isso é fácil de fazer com o **programa nslookup**, que está disponível na maioria das plataformas Windows e UNIX. Por exemplo, se um hospedeiro executar Windows, abra o Command Prompt e chame o programa nslookup simplesmente digitando 'nslookup'. Depois de chamar o programa, você pode enviar uma consulta DNS a qualquer servidor de nomes (raiz, TLD ou com autoridade). Após receber a mensagem de resposta do servidor de nomes, o nslookup apresentará os registros incluídos na resposta (em formato que pode ser lido normalmente). Como alternativa a executar nslookup na sua própria máquina, você pode visitar um dos muitos sites Web que permitem o emprego remoto do programa. (Basta digitar 'nslookup' em um buscador e você será levado a um desses sites.)

Para inserir registros no banco de dados do DNS

A discussão anterior focalizou como são extraídos registros do banco de dados DNS. É possível que você esteja se perguntando como os registros entraram no banco de dados em primeiro lugar. Vamos examinar como isso é feito no contexto de um exemplo específico. Imagine que você acabou de criar uma nova empresa muito interessante denominada Network Utopia. A primeira coisa que você certamente deverá fazer é registrar o nome de domínio `networkutopia.com` em uma entidade registradora. Uma **entidade registradora** é uma entidade comercial que verifica se o nome de domínio é exclusivo, registra-o no banco de dados do DNS (como discutiremos mais adiante) e cobra uma pequena taxa por seus serviços. Antes de 1999, uma única entidade registradora, a Network Solutions, detinha o monopólio do registro de nomes de domínio para domínios `com`, `net` e `org`. Mas agora existem muitas entidades registradoras credenciadas pela Internet Corporation for Assigned Names and Numbers (ICANN) competindo por clientes. Uma lista completa dessas entidades está disponível em `http://www.internic.net`.

Ao registrar o nome de domínio `networkutopia.com`, você também precisará informar os nomes e endereços IP dos seus servidores DNS com autoridade, primários e secundários. Suponha que os nomes e endereços IP sejam `dns1.networkutopia.com`, `dns2.networkutopia.com`, `212.212.212.1` e `212.212.212.2`. A entidade registradora ficará encarregada de providenciar a inserção dos registros Type NS e de um registro Type A nos servidores TLD do domínio com para cada um desses dois servidores de nomes com autoridade. Especificamente para o servidor primário com autoridade `networkutopia.com`, a autoridade registradora inseriria no sistema DNS os dois registros de recursos seguintes:

```
(networkutopia.com, dns1.networkutopia.com, NS)

(dns1.networkutopia.com, 212.212.212.1, A)
```

Não esqueça de providenciar também a inserção em seus servidores de nomes com autoridade do registro de recurso Type A para seu servidor Web `www.networkutopia.com` e o registro de recurso Type MX para seu servidor de correio `mail.networkutopia.com`. (Até há pouco tempo, o conteúdo de cada servidor DNS era configurado estaticamente, por exemplo, a partir de um arquivo de configuração criado por um gerenciador de sistema. Mais recentemente, foi acrescentada ao protocolo DNS uma opção UPDATE que permite que dados sejam dinamicamente acrescentados no banco de dados ou apagados deles por meio de mensagens DNS. O RFC 2136 especifica atualizações dinâmicas do DNS.)

Quando todas essas etapas estiverem concluídas, o público em geral poderá visitar seu site Web e enviar e-mails aos empregados de sua empresa. Vamos concluir nossa discussão do DNS verificando que essa afirmação é verdadeira, o que também ajudará a solidificar aquilo que aprendemos sobre o DNS. Suponha que Alice, que está na Austrália, queira consultar a página Web `www.networkutopia.com`. Como discutimos anteriormente, seu hospedeiro primeiramente enviará uma consulta DNS a seu servidor de nomes local, que então contatará um servidor TLD do domínio `com`. (O servidor de nomes local também terá de contatar um servidor de nomes raiz caso não tenha em seu cache o endereço de um servidor TLD `com`.) Esse servidor TLD contém os registros de recursos Type NS e Type A citados anteriormente, porque a entidade registradora já os tinha inserido em todos os servidores TLD com. O servidor TLD `com` envia uma resposta ao servidor de nomes local de Alice, contendo os dois registros de recursos. Então, o servidor de nomes local envia uma consulta DNS a `212.212.212.1`, solicitando o registro Type A correspondente a `www.networkutopia.com`. Este registro provê o endereço IP do servidor Web desejado, digamos, `212.212.71.4`, que o servidor local de nomes transmite para o hospedeiro de Alice. Agora, o browser de Alice pode iniciar uma conexão TCP com o hospedeiro `212.212.71.4` e enviar uma requisição HTTP pela conexão. Ufa! Acontecem muito mais coisas do que percebemos quando navegamos na Web!

2.6 Compartilhamento de arquivos P2P

Quando escrevíamos este livro (início de 2004), o compartilhamento de arquivos P2P respondia por mais tráfego na Internet do que qualquer outra aplicação, incluindo a Web. Assim, exclusivamente em termos de tráfego, o compartilhamento de arquivos P2P pode ser considerado a aplicação mais importante da Internet. Sistemas modernos de compartilhamento de arquivos P2P não somente compartilham MP3 (normalmente de 3 a 8 Mbytes), mas também vídeos (geralmente de 10 a 1.000 Mbytes), software, documentos e imagens. Nesta seção, nossa discussão focalizará as questões de protocolo e de rede no compartilhamento de arquivos P2P. Também há muitos tópicos importantes relativos à segurança, privacidade, anonimato, violação de direitos autorais e propriedade intelectual; indicamos [Von Lohmann, 2003; Clarke, 2002] caso você queira consultar uma discussão sobre esses e outros assuntos.

Antes de descrever os componentes internos do compartilhamento de arquivos P2P, vamos percorrer um exemplo típico de como alguém poderia usar tal sistema. Suponha que Alice use a aplicação de compartilhamento de arquivo P2P para baixar MP3. Ela executa o software de compartilhamento de arquivos P2P em seu computador (um par) para acessar a Internet por meio de uma conexão ADSL e desliga o computador todas as noites. Esse computador não tem um nome de hospedeiro e toda vez que Alice se conecta novamente com a Internet, seu ISP designa ao PC um novo endereço IP.

Imagine que Alice esteja ligada à Internet agora e que lance sua aplicação de compartilhamento de arquivos P2P. Usando essa aplicação, ela consulta o MP3 em busca de uma canção denominada "Amor na Internet". Logo após dar o comando de busca, a aplicação exibirá uma lista de pares que têm uma cópia da canção para compartilhar e que estão conectados na Internet naquele momento. Cada um desses pares normalmente é um PC comum, pertencente a um usuário comum da Internet como Alice e, para cada um deles, a aplicação pode apresentar algumas informações adicionais, como o acesso em banda larga e a estimativa do tempo de transferência. Em seguida, Alice requisita o arquivo MP3 de algum dos pares, por exemplo, o PC de Bob. Uma conexão direta é estabelecida entre os dois computadores e o arquivo MP3 é enviado de Bob para Alice. Se Bob inadvertidamente desconectar seu PC da Internet durante a transferência, então o software de compartilhamento de arquivos P2P de Alice pode tentar obter o restante do arquivo de um outro par que o tenha. E, enquanto Alice estiver baixando "Amor na Internet" do computador de Bob, algum outro par, digamos, o PC de Claire, também pode baixar outro MP3, por exemplo, "Network 'n Roll" diretamente do PC de Alice. Desse modo, todos os pares participantes são consumidores e distribuidores de conteúdo.

O compartilhamento de arquivos P2P é um paradigma de distribuição atraente porque todo o conteúdo é transferido diretamente entre pares comuns, sem passar por servidores de terceiros. Assim, essa aplicação tira proveito dos recursos (largura de banda, armazenamento e CPU) de um grande conjunto de pares — às vezes milhões deles! Em outras palavras, o compartilhamento de arquivos P2P é altamente escalável.

Embora não exista envolvimento de nenhum servidor centralizado de terceiros na transferência de arquivos, é importante ter sempre em mente que o compartilhamento de arquivos P2P ainda se baseia no paradigma cliente-servidor (veja Seção 2.1.2). Realmente, o par requisitante é o cliente e o par escolhido é o servidor. O arquivo é enviado do par servidor ao par cliente com um protocolo de transferência de arquivos. Visto que qualquer par pode requisitar ou ser escolhido, todos os pares podem executar tanto o lado cliente quanto o lado servidor do protocolo de transferência de arquivos.

Agora, suponha que o protocolo de transferência de arquivos seja HTTP, o que normalmente acontece. Ampliando nosso exemplo acima, quando Alice seleciona Bob para baixar "Amor na Internet", seu computador envia a Bob uma requisição HTTP para essa canção e Bob envia uma resposta HTTP contendo a canção. Note que, enquanto Alice estiver executando a aplicação de compartilhamento de arquivos P2P, seu computador (par) é um cliente Web e também um **servidor Web transitório**. Seu par é um servidor Web porque está servindo conteúdo dentro de respostas HTTP; é transitório porque está conectado apenas intermitentemente com a Internet e pode obter um novo endereço IP toda vez que se conectar novamente com a rede. Até aqui, explicamos a parte fácil do compartilhamento de arquivos P2P, mas não ensinamos como um par determina quais pares têm os objetos (isto é, arquivos) que ele deseja. Em um sistema de compar-

tilhamento de arquivos P2P, normalmente há um grande número de pares conectados, cada um deles contendo objetos para compartilhar, incluindo MP3, vídeos, imagens e software. Se o par X estiver interessado em obter um determinado objeto, então ele deverá ter um modo de determinar os endereços IP dos pares conectados que tenham cópias do objeto desejado. Porque os pares se conectam e se desconectam, esse problema não é trivial. Em seguida, discutiremos três arquiteturas para localizar conteúdo, cada uma delas amplamente empregada por diferentes sistemas de compartilhamento de arquivos P2P. Aconselhamos o leitor interessado a consultar também os artigos de pesquisa [Stoica, 2001; Rowstron, 2001; Ratnasamy, 2001; Zhao, 2004; Maymounkov, 2002; Garces-Erce, 2003], que apresentam soluções inovadoras para o problema de localização de conteúdo.

Diretório centralizado

Uma das abordagens mais diretas da localização de conteúdo é prover um **diretório centralizado**, como fazia o Napster, a primeira empresa comercial a disponibilizar uma aplicação P2P de grande escala, bem-sucedida, para distribuição de MP3. Nesse projeto, o serviço de compartilhamento de arquivos P2P usa um servidor de grande porte (ou conjunto de servidores) para prover o serviço de diretório. Como mostra a Figura 2.23, quando um usuário lança a aplicação de compartilhamento de arquivos P2P, ela contata o servidor de diretório. Especificamente, a aplicação que está executando no par informa ao servidor de diretório seu endereço IP e os nomes dos objetos que estão disponíveis para compartilhamento em seu disco local (por exemplo, os títulos de todos os MP3 armazenados em seus arquivos). Desse modo, o servidor de diretório sabe quais objetos o par tem disponíveis para compartilhar. O servidor de diretório coleta essa informação de cada par que fica ativo, criando, dessa forma, um banco de dados centralizado, dinâmico, que mapeia cada nome de objeto para um conjunto de endereços IP. Quando um par ativo obtém um novo objeto ou remove um objeto, informa ao servidor de diretório, de modo que este possa atualizar seu banco de dados.

Para manter seu banco de dados atualizado, o servidor de diretório deve poder determinar quando um par se desconecta. Um par pode ficar desconectado fechando sua aplicação P2P cliente ou simplesmente saindo da Internet. Um modo de monitorar quais pares continuam conectados é enviar mensagens periodicamente para verificar se eles respondem. Se o servidor de diretório determinar que um par não está mais conectado, ele remove do banco de dados o endereço IP dele.

O compartilhamento de arquivos P2P com diretório centralizado usa a arquitetura híbrida cliente-servidor/P2P discutida na Seção 2.1.1. Utilizar um diretório centralizado para localizar conteúdo é conceitualmente direto, mas tem várias desvantagens:

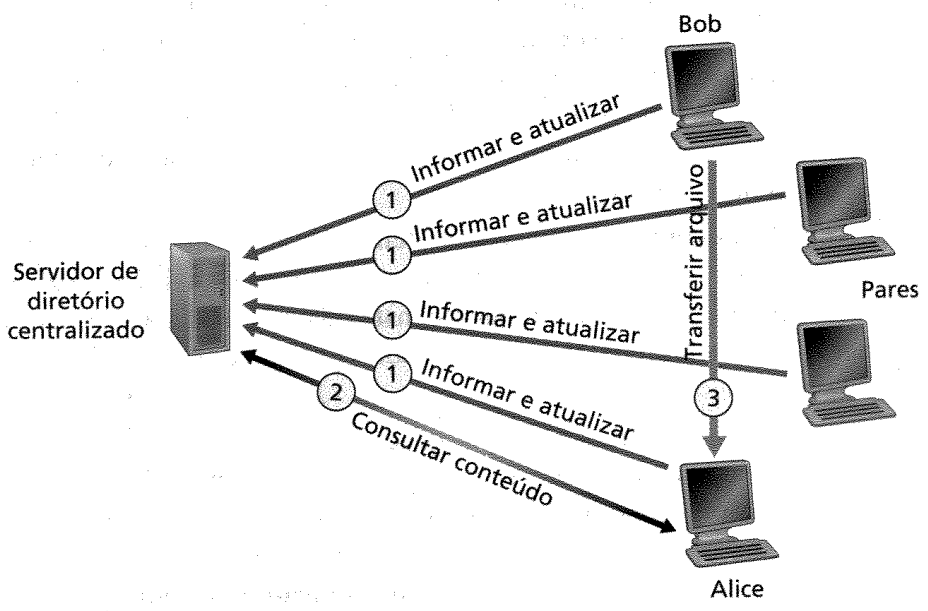


Figura 2.23 O paradigma P2P com um diretório centralizado

- *Um único ponto de falha.* Se o servidor de diretório cair, toda a aplicação P2P cairá. Mesmo utilizando um conjunto de servidores redundantes, conexões da Internet com o servidor podem falhar, o que provoca a queda da aplicação inteira.
- *Gargalo de desempenho.* Em um sistema P2P de grande porte, com centenas de milhares de usuários conectados, um servidor centralizado tem de manter um banco de dados imenso e deve responder a milhares de consultas por segundo. Realmente, em 2000, quando a aplicação P2P era mais popular, o Napster sofria muito com problemas de tráfego em seu servidor centralizado.
- *Violação de direitos autorais.* Embora esse assunto esteja fora do escopo deste livro, mencionaremos brevemente que a indústria fonográfica está preocupada (para dizer o mínimo!) que sistemas de compartilhamento de arquivos P2P permitam que usuários obtenham facilmente acesso gratuito a conteúdo protegido por direitos autorais. (Uma excelente discussão da influência das leis do direito autoral sobre o P2P pode ser encontrada em [Von Lohmann, 2003]. Quando uma empresa P2P tem um servidor de diretório centralizado, pode sofrer sanções judiciais que a obriguem a fechar o servidor de diretório. É mais difícil fechar as arquiteturas mais descentralizadas.

Em resumo, a desvantagem mais notória da utilização de um servidor de diretório centralizado é que a aplicação P2P é apenas parcialmente descentralizada. A transferência de arquivos entre pares é descentralizada, mas o processo de localização de conteúdo é altamente centralizado — uma preocupação quanto à confiabilidade e o desempenho.

Inundação de consultas

Gnutella [Gnutella, 2004], uma aplicação de compartilhamento de arquivos de domínio público, localiza conteúdo utilizando uma abordagem totalmente distribuída. Diferentemente do Napster, a Gnutella não utiliza um servidor centralizado para monitorar conteúdo nos pares. Na verdade, a abordagem da localização de conteúdo dessa aplicação é diametralmente oposta à do Napster.

O cliente Gnutella (que inclui uma interface de usuário) implementa o protocolo Gnutella e executa em um par (máquina) comum. A especificação do protocolo é mínima, o que dá flexibilidade significativa para o projeto do cliente Gnutella. Exatamente como há muitos browsers Web que implementam HTTP, há muitos clientes Gnutella que implementam o protocolo Gnutella.

Em Gnutella, os pares formam uma rede abstrata, lógica, denominada **rede de sobreposição,** cuja definição em termos teóricos e gráficos é apresentada a seguir. Se o par X mantiver uma conexão TCP com um outro par Y, então dizemos que há uma **aresta** entre X e Y. O grafo que contém todos os pares ativos e arestas de conexão (conexões TCP em curso) define a rede de sobreposição Gnutella existente no momento em questão. Note que uma aresta não é um enlace físico de comunicação; é um enlace abstrato que pode consistir em dezenas de enlaces físicos subjacentes. Uma aresta pode representar a conexão TCP entre um par localizado na Lituânia e outro no Brasil, por exemplo.

Embora uma rede Gnutella possa ter centenas de milhares de pares participantes, um determinado par normalmente estará conectado a menos do que dez outros nós na rede de sobreposição; veja a Figura 2.24. Mais adiante explicaremos como uma rede Gnutella é construída e mantida à medida que pares entram e saem da rede. Por enquanto, vamos admitir que a rede de sobreposição esteja montada e focalizar como um par localiza e extrai conteúdo.

Em Gnutella, pares enviam mensagens a pares próximos na rede de sobreposição montada sobre conexões TCP já existentes. Quando Alice quer localizar a canção "Amor na Internet", seu cliente Gnutella envia a todos os seus vizinhos uma mensagem Gnutella Query que inclui as palavras-chave "Amor na Internet". Esses vizinhos, por sua vez, retransmitem a mensagem a todos os seus vizinhos e assim por diante. Esse processo, mostrado na Figura 2.24, é denominado **inundação de consultas**. Quando um par recebe uma mensagem Query, verifica se a palavra-chave combina com qualquer um dos arquivos disponíveis

para compartilhamento no par. Se houver uma compatibilização, o par devolve a Alice uma mensagem QueryHit contendo o nome e o tamanho do arquivo compatível. A mensagem QueryHit segue o caminho inverso da mensagem Query; portanto, utiliza conexões TCP já existentes.

O processo Gnutella de Alice pode receber mensagens QueryHit de mais de um par e ela seleciona um par desse conjunto, por exemplo, o de Bob, para baixar "Amor na Internet". Seu processo Gnutella estabelece uma conexão TCP direta com o de Bob e envia para essa conexão uma mensagem GET HTTP que inclui o nome do arquivo específico. Então, Bob envia o arquivo a Alice dentro de uma mensagem de resposta HTTP. Embora as mensagens Query e QueryHit transitem pela rede de sobreposição, as mensagens HTTP GET e de resposta são enviadas fora da rede de sobreposição, diretamente entre os dois pares, como mostra a Figura 2.24. Tão logo Alice receba o arquivo inteiro, a conexão TCP entre ela e Bob é encerrada.

Embora o projeto descentralizado da Gnutella seja simples e bem estruturado, muitas vezes é criticado por não ser escalável. Em particular, com inundação de mensagens, sempre que um par inicia uma consulta, ela se propaga para todos os outros pares presentes na rede de sobreposição, introduzindo uma quantidade significativa de tráfego na Internet. Os projetistas da Gnutella enfrentaram esse problema utilizando **inundação de consultas de escopo limitado**. Especificamente, quando Alice envia sua mensagem de consulta inicial, um campo de contagem de pares contido na mensagem é configurado para um certo limite (digamos, 7). Toda vez que a mensagem de consulta chega a um novo par, este decrementa o campo de contagem de pares antes de retransmiti-la para seus vizinhos na rede de sobreposição. Quando um par recebe uma consulta cujo campo de contagem de pares é igual a zero, ele pára de retransmiti-la. Desse modo, quando um par inicia uma consulta, a inundação fica localizada em uma região da rede de sobreposição. Evidentemente, esta inundação de consultas de escopo limitado reduz o tráfego de consultas introduzido na Internet. Contudo, também reduz o número de pares que são consultados. Utilizando Gnutella, pode haver uma alta probabilidade de você não localizar o conteúdo que deseja mesmo que ele esteja presente em algum par que está fora do escopo.

Uma questão fundamental em aplicações P2P distribuídas é a da manipulação de junções e saídas de pares. Descrevemos a seguir o que acontece quando um par X quer juntar-se à rede Gnutella. Nos exercícios de fixação, examinaremos as ações que a aplicação realiza na saída de pares.

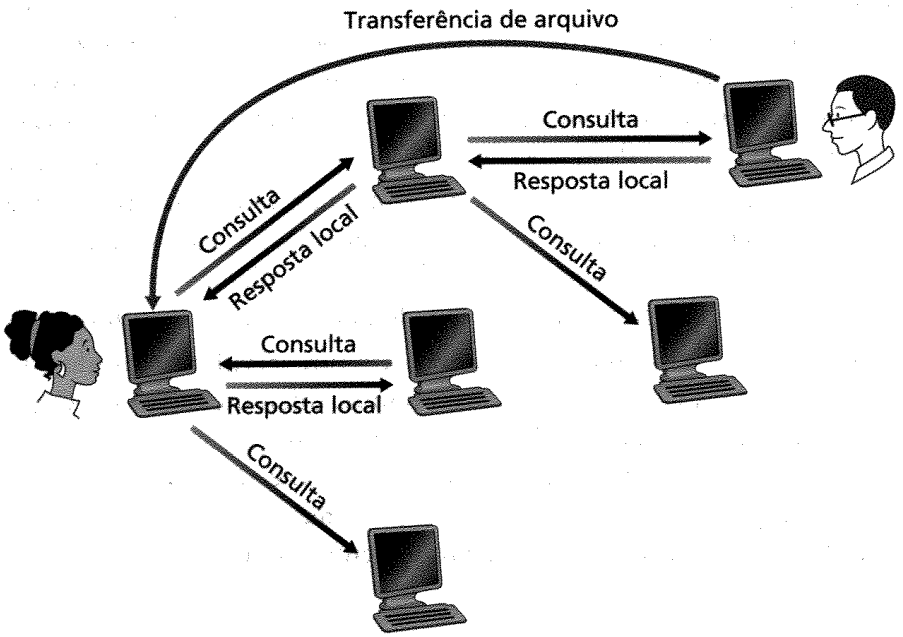


Figura 2.24 Busca e transferência de arquivo em Gnutella

1. Em primeiro lugar, o par X deve achar algum outro par que já esteja na rede de sobreposição. Uma abordagem para resolver esse **problema de autocarregamento** (*bootstrap*) é o cliente Gnutella de X manter uma lista de pares (endereços IP) que estão freqüentemente ativos na rede Gnutella; alternativamente, X pode contatar um site Gnutella que mantenha uma dessas listas.
2. Tão logo obtenha acesso a essa lista, X faz tentativas seqüenciais para estabelecer uma conexão com pares presentes na lista até ser criada uma conexão com algum par Y.
3. Após estabelecida a conexão TCP entre X e Y, o par X envia a Y uma mensagem Gnutella Ping, que também inclui um campo de contagem de pares. Ao receber a mensagem Ping, Y a transmite a todos os seus vizinhos na rede de sobreposição. Os pares continuam a retransmitir a mensagem Ping até o campo de contagem de pares ser zero.
4. Sempre que um par Z recebe uma mensagem Ping, responde retornando uma mensagem Gnutella Pong para X através da rede de sobreposição. A mensagem Pong inclui o endereço IP de Z, o número de arquivos que ele está compartilhando e o número total de kbytes ocupado pelos arquivos que está compartilhando.
5. Quando recebe as mensagens Pong, X sabe os endereços IP de muitos pares presentes na rede Gnutella, além de Y. Então, pode estabelecer conexões TCP com alguns desses outros pares, criando, assim, várias arestas ligadas a ele dentro da rede de sobreposição. O par X também pode tentar estabelecer conexões TCP com outros nós que estão se juntando à rede. A especificação Gnutella não determina com quantos pares X deve se conectar, portanto, isso fica a critério do projetista do cliente Gnutella.

Agora já apresentamos as características essenciais do protocolo Gnutella. Mais detalhes serão abordados nos exercícios de fixação. Em resumo, Gnutella é um sistema P2P distribuído simples, que permite que um usuário consulte arquivos localizados em pares próximos ('próximos' quer dizer à distância de um pequeno número de saltos na rede de sobreposição).

Explorando a heterogeneidade

Aprendemos que Napster e Gnutella utilizam duas abordagens diametralmente opostas da localização de conteúdo. O Napster utiliza um servidor de diretório centralizado e sempre localiza conteúdo quando este estiver presente em algum par participante. Gnutella utiliza uma arquitetura totalmente distribuída, mas localiza conteúdo somente em pares próximos na rede de sobreposição. O KaZaA [KaZaA, 2004] toma emprestadas idéias do Napster e da Gnutella, resultando em um poderoso sistema de compartilhamento de arquivos P2P que, na época em que escrevíamos este livro, contribuía com mais tráfego na Internet do que qualquer outra aplicação [Gummadi, 2003] [Saroiu, 2002].

A tecnologia KaZaA é proprietária e, além disso, criptografa todo o controle de tráfego (mas não os arquivos de dados). Todavia, pelas descrições básicas apresentadas na página Web do KaZaA, juntamente com alguns trabalhos de medição dessa aplicação [Liang, 2004], podemos explicar, em linhas gerais, como funciona o protocolo. O KaZaA explora a heterogeneidade de uma maneira intuitiva e natural.

Ele se parece com Gnutella, no sentido de que não usa um servidor dedicado (nem um conjunto de servidores) para monitorar e localizar conteúdo. Entretanto, diferentemente da Gnutella, nem todos os pares são iguais no KaZaA. Os pares mais poderosos — isto é, os que têm grande largura de banda e alta conectividade com a Internet — são designados líderes de grupo e têm maiores responsabilidades. Como mostra a Figura 2.25, se um par não for um líder de grupo, então é normal e fica designado a um líder de grupo. Normalmente, um líder de grupo terá até algumas centenas de pares filhos.

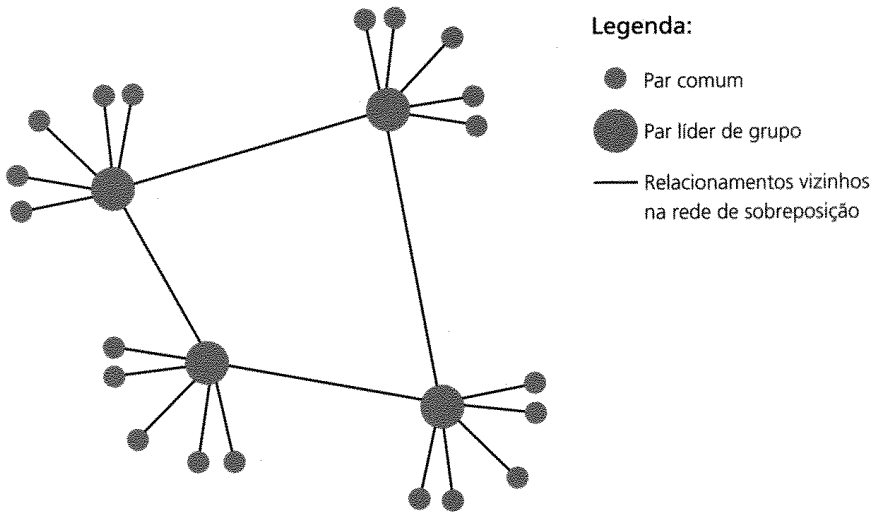


Figura 2.25 Rede de sobreposição hierárquica para compartilhamento de arquivos P2P

Quando um par lança a aplicação KaZaA, estabelece uma conexão TCP com um dos líderes de grupo. Então, o par informa a seu líder de grupo todos os arquivos que está disponibilizando para compartilhamento, o que permite ao líder manter um banco de dados que inclui os identificadores de todos os arquivos que seus filhos estão compartilhando, os metadados sobre os arquivos e os endereços IP correspondentes dos filhos que contêm os arquivos. Desse modo, cada líder de grupo torna-se um hub (distribuidor) semelhante a um mini-Napster mas, ao contrário do Napster, não é um servidor dedicado, e sim um par comum.

Se cada um dos minihubs, juntamente com seus filhos, fossem entidades isoladas no KaZaA, então a aplicação consistiria, essencialmente, em milhares de mini-Napsters isolados, sendo que cada distribuidor executaria em um par e não em um servidor de aplicação dedicado. Essa abordagem de isolamento limitaria seriamente a quantidade de conteúdo disponível em qualquer um dos pares. Para resolver essa limitação, os líderes de grupo interconectam-se com conexões TCP, criando uma rede de sobreposição entre eles semelhante à Gnutella. O KaZaA compartilha conteúdo de informação entre vários grupos. Por exemplo, um hub em um grupo poderia conectar-se com hubs em alguns outros grupos e obter cópias dos bancos de dados desses outros hubs, fundindo-os com seus próprios bancos de dados e, conseqüentemente, monitorando o conteúdo de um número maior de pares. Alternativamente, cada hub poderia monitorar apenas o conteúdo de seus filhos, mas, quando recebesse uma consulta, a retransmitiria para os hubs aos quais está conectado. (Essa última abordagem é semelhante à inundação de consultas em Gnutella, mas com a inundação de escopo limitado ocorrendo na rede de sobreposição de líderes de grupo.)

No KaZaA, cada arquivo é identificado por um hash do arquivo. Em breve veremos como esse identificador é usado. Também no KaZaA, cada objeto tem um descritor, que inclui o nome do arquivo e um texto descritivo não estruturado do objeto. Pares fazem consultas com palavras-chave, semelhante ao modo como usuários fazem consultas em buscadores. Para procurar uma compatibilização para palavras-chave, um par envia a seu líder de grupo uma consulta contendo as palavras-chave. O líder de grupo responde com uma lista de pares (dentro ou fora de seu grupo) que possuem arquivos cujos descritores são compatíveis com as palavras-chave, juntamente com os identificadores desses arquivos. Então, o líder de grupo pode retransmitir a consulta para um ou mais líderes de grupo com os quais está conectado. Respostas de líderes de grupo seguem o caminho inverso na rede de sobreposição.

Em resumo, a arquitetura do KaZaA explora a *heterogeneidade dos pares*, designando como líderes de grupo uma pequena fração dos pares mais poderosos que formam a camada superior de uma rede de sobreposição hierárquica, como mostra a Figura 2.25. Em comparação com o projeto da Gnutella, com sua sobreposição plana e inundação de escopo limitado, o projeto hierárquico permite a verificação de compatibilidade

em um número significativamente maior de pares sem criar um tráfego de consultas excessivo. (Hoje, a vasta maioria do tráfego do KaZaA deve-se à transferência de arquivos.)

O KaZaA também emprega diversas técnicas que melhoram seu desempenho. Algumas dessas técnicas, que podem ser empregadas em grande parte dos sistemas de compartilhamento de arquivos P2P, são:

1. **Enfileiramento de requisições.** O usuário pode configurar seu par de modo a limitar o número de transferências simultâneas em qualquer valor. (O valor de default está na faixa de três a sete.) Se Alice limitar seu par a três transferências, e se o par já estiver carregando três arquivos quando ela receber uma nova requisição de transferência de Bob, Alice põe as requisições de Bob, e todas as subseqüentes, em uma fila local. Essas limitações locais garantem que cada arquivo que está sendo transferido receba uma quantidade não desprezível de largura de banda do nó que está enviando o arquivo.
2. **Prioridades de incentivo.** Alice dará prioridade de enfileiramento a usuários que, no passado, tenham carregado mais arquivos do que baixado. Por exemplo, se ambos, Bob e Claire, estiverem na fila de Alice e Claire tiver carregado muito mais do que baixado, então Alice atenderá a requisição de Claire antes da de Bob. Esse mecanismo proporciona a usuários um incentivo para carregar arquivos, aprimorando a escalabilidade geral do sistema KaZaA.
3. **Transferência paralela.** Quando Claire faz uma consulta em busca da canção "Amor na Internet", o KaZaA freqüentemente retornará uma lista de vários pares que apresentam compatibilidade com as palavras-chave. Algumas dessas compatibilidades corresponderão ao mesmo arquivo, o que pode ser verificado pelos identificadores de arquivo. O cliente de Claire pode usar o cabeçalho 'faixa de bytes' do HTTP para requisitar diferentes porções do arquivo de diferentes pares, baixando, assim, porções do arquivo de diferentes pares em paralelo. Por exemplo, se Alice e Bob têm o arquivo desejado, o cliente KaZaA de Claire pode solicitar a primeira metade do arquivo à Alice e a segunda, a Bob. Essa característica permite que, em muitas circunstâncias, usuários extraiam um arquivo mais rapidamente.

Mencionamos brevemente que têm sido realizados muitos esforços de engenharia reversa para o KaZaA. Realmente, alguns projetos desenvolveram clientes KaZaA modificados que permitem ao usuário acessar a rede KaZaA sem seguir estritamente o seu protocolo. Atualmente, um cliente modificado notoriamente popular é o kazaa-lite, que não somente elimina a propaganda embutida no KaZaA e dá aos usuários a prioridade mais alta, mas também executa saltos de supernós, isto é, saltos de um líder de grupo para outro para uma única consulta de palavra-chave.

O KaZaA, assim como o Napster antes dele, tem conseguido um sucesso estrondoso na Internet, espalhando-se para literalmente milhões de usuários no espaço de poucos meses. A adoção vertiginosamente rápida e abrangente de aplicações de compartilhamento de arquivos P2P, e de mensagem instantânea pela Web antes delas, é um testemunho vivo do acerto da arquitetura geral da Internet, um projeto que não poderia ter previsto o volumoso e sempre crescente conjunto de aplicações de Internet que seriam desenvolvidas nos 25 anos seguintes. Os serviços de rede oferecidos a aplicações de Internet — transporte de datagramas não orientado para conexão, transferência confiável de datagramas orientada para conexão, a interface socket, endereçamento, nomeação (DNS), entre outras — provaram ser suficientes para permitir o desenvolvimento de milhares de aplicações. Visto que todas essas aplicações foram colocadas em uma camada que fica em cima das quatro camadas mais baixas já existentes na pilha do protocolo da Internet, elas envolvem somente o desenvolvimento de novo software cliente-servidor para ser usado em sistemas finais. Isso, por sua vez, permitiu que essas aplicações fossem distribuídas, e também adotadas, rapidamente.

Mas nem todas as novas aplicações de Internet alcançaram um sucesso estrondoso como a Web, a mensagem instantânea e o compartilhamento de arquivos P2P. Conferência multimídia e multimídia em tempo real são duas aplicações que ainda têm de alcançar sucesso em grande escala. Talvez isso se deva a um conjunto insuficiente de serviços provido pela arquitetura atual da Internet (abordaremos extensões propostas para a arquitetura da Internet para suportar essas aplicações no Capítulo 7), ou talvez a fatores sociais e/ou econômicos; o tempo dirá.

História

KaZaA

Na época em que escrevíamos este livro (abril de 2004), em um dia típico, a rede KaZaA tinha mais de 3 milhões de usuários compartilhando mais de 5 mil terabytes de conteúdo. Como grande parte desse conteúdo são canções e filmes protegidos por direitos autorais, o KaZaA é a nêmesis das indústrias musical e cinematográfica. Essa rede também traz pesadas cargas de tráfego para os ISPs locais. Por exemplo, em junho de 2002, aproximadamente 37 por cento de todo o tráfego TCP na rede do *campus* da Universidade de Washington eram consumidos pelo KaZaA — mais de duas vezes o tráfego da Web no mesmo local, na mesma época [Saroiu, 2002]. Então, de onde veio essa aplicação avassaladora e para onde ela vai?

A FastTrack, uma empresa privada holandesa, liberou o programa gratuito KaZaA pela primeira vez em abril de 2002. A primeira liberação incluía a arquitetura hierárquica de supernós que, até hoje, continua sendo uma característica distintiva. Com o passar dos anos, várias empresas licenciaram o software FastTrack, entre elas Music City (Morpheus), Grokster e iMesh. A FastTrack, acionada judicialmente por violação de direitos autorais pela Recording Industry Association of America (RIAA) e pelos tribunais holandeses, vendeu uma grande parte da empresa para a Sharman Networks em fevereiro de 2002. A estrutura corporativa resultante é complexa e confusa. A Sharman Networks é incorporada em Vanuatu, um minúsculo país insular no Oceano Pacífico. A sede da empresa fica na Austrália. E muitos dos desenvolvedores de software estão na Estônia! Essa estrutura corporativa complexa e confusa não facilita em nada o trabalho das organizações norte-americanas que gostariam de acionar o KaZaA judicialmente.

Uma questão interessante é se a rede KaZaA continuaria a prosperar, mesmo que as empresas responsáveis por ela fechassem (por razões legais ou financeiras, por exemplo). Como acabamos de aprender, o KaZaA emprega uma arquitetura altamente descentralizada, que não depende de nenhum servidor central. É muito provável que continue sozinho, de uma forma ou de outra. A indústria fonográfica está constantemente buscando meios alternativos de reduzir o compartilhamento de arquivos pelo KaZaA, inclusive acionando judicialmente usuários individuais da rede e contratando empresas para espalhar arquivos falsos [Overpeer, 2004].

Finalmente, nós (os autores deste livro) prevemos o eventual surgimento de sistemas sem fio de compartilhamento de arquivos P2P no qual indivíduos — tal como estudantes secundários — terão PDAs capacitados com gigabytes de armazenamento e comunicação Wi-Fi (sem fio). Esses equipamentos, normalmente escondidos em mochilas escolares, funcionarão em modo *ad hoc*, permitindo que dispositivos próximos consultem uns aos outros e troquem conteúdo desapercebidamente pelo ar! É evidente que será difícil para a RIAA ou organizações similares monitorar ou interferir nesses sistemas P2P sem fio.

2.7 Programação e desenvolvimento de aplicações com TCP

Agora que já examinamos várias importantes aplicações de rede, vamos explorar como são escritos programas de aplicação de rede. Nesta seção escreveremos programas de aplicações que usam TCP; na seção seguinte, escreveremos programas que usam UDP.

Lembre-se de que na Seção 2.1 dissemos que muitas aplicações de rede consistem em um par de programas — um programa cliente e um programa servidor — que residem em dois sistemas finais diferentes. Quando esses programas são executados, criam-se um processo cliente e um processo servidor, que se comunicam entre si lendo de seus sockets e escrevendo através deles. Ao criar uma aplicação de rede, a tarefa principal do programador é escrever o código tanto para o programa cliente como para o programa servidor.

Há dois tipos de aplicações de rede. Um deles é uma *implementação* de um protocolo padrão definido, por exemplo, em um RFC. Para essa implementação, os programas, cliente e servidor, devem obedecer às regras ditadas pelo RFC. Por exemplo, o programa cliente poderia ser uma implementação do lado do clien-

te do protocolo FTP descrito na Seção 2.3 e definido explicitamente no RFC 959 e o programa servidor, uma implementação do protocolo de servidor FTP também descrito explicitamente no RFC 959. Se um programador escrever codificação para o programa cliente e um outro programador independente escrever uma codificação para o programa servidor e ambos seguirem cuidadosamente as regras do RFC, então os dois programas poderão interagir. Realmente, muitas das aplicações de rede de hoje envolvem comunicação entre programas cliente e servidor que foram criados por programadores diferentes — por exemplo, um browser Netscape que se comunica com um servidor Web Apache, ou um cliente FTP em um PC que carrega um arquivo em um servidor FTP UNIX. Quando um programa, cliente ou servidor, implementa um protocolo definido em um RFC, deve usar o número de porta associado com o protocolo. (Números de portas foram discutidos brevemente na Seção 2.1. Serão examinados mais detalhadamente no Capítulo 3.)

O outro tipo de aplicação cliente–servidor é uma aplicação *proprietária*. Nesse caso, o protocolo de camada de aplicação utilizado pelos programas cliente e servidor não obedecem necessariamente a nenhum RFC existente. Um único programador (ou equipe de desenvolvimento) cria ambos os programas cliente e servidor, e tem completo controle sobre o que entra no código. Mas, como a codificação não implementa um protocolo de domínio público, outros programadores independentes não poderão desenvolver programas que interajam com a aplicação. Ao desenvolver uma aplicação proprietária, o programador deve ter o cuidado de não usar um dos números de porta bem conhecidos, definidos em RFCs.

Nesta seção e na próxima, examinaremos as questões fundamentais do desenvolvimento de uma aplicação cliente–servidor proprietária. Durante a fase de desenvolvimento, uma das primeiras decisões que o programador deve tomar é se a aplicação rodará em TCP ou UDP. Lembre-se de que o TCP é orientado para conexão e provê um *canal confiável de cadeia de bytes*, pelo qual fluem dados entre dois sistemas finais. O UDP não é orientado para conexão e envia *pacotes* de dados independentes de um sistema final ao outro, sem nenhuma garantia de entrega.

Nesta seção, desenvolveremos uma aplicação cliente simples que roda em TCP; na seção seguinte, desenvolveremos uma aplicação cliente simples que roda em UDP. Apresentaremos essas aplicações TCP e UDP simples em Java. Poderíamos escrevê-las em linguagem C ou C++, mas optamos por Java por diversas razões. Em primeiro lugar, o código das aplicações é mais elegante e mais limpo em Java; com Java, há menos linhas de codificação e cada uma delas pode ser explicada a programadores iniciantes sem muita dificuldade. Em segundo lugar, programação cliente–servidor em Java está se tornando cada vez mais popular e poderá até ser a norma nos próximos anos. Mas não precisa ficar assustado se não estiver familiarizado com a linguagem Java. Você conseguirá acompanhar a codificação se tiver experiência de programação em outra linguagem.

Para leitores interessados em programação cliente-servidor em linguagem C, há várias boas referências à disposição [Donahoo, 2000; Stevens, 1997; Frost, 1994; Kurose, 1996].

2.7.1 Programação de aplicações com TCP

Comentamos na Seção 2.1 que processos que rodam em máquinas diferentes se comunicam uns com os outros enviando mensagens para sockets. Dissemos que cada processo é análogo a uma casa e que o socket do processo é análogo a uma porta. Como ilustrado na Figura 2.26, o socket é a porta entre o processo da aplicação e o TCP. O desenvolvedor da aplicação controla tudo que está no lado da camada de aplicação da porta; contudo, tem pouco controle do lado da camada de transporte. (No máximo, poderá fixar alguns parâmetros do TCP, tais como tamanho máximo do buffer e tamanho máximo de segmentos.)

Agora, vamos examinar mais de perto a interação dos programas cliente e servidor. O cliente tem a tarefa de iniciar contato com o servidor. Para que o servidor possa reagir ao contato inicial do cliente, tem de estar pronto, o que implica duas coisas. Em primeiro lugar, o programa servidor não pode estar inativo, isto é, tem de estar rodando como um processo antes de o cliente tentar iniciar contato. Em segundo lugar, o programa tem de ter alguma porta — mais precisamente, um socket — que acolha algum contato inicial de um processo cliente que esteja rodando em uma máquina qualquer. Recorrendo à nossa analogia casa/porta para processo/socket, às vezes, nos referiremos ao contato inicial do cliente como 'bater à porta'.

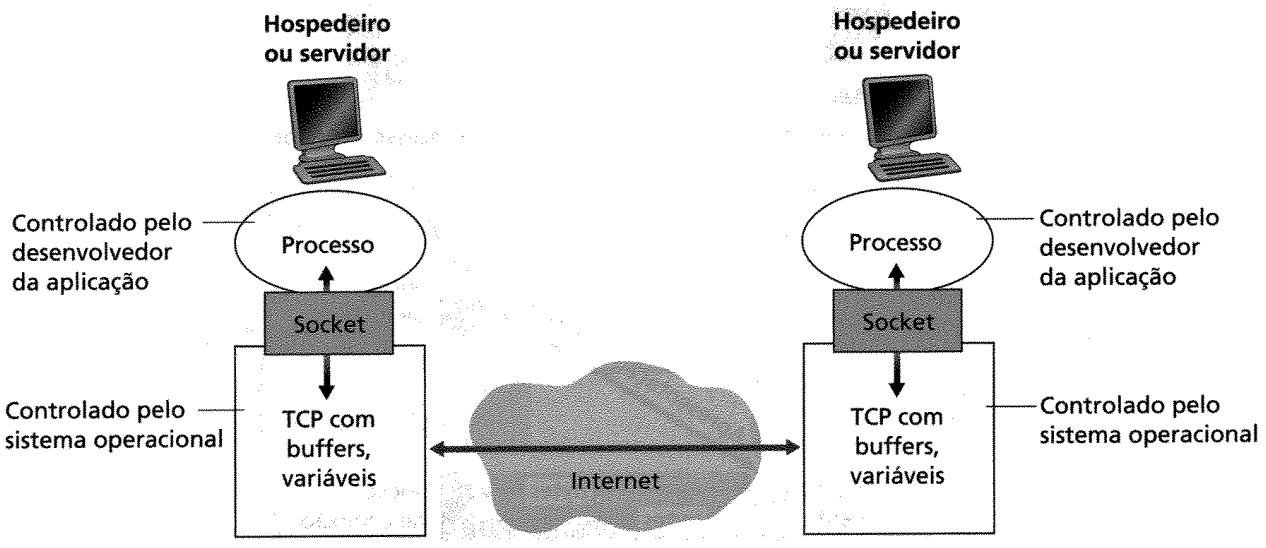


Figura 2.26 Processos que se comunicam através de sockets TCP

Com o processo servidor em execução, o processo cliente pode iniciar uma conexão TCP com o servidor, o que é feito no programa cliente pela criação de um socket. Quando cria seu socket, o cliente especifica o endereço do processo servidor, a saber, o endereço IP do hospedeiro servidor e o número de porta do processo servidor. Com a criação do socket no programa cliente, o TCP no cliente inicia uma apresentação de três vias e estabelece uma conexão TCP com o servidor. A apresentação de três vias é completamente transparente para os programas cliente e servidor.

Durante a apresentação de três vias, o processo cliente bate no socket de entrada do processo servidor. Quando o servidor 'ouve' a batida, cria uma nova porta (mais precisamente, um novo socket) dedicada àquele cliente específico. No exemplo a seguir, a porta de entrada é um objeto `ServerSocket` que denominamos `welcomeSocket`. Quando um cliente bate nesse socket, o programa chama o método `accept( )` do `welcomeSocket`, que cria um novo socket para o cliente. Ao final da fase de apresentação, existe uma conexão TCP entre o socket do cliente e o novo socket do servidor. Daqui em diante, vamos nos referir ao novo socket dedicado do servidor como **socket de conexão** do servidor.

Da perspectiva da aplicação, a conexão TCP é uma tubulação virtual direta entre o socket do cliente e o socket de conexão do servidor. O processo cliente pode enviar bytes para seu socket arbitrariamente; o TCP garante que o processo servidor receberá (através do socket de conexão) cada byte na ordem em que foram enviados. Assim, o TCP provê um **serviço confiável de corrente de bytes** entre os processos cliente e servidor. Além disso, exatamente como pessoas podem entrar e sair pela mesma porta, o processo cliente não somente envia bytes a seu socket, mas também os recebe dele; de modo semelhante, o processo servidor não só recebe bytes de seu socket de conexão, mas também os envia por ele. Isso é ilustrado na Figura 2.27. Como sockets desempenham um papel central em aplicações cliente–servidor, o desenvolvimento dessas aplicações também é denominado programação de sockets.

Antes de apresentarmos nosso exemplo de aplicação cliente–servidor, é útil discutirmos a noção de cadeia. Uma **cadeia** é uma seqüência de caracteres que fluem para dentro ou para fora de um processo. Cada uma é, para o processo, uma cadeia de entrada ou uma cadeia de saída. Se a cadeia for de **entrada**, estará ligada a alguma fonte de entrada para o processo, tal como uma entrada-padrão (o teclado) ou um socket para o qual fluem dados vindos da Internet. Se a cadeia for de **saída**, estará ligada a alguma fonte de saída para o processo, tal como uma saída padrão (o monitor) ou um socket do qual fluem dados para a Internet.

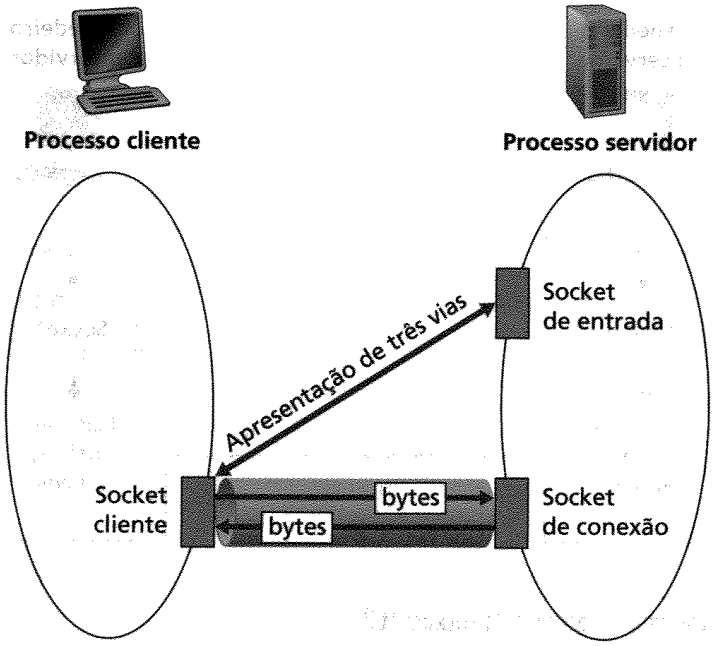


Figura 2.27 Socket cliente, socket de entrada e socket de conexão

2.7.2 Um exemplo de aplicação cliente–servidor em Java

Usaremos a seguinte aplicação cliente–servidor simples para demonstrar programação de sockets para TCP e UDP:

1. Um cliente lê uma linha a partir de sua **entrada padrão** (teclado) e a envia através de seu socket para o servidor.
2. O servidor lê uma linha a partir de seu socket de conexão.
3. O servidor converte a linha para letras maiúsculas.
4. O servidor envia a linha modificada ao cliente através de seu socket de conexão.
5. O cliente lê a linha modificada através de seu socket e apresenta a linha na sua **saída padrão** (monitor).

A Figura 2.28 ilustra a principal atividade do cliente e do servidor relacionada ao socket.

Em seguida, forneceremos o par de programas cliente-servidor para uma implementação da aplicação utilizando TCP. Apresentaremos também uma análise detalhada, linha a linha, após cada programa. O programa cliente será denominado `TCPClient.java` e o programa servidor, `TCPServer.java`. Para dar ênfase às questões fundamentais, forneceremos, intencionalmente, uma codificação objetiva, mas não tão à prova de fogo. Uma 'boa codificação' certamente teria algumas linhas auxiliares a mais.

Tão logo os dois programas estejam compilados em seus respectivos hospedeiros, o programa servidor será primeiramente executado no hospedeiro servidor, o que criará nele um processo servidor.

Como discutido anteriormente, o processo servidor espera para ser contatado por um processo cliente. Nesse exemplo de aplicação, quando o programa cliente é executado, um processo é criado no cliente, e esse processo imediatamente contata o servidor e estabelece uma conexão TCP com ele. O usuário no cliente pode então usar a aplicação para enviar uma linha e, em seguida, receber uma versão dessa linha em letras maiúsculas.

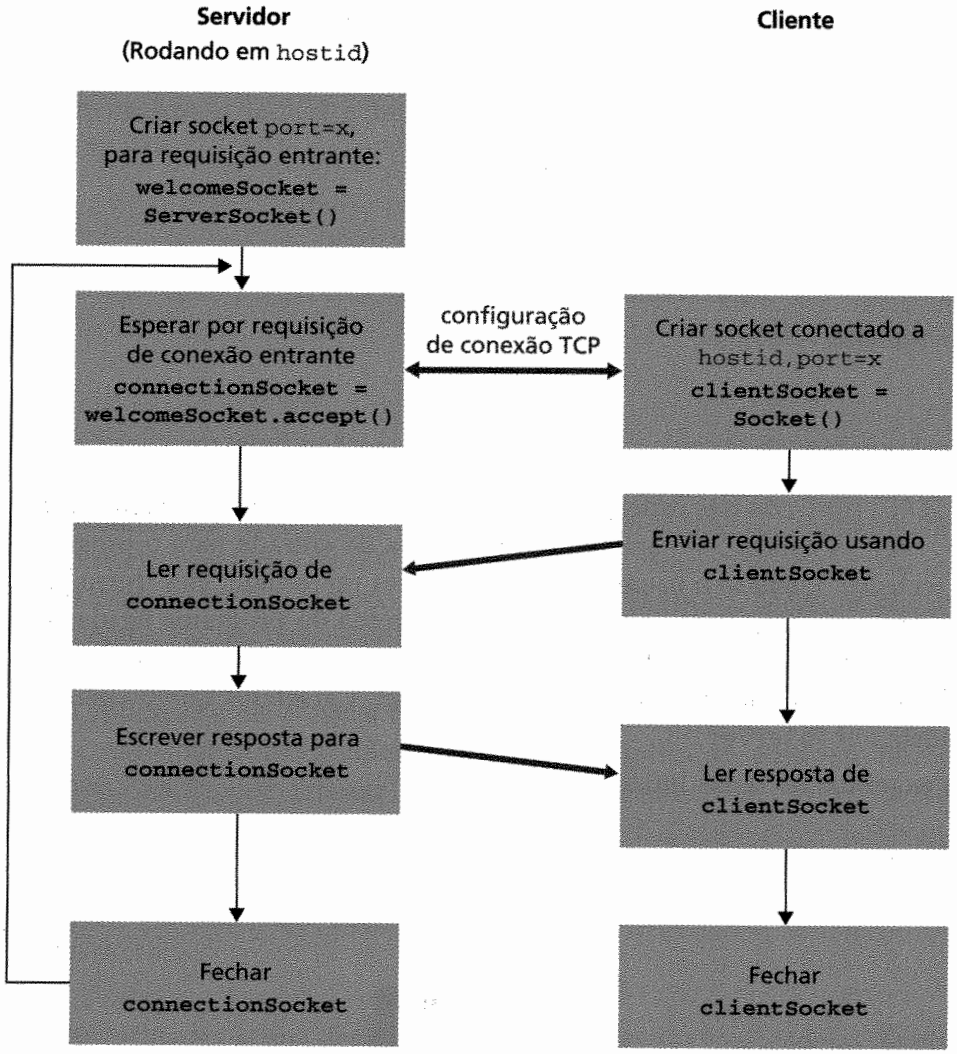


Figura 2.28 A aplicação cliente–servidor, usando serviços de transporte orientados à conexão

TCPClient.java

Eis a codificação para o lado cliente da aplicação:

```
import java.io.*;
import java.net.*;
class TCPClient {
    public static void main(String argv[]) throws Exception
    {
        String sentence;
        String modifiedSentence;
        BufferedReader inFromUser = new BufferedReader(
                new InputStreamReader(System.in));
        Socket clientSocket = new Socket("hostname", 6789);
        DataOutputStream outToServer = new DataOutputStream(
                clientSocket.getOutputStream());
```

```
        BufferedReader inFromServer =
            new BufferedReader(new InputStreamReader(
                clientSocket.getInputStream()));
        sentence = inFromUser.readLine();
        outToServer.writeBytes(sentence + '\n');
        modifiedSentence = inFromServer.readLine();
        System.out.println("FROM SERVER: " +
                           modifiedSentence);
        clientSocket.close();
    }
}
```

O programa `TCPClient` cria três cadeias e um socket, como mostrado na Figura 2.29. O socket é denominado `clientSocket`. A cadeia `inFromUser` é uma cadeia de entrada para o programa ligada à entrada padrão (isto é, o teclado). Quando o usuário digita caracteres no teclado, eles fluem para dentro da cadeia `inFromUser`. A cadeia `inFromServer` é outra cadeia de entrada do programa ligada ao socket. Caracteres que chegam da rede fluem para dentro da cadeia `inFromServer`. Finalmente, a cadeia `outToServer` é uma cadeia de saída do programa que também é ligada ao socket. Caracteres que o cliente envia à rede fluem para dentro da cadeia `outToServer`.

Vamos agora examinar as várias linhas da codificação.

```
import java.io.*;
import java.net.*;
```

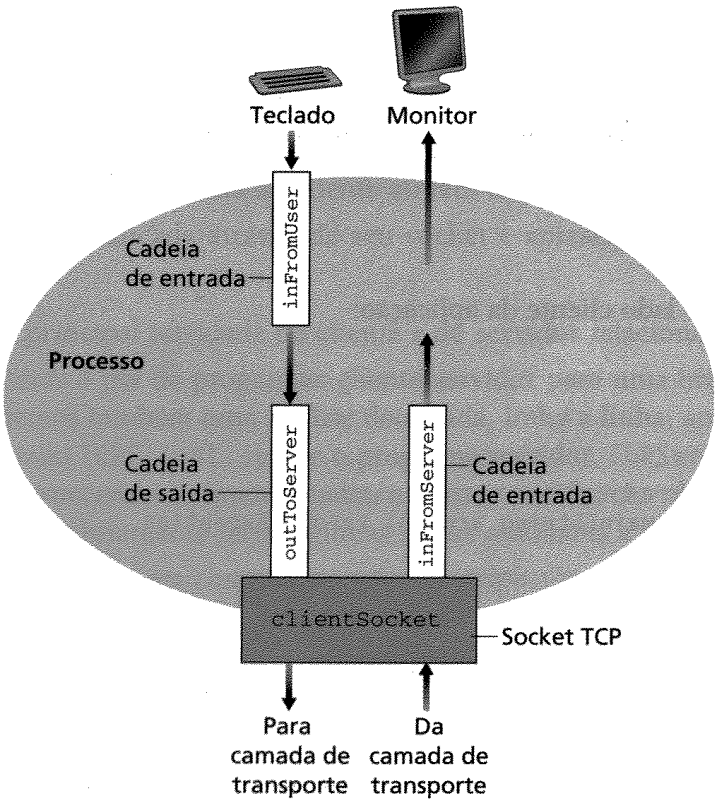


Figura 2.29 `TCPClient` tem três cadeias através das quais fluem caracteres

`java.io` e `java.net` são pacotes Java. O pacote `java.io` contém classes para cadeias de entrada e de saída. Em particular, contém as classes `BufferedReader` e `DataOutputStream` — classes que o programa usa para criar as três cadeias previamente ilustradas. O pacote `java.net` provê classes para suporte de rede. Em particular, contém as classes `Socket` e `ServerSocket`. O objeto `clientSocket` desse programa é derivado da classe `Socket`.

```
class TCPClient {
    public static void main(String argv[]) throws Exception
            {.......}
}
```

Até aqui, o que vimos é material padronizado que você vê no início da maioria das codificações Java. A terceira linha é o começo de um bloco de definição de classe. A palavra-chave `class` inicia a definição de classe para a classe denominada `TCPClient`. Uma classe contém variáveis e métodos, limitados pelas chaves { } que iniciam e encerram o bloco de definição de classe. A classe `TCPClient` não tem nenhuma variável de classe e possui exatamente um método, que é o método `main( )`. Métodos são semelhantes às funções ou aos procedimentos em linguagens como C; o método principal na linguagem Java é semelhante à função principal em C e C++. Quando o interpretador Java executa uma aplicação (ao ser chamado pela classe de controle da aplicação), ele começa chamando o método principal da classe. O método principal então chama todos os outros métodos exigidos para executar a aplicação. Para essa introdução à programação de portas em Java, você pode ignorar as palavras-chave `public`, `static`, `void`, `main` e `throws Exceptions` (embora deva incluí-las no código).

```
String sentence;
String modifiedSentence;
```

As duas linhas apresentadas acima declaram objetos do tipo `String` (cadeia). O objeto `sentence` é a cadeia digitada pelo usuário e enviada ao servidor. O objeto `modifiedSentence` é a cadeia obtida do servidor e enviada à saída padrão do usuário.

```
BufferedReader inFromUser = new BufferedReader(new InputStreamReader(System.in));
```

Esta linha cria o objeto de cadeia `inFromUser` do tipo `BufferedReader`. A cadeia de entrada é inicializada com `System.in`, que vincula a cadeia à entrada padrão. O comando permite que o cliente leia o texto de seu teclado.

```
Socket clientSocket = new Socket("hostname", 6789);
```

Esta linha cria o objeto `clientSocket` do tipo `Socket`. Ela também ativa a conexão TCP entre cliente e servidor. A cadeia `hostname` deve ser substituída pelo nome de hospedeiro do servidor (por exemplo, `apple.poly.edu`). Antes de a conexão TCP ser realmente iniciada, o cliente realiza um procedimento de consulta ao DNS do nome de hospedeiro para obter o endereço IP da máquina. O número 6789 é o número de porta. Você pode usar um número de porta diferente, mas precisa obrigatoriamente usar o mesmo do lado servidor da aplicação. Como discutimos anteriormente, o processo servidor é identificado pelo endereço IP do hospedeiro juntamente com o número de porta da aplicação.

```
DataOutputStream outToServer =
    new DataOutputStream(clientSocket.getOutputStream());
```

```
BufferedReader inFromServer =
    new BufferedReader(new inputStreamReader(
                        clientSocket.getInputStream()));
```

Estas linhas criam objetos de cadeia que são ligados ao socket. A cadeia `outToServer` fornece a saída do processo para o socket. A cadeia `inFromServer` fornece ao processo a entrada do socket (veja a Figura 2.29).

```
sentence = inFromUser.readLine();
```

Esta linha coloca uma linha digitada pelo usuário na cadeia `sentence`. A cadeia sentence continua a juntar caracteres até que o usuário termine a linha digitando 'carriage return'. A linha passa da entrada padrão para dentro da cadeia `sentence` por meio da cadeia `inFromUser`.

```
outToServer.writeBytes(sentence + '\n');
```

Esta linha envia a cadeia `sentence` para dentro da corrente `outToServer` ampliada com um 'carriage return'. A sentença ampliada flui pelo socket do cliente para dentro da conexão TCP. O cliente então espera para receber caracteres do servidor.

```
modifiedSentence = inFromServer.readLine();
```

Quando chegam do servidor, os caracteres fluem através da cadeia `inFromServer` e são colocados dentro da cadeia `modifiedSentence`. Continuam a se acumular em `modifiedSentence` até que a linha termine com um caractere de 'carriage return'.

```
System.out.println("FROM SERVER " + modifiedSentence);
```

Esta linha envia para o monitor a cadeia `modifiedSentence` retornada pelo servidor.

```
clientSocket.close();
```

Esta última linha fecha o socket e, por conseguinte, a conexão TCP entre o cliente e o servidor. Ela faz com que o TCP no cliente envie uma mensagem para o TCP no servidor (veja a Seção 3.5).

TCPServer.java

Agora vamos examinar o programa servidor.

```
import java.io.*;
import java.net.*;
class TCPServer {
    public static void main(String argv[]) throws Exception
        {
            String clientSentence;
            String capitalizedSentence;
            ServerSocket welcomeSocket = new ServerSocket
```

```
            (6789);
        while(true) {
            Socket connectionSocket = welcomeSocket.
                accept();
            BufferedReader inFromClient =
                new BufferedReader(new InputStreamReader(
                    connectionSocket.getInputStream()));
            DataOutputStream outToClient =
                new DataOutputStream(
                    connectionSocket.getOutputStream());
            clientSentence = inFromClient.readLine();
            capitalizedSentence =
                    clientSentence.toUpperCase() + '\n';
            outToClient.writeBytes(capitalizedSentence);
        }
    }
}
```

TCPServer tem muitas semelhanças com TCPClient. Vamos agora dar uma olhada nas linhas em TCPServer.java. Não comentaremos as linhas que são idênticas ou semelhantes aos comandos em TCPClient.java.

A primeira linha em TCPServer é substancialmente diferente da que vimos em TCPClient:

```
ServerSocket welcomeSocket = new ServerSocket(6789);
```

Esta linha cria o objeto welcomeSocket, que é do tipo ServerSocket. O welcomeSocket é uma espécie de porta que fica 'ouvindo', à espera que algum cliente bata. O welcomeSocket fica à escuta na porta número 6789. A linha seguinte é:

```
Socket connectionSocket = welcomeSocket.accept();
```

Esta linha cria um novo socket, denominado connectionSocket, quando algum cliente bate à porta welcomeSocket. O número de porta desse socket também é 6789. (Explicaremos por que ambos têm o mesmo número de porta no Capítulo 3). O TCP então estabelece uma conexão virtual direta entre clientSocket no cliente e connectionSocket no servidor. O cliente e o servidor podem então enviar bytes um para o outro pela conexão, e todos os bytes enviados chegam ao outro lado na ordem certa. Com connectionSocket estabelecido, o servidor pode continuar à escuta por outros clientes que requisitarão a aplicação usando welcomeSocket. (Essa versão do programa, na verdade, não fica à escuta por mais requisições de conexão, mas pode ser modificada com threads para fazer isso.) O programa então cria diversos objetos de cadeia, análogos aos objetos de cadeia em clientSocket. Considere agora:

```
capitalizedSentence = clientSentence.toUpperCase() + '\n';
```

Este comando é o coração da aplicação. Ele toma a linha enviada pelo cliente, passa todas as letras para maiúsculas e adiciona um 'carriage return'. Ele usa o método toUpperCase(). Todos os outros comandos no programa são periféricos, usados para a comunicação com o cliente.

Para testar o par de programas, instale e compile `TCPClient.java` em um hospedeiro e `TCPServer.java` em outro hospedeiro. Não se esqueça de incluir o nome de hospedeiro do servidor adequado em `TCPClient.java`. Então, execute `TCPServer.class`, o programa servidor compilado, no servidor. Isso cria um processo no servidor que fica ocioso até ser contatado por algum cliente. Então, execute `TCPClient.class`, o programa cliente compilado, no cliente. Isso cria um processo no cliente e estabelece uma conexão TCP entre os processos cliente e servidor. Por fim, para usar a aplicação, digite uma sentença seguida de um 'carriage return'.

Para desenvolver sua própria aplicação cliente–servidor, você pode começar modificando ligeiramente os programas. Por exemplo, em vez de converter todas as letras para maiúsculas, o servidor poderia contar o número de vezes que a letra 's' aparece no texto e retornar esse número.

2.8 Programação de sockets com UDP

Aprendemos na seção anterior que, quando dois processos se comunicam por TCP, é como se houvesse uma tubulação entre eles, que permanece ativa até que um dos dois processos a feche. Quando um dos processos quer enviar alguns bytes para o outro processo, simplesmente insere os bytes na tubulação. O processo de envio não tem de acrescentar um endereço de destino aos bytes porque a tubulação está ligada logicamente ao destino. Além disso, a tubulação provê um canal confiável de cadeia de bytes — a seqüência de bytes recebida pelo processo receptor é exatamente a mesma que o remetente inseriu na tubulação.

O UDP também permite que dois (ou mais) processos que rodam em hospedeiros diferentes se comuniquem. Contudo, é diferente do TCP de muitas maneiras fundamentais. Primeiramente, é um serviço não orientado para conexão — não há uma fase inicial de apresentação, durante a qual é estabelecida uma tubulação entre os dois processos. Como o UDP não tem uma tubulação, quando um processo quer enviar um conjunto de bytes a outro, o processo remetente deve anexar o endereço do processo destinatário ao conjunto de bytes. E isso precisa ser feito para cada conjunto de bytes que o processo remetente enviar. Como analogia, considere um grupo de 20 pessoas que toma cinco táxis para um mesmo destino; quando cada um dos grupos entra em um carro, tem de informar separadamente ao motorista o endereço de destino. Assim, o UDP é semelhante a um serviço de táxi. O endereço de destino é uma tupla que consiste no endereço IP do hospedeiro destinatário e no número de porta do processo destinatário. Vamos nos referir ao conjunto de bytes de informação, juntamente com o endereço IP do destinatário e o número de porta, como 'pacote'. O UDP provê um modelo de serviço não confiável orientado para mensagem, no sentido de que faz o melhor esforço para entregar o conjunto de bytes ao destino, mas não garante que esse conjunto seja, de fato, entregue. O serviço UDP contrasta acentuadamente (em vários aspectos) com o modelo de serviço confiável de cadeia de bytes do TCP.

Após ter criado um pacote, o processo remetente empurra-o para dentro da rede através de um socket. Continuando com nossa analogia do táxi, do outro lado do socket remetente há um táxi esperando pelo pacote. Esse táxi, então, leva o pacote até seu endereço de destino. Contudo, não garante que eventualmente entregará o pacote no seu destino final, pois pode quebrar ou sofrer algum outro problema não previsto no caminho. Em outras palavras, o UDP *provê serviço não confiável de transporte a seus processos de comunicação* — não dá nenhuma garantia de que um datagrama ('pacote') alcançará seu destino final.

Nesta seção, ilustraremos programação de socket desenvolvendo novamente a mesma aplicação da seção anterior, mas, dessa vez, com UDP. Veremos que a codificação para UDP é diferente da codificação para TCP de muitas maneiras importantes. Em particular, (1) não há apresentação inicial entre os dois processos e, portanto, não há necessidade de um socket de entrada, (2) não há cadeias ligadas aos sockets, (3) os hospedeiros remetentes criam pacotes anexando o endereço IP do destinatário e o número de porta a cada conjunto de bytes que enviam e (4) o processo destinatário deve desmontar cada pacote recebido para obter os bytes de informação do pacote.

Lembre-se, mais uma vez, da nossa aplicação simples:

1. Um cliente lê uma linha a partir de sua entrada padrão (teclado) e a envia através de seu socket para o servidor.
2. O servidor lê uma linha a partir de seu socket.
3. O servidor converte a linha para letras maiúsculas.
4. O servidor envia a linha modificada através de seu socket ao cliente.
5. O cliente lê a linha modificada através de seu socket e apresenta a linha em sua saída padrão (monitor).

A Figura 2.30 destaca a principal atividade relacionada ao socket realizada pelo cliente e pelo servidor, que se comunicam por meio de um serviço de transporte (UDP) não orientado para conexão.

UDPClient.java

Eis a codificação para o lado cliente da aplicação:

```
import java.io.*;
import java.net.*;
class UDPClient {
    public static void main(String args[]) throws Exception
    {
        BufferedReader inFromUser =
            new BufferedReader(new InputStreamReader
                    (System.in));
        DatagramSocket clientSocket = new DatagramSocket();
        InetAddress IPAddress =
                    InetAddress.getByName("hostname");
        byte[] sendData = new byte[1024];
        byte[] receiveData = new byte[1024];
        String sentence = inFromUser.readLine();
        sendData = sentence.getBytes();
        DatagramPacket sendPacket =
            new DatagramPacket(sendData, sendData.length,
                    IPAddress, 9876);
        clientSocket.send(sendPacket);
        DatagramPacket receivePacket =
            new DatagramPacket(receiveData,
                    receiveData.length);
        clientSocket.receive(receivePacket);
        String modifiedSentence =
            new String(receivePacket.getData());
        System.out.println("FROM SERVER:" +
                    modifiedSentence);
        clientSocket.close();
    }
}
```

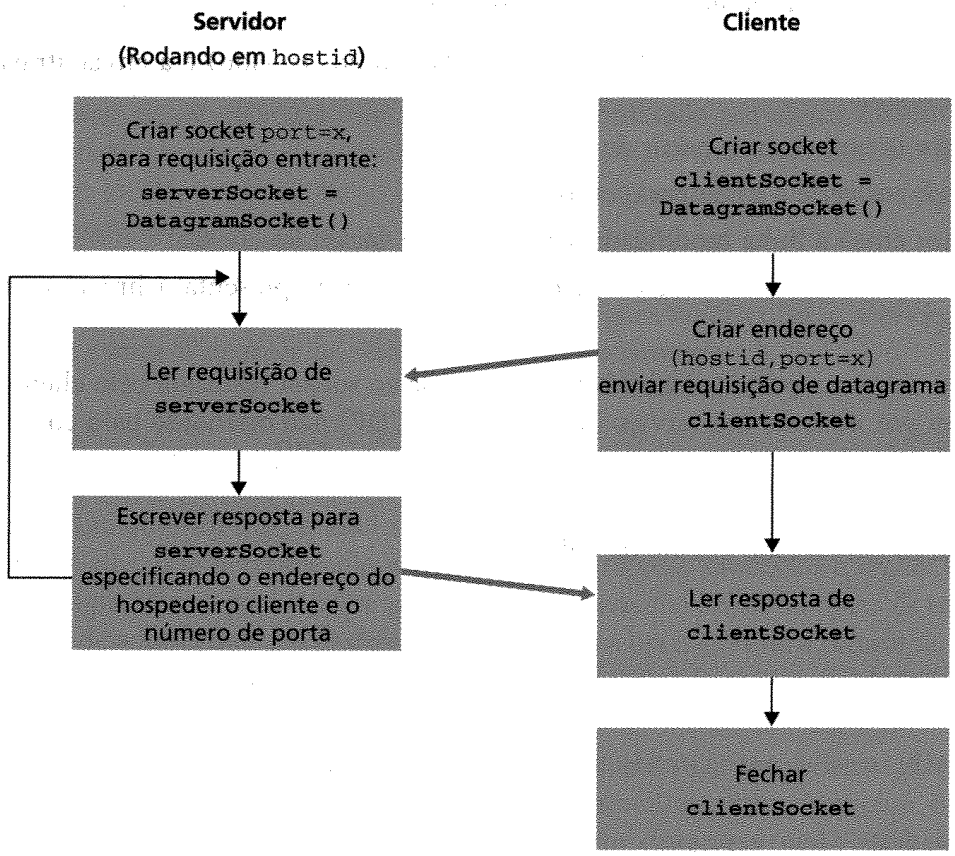


Figura 2.30 A aplicação cliente-servidor usando serviços de transporte não orientados para conexão

O programa `UDPClient.java` constrói uma cadeia e um socket, como mostra a Figura 2.31. O socket é denominado `clientSocket` e é do tipo `DatagramSocket`. Note que o UDP usa, no cliente, um tipo de socket diferente do usado no TCP. Em particular, com UDP nosso cliente usa um `DatagramSocket`, enquanto com TCP ele usou um `Socket`. A cadeia `inFromUser` é uma cadeia de entrada para o programa; está ligada à entrada padrão, isto é, ao teclado. Tínhamos uma cadeia equivalente em nossa versão TCP do programa. Quando o usuário digita caracteres no teclado, esses caracteres fluem para dentro da cadeia `inFromUser`. Porém, ao contrário do TCP, não há cadeias (de entrada e de saída) ligadas ao socket. Em vez de alimentar bytes à cadeia ligada a um objeto `Socket`, o UDP transmite pacotes individuais por meio do objeto `DatagramSocket`.

Agora, vamos examinar as linhas da codificação de `TCPClient.java` que são significativamente diferentes.

```
DatagramSocket clientSocket = new DatagramSocket();
```

Esta linha cria o objeto `clientSocket` do tipo `DatagramSocket`. Ao contrário do `TCPClient.java`, ela não ativa uma conexão TCP. Em particular, o hospedeiro cliente não contata o hospedeiro servidor durante a execução desta linha. Por essa razão, o construtor `DatagramSocket()` não toma como argumento o nome do servidor ou o número da porta. Usando nossa analogia porta–tubulação, a execução da linha acima cria um socket para o processo cliente, mas não cria uma tubulação entre os dois processos.

```
InetAddress IPAddress = InetAddress.getByName("hostname");
```

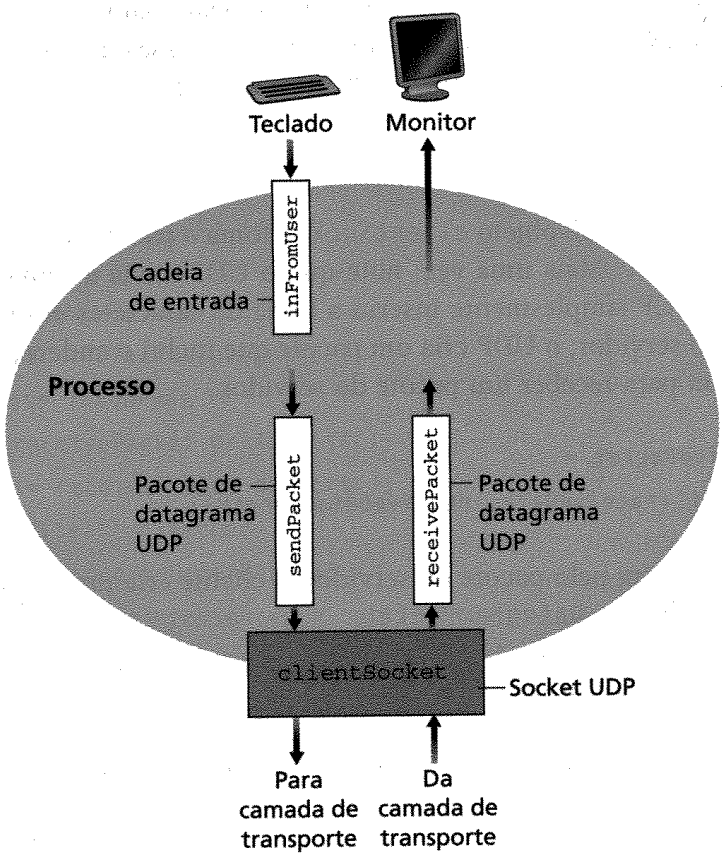


Figura 2.31 `UDPClient` tem uma corrente; o socket aceita pacotes do processo e entrega pacotes ao processo

Para enviar bytes a um processo destinatário, precisamos do endereço do processo. Parte desse endereço é o endereço IP do hospedeiro destinatário. A linha apresentada invoca uma consulta ao DNS que traduz o nome do destinatário (nesse exemplo, fornecido na codificação pelo programador) para um endereço IP. O DNS também foi chamado pela versão TCP do cliente, embora o tenha feito de maneira implícita, e não explícita. O método `getByName()` toma como argumento o nome de hospedeiro do servidor e retorna o endereço IP desse mesmo servidor. Coloca esse endereço no objeto `IPAddress` do tipo `InetAddress`.

```
byte[] sendData = new byte[1024];
byte[] receiveData = new byte[1024];
```

Os vetores de bytes `sendData` e `receiveData` reterão os dados que o cliente envia e recebe, respectivamente.

```
sendData = sentence.getBytes();
```

Esta linha realiza, essencialmente, uma conversão de tipo. Pega a cadeia sentence e a renomeia como `sendData`, que é um vetor de bytes.

```
DatagramPacket sendPacket =new DatagramPacket(
sendData, sendData.length, IPAddress, 9876);
```

Esta linha constrói o pacote, sendPacket, que o cliente enviará para a rede através do seu socket. Esse pacote inclui os dados contidos nele, sendData, o comprimento desses dados, o endereço IP do servidor e o número de porta da aplicação (escolhemos 9876). Note que sendPacket é do tipo DatagramPacket.

```
clientSocket.send(sendPacket);
```

Nesta linha, o método send() do objeto clientSocket toma o pacote recém-construído e passa-o para a rede através de clientSocket. Mais uma vez, note que o UDP envia a linha de caracteres de maneira muito diferente do TCP. O TCP simplesmente inseriu a linha de caracteres em uma cadeia que tinha uma conexão lógica direta com o servidor; o UDP cria um pacote que inclui o endereço do servidor. Após enviar esse pacote, o cliente espera para receber um pacote do servidor.

```
DatagramPacket receivePacket =
   new DatagramPacket (receiveData, receiveData.length);
```

Nesta linha, enquanto espera pelo pacote do servidor, o cliente cria um lugar reservado para o pacote, receivePacket, um objeto do tipo DatagramPacket.

```
clientSocket.receive(receivePacket);
```

O cliente fica ocioso até receber um pacote; quando enfim o recebe, ele o coloca em receivePacket.

```
String modifiedSentence =
   new String(receivePacket.getData());
```

Esta linha extrai os dados de receivePacket e realiza uma conversão de tipo, convertendo um vetor de bytes na cadeia modifiedSentence.

```
System.out.println("FROM SERVER:" + modifiedSentence);
```

Esta linha, que também está presente no TCPClient, apresenta a cadeia modifiedSentence no monitor do cliente.

```
clientSocket.close();
```

Esta última linha fecha o socket. Como o UDP não é orientado para conexão, esta linha não faz com que o cliente envie uma mensagem de camada de transporte ao servidor (ao contrário do TCPClient).

UDPServer.java

Vamos agora dar uma olhada no lado servidor da aplicação:

```
import java.io.*;
import java.net.*;
class UDPServer {
    public static void main(String args[]) throws Exception
        {
        DatagramSocket serverSocket = new
                        DatagramSocket(9876);
        byte[] receiveData = new byte[1024];
```

```
        byte[] sendData = new byte[1024];
        while (true)
            {
            DatagramPacket receivePacket =
                    new DatagramPacket(receiveData,
                        receiveData.length);
            serverSocket.receive(receivePacket);
            String sentence = new String(
                    receivePacket.getData());
            InetAddress IPAddress =
                    receivePacket.getAddress();
            int port = receivePacket.getPort();
            String capitalizedSentence =
                    sentence.toUpperCase();
            sendData = capitalizedSentence.getBytes();
            DatagramPacket sendPacket =
            new DatagramPacket (sendData,
                    sendData.length, IPAddress, port);
            serverSocket.send(sendPacket);
        }
    }
}
```

O programa `UDPServer.java` constrói um socket, como mostra a Figura 2.32. O socket é denominado `serverSocket`. É um objeto do tipo `DatagramSocket`, como era o socket do lado cliente da aplicação. Mais uma vez, nenhuma cadeia está ligada ao socket.

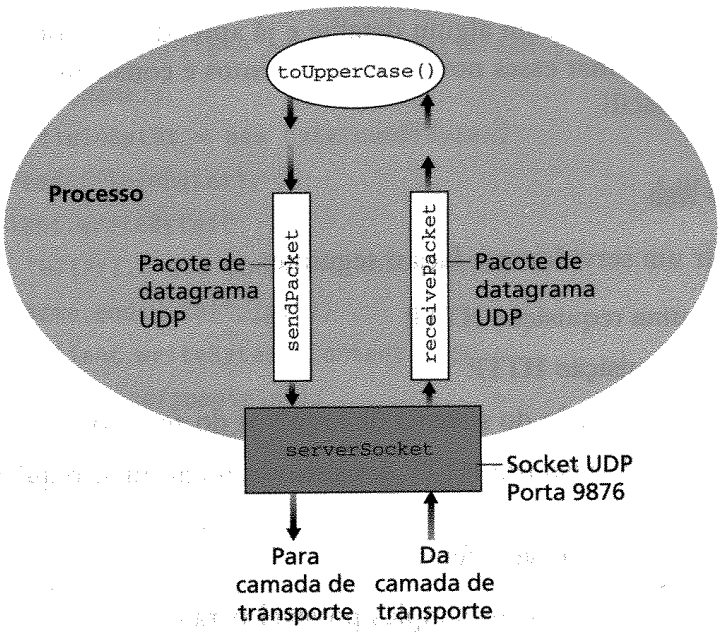


Figura 2.32 `UDPServer` não tem correntes; o socket aceita pacotes do processo e entrega pacotes ao processo

Vamos agora examinar as linhas da codificação que são diferentes de `TCPServer.java`.

```
DatagramSocket serverSocket = new DatagramSocket (9876);
```

Esta linha constrói o `DatagramSocket serverSocket` na porta 9876. Todos os dados enviados e recebidos passarão através desse socket. Como o UDP não é orientado para conexão, não temos de criar um novo socket e continuar à escuta de novas requisições de conexão, como é feito no `TCPServer.java`. Se vários clientes acessarem essa aplicação, todos enviarão seus pacotes por esse único socket, `serverSocket`.

```
String sentence = new String(receivePacket.getData());
InetAddress IPAddress = receivePacket.getAddress();
int port = receivePacket.getPort();
```

Estas três linhas desmontam o pacote que chega do cliente. A primeira delas extrai os dados do pacote e os coloca no `String sentence`; há uma linha análoga em `UDPClient`. A segunda linha extrai o endereço IP. A terceira linha extrai o número de porta do cliente, que é escolhido pelo cliente e é diferente do número de porta 9876 do servidor. (Discutiremos números de porta do cliente com mais detalhes no capítulo seguinte.) É necessário que o servidor obtenha o endereço (endereço IP e número de porta) do cliente para que possa enviar a sentença em letras maiúsculas de volta para ele.

Isso conclui nossa análise sobre o par de programas UDP. Para testar a aplicação, instale e compile `UDPClient.java` em um hospedeiro e `UDPServer.java` em outro. (Não esqueça de incluir o nome apropriado do hospedeiro do servidor em `UDPClient.java`.) Em seguida, execute os dois programas em seus respectivos hospedeiros. Ao contrário do TCP, você pode executar primeiramente o lado cliente e depois o lado servidor. Isso acontece porque, quando você executa o programa cliente, o processo cliente não tenta iniciar uma conexão com o servidor. Assim que tiver executado os programas cliente e servidor, você poderá usar a aplicação digitando uma linha no cliente.

2.9 Construindo um servidor Web simples

Agora que já estudamos o HTTP com algum detalhe e já aprendemos como escrever aplicações cliente–servidor em Java, vamos combinar esses novos conhecimentos e construir um servidor Web simples em Java. Você verá que a tarefa é fácil!

2.9.1 Funções do servidor Web

Nossa meta é construir um servidor que faça o seguinte:

- Manipule apenas uma requisição HTTP.
- Aceite e analise a requisição HTTP.
- Obtenha o arquivo solicitado do sistema de arquivos do servidor.
- Crie uma mensagem de resposta HTTP consistindo no arquivo requisitado precedido de linhas de cabeçalho.
- Envie a resposta diretamente ao cliente.

Vamos tentar fazer a codificação o mais simples possível para esclarecer questões de rede. A codificação que apresentaremos está longe de ser à prova de fogo! Por exemplo, não vamos nos preocupar com procedimentos de exceção (isto é, com tratamento de erro). Vamos admitir também que o cliente requisite um objeto que está — com certeza — no sistema de arquivos do servidor.

WebServer.java

Aqui está a codificação para um servidor Web simples:

```
import java.io.*;
import java.net.*;
import java.util.*;
class WebServer {
    public static void main(String argv[])throws Exception{
        String requestMessageLine;
        String fileName;
        ServerSocket listenSocket = new ServerSocket(6789);
        Socket connectionSocket = listenSocket.accept();
        BufferedReader inFromClient =
            new BufferedReader(new InputStreamReader(
                connectionSocket.getInputStream()));
        DataOutputStream outToClient =
            new DataOutputStream(
                connectionSocket.getOutputStream());
        requestMessageLine = inFromClient.readLine();
        StringTokenizer tokenizedLine =
            new StringTokenizer(requestMessageLine);
        if (tokenizedLine.nextToken().equals("GET")){
            fileName = tokenizedLine.nextToken();
            if (fileName.startsWith("/") == true )
                fileName = fileName.substring(1);
            File file = new File(fileName);
            int numOfBytes = (int) file.length();
            FileInputStream inFile = new FileInputStream (
                fileName);
            byte[ ] fileInBytes = new byte[numOfBytes];
            inFile.read(fileInBytes);
            outToClient.writeBytes(
                "HTTP/1.0 200 Document Follows\r\n");
            if (fileName.endsWith(".jpg"))
                outToClient.writeBytes("Content-Type:
                    image/jpeg\r\n");
            if (fileName.endsWith(".gif"))
                outToClient.writeBytes("Content-Type:
                    image/gif\r\n");
            outToClient.writeBytes("Content-Length:" +
                numOfBytes + "\r\n");
            outToClient.writeBytes("\r\n");
            outToClient.write(fileInBytes, 0, numOfBytes);
            connectionSocket.close();
```

```
            }
        else System.out.println("Bad Request Message");
        }
}
```

Vamos agora examinar a codificação. A primeira metade do programa é quase idêntica ao `TCPServer.java`. Como acontece no `TCPServer.java`, importamos os pacotes `java.io` e `java.net`. Além desses dois pacotes, também importamos o pacote `java.util`, que contém a classe `StringTokenizer`, usada para analisar mensagens de requisição HTTP. Examinando agora as linhas dentro da classe `WebServer`, definimos dois objetos do tipo cadeia:

```
String requestMessageLine;
String fileName;
```

O objeto `requestMessageLine` é uma cadeia que conterá a primeira linha da mensagem de requisição HTTP. O objeto `fileName` é uma cadeia que conterá o nome do arquivo requisitado. O conjunto de comandos seguinte é idêntico ao conjunto de comandos correspondente em `TCPServer.java`.

```
ServerSocket listenSocket = new ServerSocket(6789);
Socket connectionSocket = listenSocket.accept();
BufferedReader inFromClient =
    new BufferedReader(new InputStreamReader
            (connectionSocket.getInputStream()));
    DataOutputStream outToClient =
        new DataOutputStream (connectionSocket.
                getOutputStream());
```

São criados dois objetos do tipo socket. O primeiro desses objetos é `listenSocket`, que é do tipo `ServerSocket`. O objeto `listenSocket` é criado pelo programa servidor antes de ele receber de um cliente uma requisição para uma conexão TCP. Ele fica à escuta na porta 6789 e espera uma requisição de conexão TCP de algum cliente. Quando a requisição chega, o método `accept()` do `listenSocket` cria um novo objeto, `connectionSocket`, do tipo `Socket`. Em seguida, duas cadeias são criadas: a `BufferedReader inFromClient` e a `DataOutputStream outToClient`. A mensagem de requisição HTTP vem da rede, através de `connectionSocket`, e vai para dentro de `inFromClient`; a mensagem de resposta HTTP entra em `outToClient`, através de `connectionSocket`, e vai para dentro da rede. A parte restante da codificação é significativamente diferente de `TCPServer.java`.

```
requestMessageLine = inFromClient.readLine();
```

Este comando lê a primeira linha da mensagem de requisição HTTP, que deve ser do tipo:

```
GET file_name HTTP/1.0
```

Nosso servidor deverá agora analisar a linha para extrair o nome do arquivo.

```
StringTokenizer tokenizedLine =
                        new StringTokenizer(requestMessageLine);
```

```
if (tokenizedLine.nextToken( ).equals("GET")){
    fileName = tokenizedLine.nextToken();
    if (fileName.startsWith("/") == true )
            fileName = fileName.substring(1);
```

Estes comandos analisam a primeira linha da mensagem de requisição para obter o nome do arquivo requisitado. O objeto `tokenizedLine` pode ser considerado como a linha de requisição original, com cada uma das 'palavras' `GET`, `file_name` e `HTTP/1.0` colocadas em lugares reservados e separados, denominados tokens. O servidor sabe, pelo RFC do HTTP, que o nome de arquivo para o arquivo requisitado está contido no token que segue aquele que contém "GET". Esse nome de arquivo é colocado em uma cadeia denominada `fileName`. A finalidade da última declaração `if` nessa codificação é remover a barra inclinada (/) que pode preceder o nome do arquivo.

```
FileInputStream inFile = new FileInputStream (fileName);
```

Este comando anexa uma cadeia, `inFile`, ao arquivo `fileName`.

```
byte[] fileInBytes = new byte[numOfBytes];
inFile.read(fileInBytes);
```

Estes comandos determinam o tamanho do arquivo e constroem um vetor de bytes desse tamanho, que tem o nome de `fileInBytes`. O último comando lê da cadeia `inFile` para o vetor de bytes `fileInBytes`. O programa deve converter para bytes porque a cadeia de saída `outToClient` só pode ser alimentada com bytes.

Agora estamos prontos para construir a mensagem de resposta HTTP. Para esse fim, devemos primeiramente enviar as linhas de cabeçalho da resposta HTTP para dentro de `DataOutputStream outToClient`:

```
outToClient.writeBytes("HTTP/1.0 200 Document
        Follows\r\n");
if (fileName.endsWith(".jpg"))
        outToClient.writeBytes("Content-Type:
        image/jpeg\r\n");
if (fileName.endsWith(".gif"))
        outToClient.writeBytes("Content-Type:
        image/gif\r\n");
outToClient.writeBytes("Content-Length: " + numOfBytes +
            "\r\n");
outToClient.writeBytes("\r\n")
```

Este conjunto de comandos é particularmente interessante. Os comandos preparam as linhas de cabeçalho para a mensagem de resposta HTTP e as enviam para o buffer TCP de envio. O primeiro comando envia a linha de estado obrigatória: `HTTP/1.0 200 Document Follows`, seguida de um 'carriage return' e um 'line feed'. As duas linhas de comando seguintes preparam uma única linha de cabeçalho do tipo Content-Type. Se for para o servidor transferir uma imagem GIF, ele preparará a linha de cabeçalho `Content-Type: image/gif`. Se, por outro lado, for para o servidor transferir uma imagem JPEG, ele preparará a linha de cabeçalho `Content-Type: image/jpeg`. (Nesse servidor Web simples, nenhuma linha de conteúdo será enviada se o objeto não for nem um GIF nem uma imagem JPEG.) O servidor então prepa-

ra e envia uma linha de cabeçalho de comprimento de conteúdo e uma linha em branco obrigatória que antecede o objeto a ser enviado. Em seguida, devemos enviar o arquivo `FileName` para dentro de `DataOutputStream outToClient`.

Agora podemos enviar o arquivo requisitado:

```
outToClient.write(fileInBytes, 0, numOfBytes);
```

Esse comando envia o arquivo requisitado, `fileInBytes`, ao buffer TCP de envio. O TCP concatenará o arquivo, `fileInBytes`, com as linhas de cabeçalho recém-criadas, segmentará a concatenação, se necessário, e enviará os segmentos TCP ao cliente. Após atender a uma requisição para um arquivo, o servidor põe a casa em ordem, fechando o socket `connectionSocket`.

```
connectionSocket.close();
```

Para testar esse servidor Web, instale-o em um hospedeiro. Ponha também alguns arquivos no hospedeiro. Então, use um browser que esteja rodando em qualquer máquina para requisitar um arquivo do servidor. Nesse momento, você precisará usar o número de porta que incluiu na codificação do servidor (por exemplo, 6789). Portanto, se seu servidor está localizado em `somehost.somewhere.edu`, o arquivo é `somefile.html` e o número de porta é 6789 e, então, o browser deverá requisitar o seguinte:

```
http://somehost.somewhere.edu:6789/somefile.html
```

2.10 Resumo

Neste capítulo, estudamos os aspectos conceituais e os aspectos de implementação de aplicações de rede. Conhecemos a onipresente arquitetura cliente–servidor adotada por aplicações da Internet e examinamos sua utilização nos protocolos HTTP, FTP, SMTP, POP3 e DNS. Analisamos esses importantes protocolos de camada de aplicação e suas aplicações associadas (Web, transferência de arquivos, e-mail e DNS) com algum detalhe. Conhecemos também a arquitetura P2P, cada vez mais dominante, e examinamos sua utilização no compartilhamento de arquivos P2P. Vimos como o API socket pode ser usado para construir aplicações de rede. Examinamos a utilização de portas para serviços de transporte fim-a-fim orientados para conexão (TCP) e não orientados para conexão (UDP) e também construímos um servidor Web simples usando sockets. A primeira etapa de nossa jornada de descida pela arquitetura das camadas da rede está concluída!

Logo no começo deste livro, na Seção 1.1, demos uma definição um tanto vaga e despojada de um protocolo. Dissemos que um protocolo é "o formato e a ordem das mensagens trocadas entre duas ou mais entidades comunicantes, bem como as ações realizadas na transmissão e/ou no recebimento de uma mensagem ou outro evento". O material deste capítulo — em particular, o estudo detalhado dos protocolos HTTP, FTP, SMTP, POP3 e DNS — agregou considerável substância a essa definição. Protocolos são o conceito fundamental de redes. Nosso estudo sobre protocolos de aplicação nos deu agora a oportunidade de desenvolver uma noção mais intuitiva do que eles realmente são.

Na Seção 2.1, descrevemos os modelos de serviço que o TCP e o UDP oferecem às aplicações que os chamam. Examinamos esses modelos de serviço ainda mais de perto quando desenvolvemos, nas seções 2.7 a 2.9, aplicações simples que executam em TCP e UDP. Contudo, pouco dissemos sobre como o TCP e o UDP fornecem esses modelos de serviços. Por exemplo, dissemos pouco sobre como o TCP provê um serviço confiável de transferência de dados às suas aplicações. No próximo capítulo, examinaremos cuidadosamente não apenas o que são protocolos de transporte, mas também o como e o porquê deles.

Agora que conhecemos a estrutura da aplicação da Internet e os protocolos de camada de aplicação, estamos prontos para continuar a descer a pilha de protocolos e examinar a camada de transporte no Capítulo 3.

Exercícios de fixação

Capítulo 2 Questões de revisão

Seção 2.1

1. Relacione cinco aplicações da Internet não proprietárias e os protocolos de camada de aplicação que elas usam.
2. Qual é a diferença entre arquitetura de rede e arquitetura de aplicação?
3. De que modo mensagem instantânea é um híbrido das arquiteturas cliente-servidor e P2P?
4. Para uma sessão de comunicação entre um par de processos, qual processo é o cliente e qual é o servidor?
5. Em uma aplicação de compartilhamento de arquivos P2P, você concorda com a afirmação: "não existe nenhuma noção de lados cliente e servidor de uma sessão de comunicação"? Por que sim ou por que não?
6. Que informação é usada por um processo que está rodando em um hospedeiro para identificar um processo que está rodando em outro hospedeiro?
7. Relacione os vários agentes de usuário de aplicação de rede que você utiliza no dia-a-dia.
8. Com referência à Figura 2.4, vemos que nenhuma das aplicações relacionadas nela requer 'sem perda de dados' e 'temporização'. Você consegue imaginar uma aplicação que requeira 'sem perda de dados' e seja também altamente sensível ao atraso?

Seções 2.2 a 2.6

9. O que significa protocolo de apresentação (handshaking protocol)?
10. Por que HTTP, FTP, SMTP, POP3 e IMAP rodam sobre TCP e não sobre UDP?
11. Considere um site de comércio eletrônico que quer manter um registro de compras para cada um de seus clientes. Descreva como isso pode ser feito com cookies.
12. Qual é a diferença entre HTTP persistente com paralelismo e HTTP persistente sem paralelismo? Qual dos dois é usado pelo HTTP/1.1?
13. Descreva como o cache Web pode reduzir o atraso na recepção de um objeto desejado. O cache Web reduzirá o atraso para todos os objetos requisitados por um usuário ou somente para alguns objetos? Por quê?
14. Digite um comando Telnet em um servidor Web e envie uma mensagem de requisição com várias linhas. Inclua nessa mensagem a linha de cabeçalho `If-modified-since:` para forçar uma mensagem de resposta com a codificação de estado `304 Not Modified`.
15. Por que se diz que o FTP envia informações de controle 'fora da banda'?
16. Suponha que Alice envie uma mensagem a Bob por meio de uma conta de e-mail da Web (como o Hotmail), e que Bob acesse seu e-mail por seu servidor de correio usando POP3. Descreva como a mensagem vai do hospedeiro de Alice até o hospedeiro de Bob. Não se esqueça de relacionar a série de protocolos de camada de aplicação usados para movimentar a mensagem entre os dois hospedeiros.
17. Imprima o cabeçalho de uma mensagem de e-mail que acabou de receber. Quantas linhas de cabeçalho `Received:` há nela? Analise cada uma das linhas.
18. Da perspectiva de um usuário, qual é a diferença entre o modo ler-e-apagar e o modo ler-e-guardar no POP3?
19. É possível que o servidor Web e o servidor de correio de uma organização tenham exatamente o mesmo apelido para um nome de hospedeiro (por exemplo, `foo.com`)? Qual seria o tipo de RR que contém o nome de hospedeiro do servidor de correio?

20. O que é uma rede de sobreposição em um sistema de compartilhamento de arquivos P2P? Ela inclui roteadores? O que são as arestas da rede de sobreposição? Como a rede de sobreposição Gnutella é criada e como é mantida?

21. Descubra três empresas que oferecem serviços de compartilhamento de arquivos P2P. Que tipo de conteúdo é distribuído por essas empresas? Como cada um dos projetos habilita usuários a localizar o conteúdo?

Seções 2.7 a 2.9

22. O servidor UDP descrito na Seção 2.8 precisava de uma porta apenas, ao passo que o servidor TCP descrito na Seção 2.7 precisava de duas portas. Por quê? Se o servidor TCP tivesse de suportar n conexões simultâneas, cada uma de um hospedeiro cliente diferente, de quantas portas precisaria?

23. Para a aplicação cliente-servidor por TCP descrita na Seção 2.7, por que o programa servidor deve ser executado antes do programa cliente? Para a aplicação cliente–servidor por UDP descrita na Seção 2.8, por que o programa cliente pode ser executado antes do programa servidor?

Problemas

1. Falso ou verdadeiro?

a. Suponha que um usuário requisite uma página Web que consiste em texto e duas imagens. Para essa página, o cliente enviará uma mensagem de requisição e receberá três mensagens de resposta.

b. Duas páginas Web distintas (por exemplo, `www.mit.edu/research.html` e `www.mit.edu/students.html`) podem ser enviadas pela mesma conexão persistente.

c. Com conexões não persistentes entre browser e servidor de origem, é possível que um único segmento TCP transporte duas mensagens distintas de requisição HTTP.

d. O cabeçalho `Date:` na mensagem de resposta HTTP indica a última vez que o objeto da resposta foi modificado.

2. Leia o RFC 959 para FTP. Relacione todos os comandos de cliente que são suportados pelo RFC.

3. Visite `http://www.iana.org`. Quais são os números de porta bem conhecidos para o protocolo simples de transferência de arquivos (STFP)? E para o protocolo de transferência de notícias pela rede (NNTP)?

4. Considere um cliente HTTP que queira obter um documento Web em um dado URL. Inicialmente, o endereço IP do servidor HTTP é desconhecido. O documento Web no URL tem uma imagem GIF inserida que reside no mesmo servidor do documento original. Nesse cenário, quais protocolos de transporte e de camada de aplicação são necessários, além do HTTP?

5. Obtenha a especificação HTTP/1.1 (RFC 2616). Responda às seguintes perguntas:

a. Explique o mecanismo de sinalização que cliente e servidor utilizam para indicar que uma conexão persistente está sendo fechada. O cliente, o servidor, ou ambos, podem sinalizar o encerramento de uma conexão?

b. Que serviços de criptografia são providos pelo HTTP?

6. Suponha que você clique com seu browser Web sobre um ponteiro para obter uma página Web e que o endereço IP para o URL associado não esteja no cache de seu hospedeiro local. Portanto, será necessária uma consulta ao DNS para obter o endereço IP. Considere que `n` servidores DNS sejam visitados antes que seu hospedeiro receba o endereço IP do DNS; as visitas sucessivas incorrem em um RTT de $RTT_1, \ldots, RTT_n$. Suponha ainda que a página Web associada ao ponteiro contenha exatamente um objeto que consiste em uma pequena quantidade de texto HTML. Seja RTT_0 o RTT entre o hospedeiro local e o servidor que contém o objeto. Admitindo que o tempo de transmissão do objeto seja zero, quanto tempo passará desde que o cliente clica o ponteiro até que receba o objeto?

7. Com referência ao problema 6, suponha que o arquivo HTML referencie três objetos muito pequenos no mesmo servidor. Desprezando tempos de transmissão, quanto tempo passa, usando-se:

a. HTTP não persistente sem conexões TCP paralelas?

b. HTTP não persistente com conexões paralelas?

c. HTTP persistente com paralelismo?

8. GET e POST são dois métodos de requisição HTTP. Há quaisquer outros métodos em HTTP/1.0? Se houver, para que são usados? Há outros métodos em HTTP/1.1?

9. Considere a Figura 2.11, que mostra uma rede institucional conectada à Internet. Suponha que o tamanho médio do objeto seja 900 mil bits e que a taxa média de requisição dos browsers da instituição aos servidores de origem seja 1,5 requisição por segundo. Suponha também que a quantidade de tempo que leva desde o instante em que o roteador do lado da Internet do enlace de acesso transmite uma requisição HTTP até que receba a resposta seja 2 segundos em média (veja Seção 2.2.6). Modele o tempo total médio de resposta como a soma do atraso de acesso médio (isto é, o atraso entre o roteador da Internet e o roteador da instituição) e o tempo médio de atraso da Internet. Para a média de atraso de acesso, use $\Delta/(1 - \Delta\beta)$, onde Δ é o tempo médio requerido para enviar um objeto pelo enlace de acesso e β é a taxa de chegada de objetos ao enlace de acesso.

a. Determine o tempo total médio de resposta.

b. Agora, considere que um cache é instalado na LAN institucional e que a taxa de resposta local seja 0,4. Determine o tempo total de resposta.

10. Escreva um programa TCP simples para um servidor que aceite linhas de entrada de um cliente e envie as linhas para a saída padrão do servidor. (Você pode fazer isso modificando o programa TCPServer.java no texto.) Compile e execute seu programa. Em qualquer outra máquina que contenha um browser Web, defina o servidor proxy no browser para a máquina que está executando seu programa servidor e também configure o número de porta adequadamente. Seu browser deverá agora enviar suas mensagens de requisição GET a seu servidor, e este deverá apresentar as mensagens em sua saída padrão. Use essa plataforma para determinar se seu browser gera mensagens GET condicionais para objetos que estão em caches locais.

11. Leia o RFC do POP3 [RFC 1939]. Qual é a finalidade do comando UIDL do POP3?

12. Considere acessar seu e-mail com POP3.

a. Suponha que você configure seu cliente de correio POP para funcionar no modo ler-e-apagar. Conclua a seguinte transação:

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: blah blah ...
S: .........blah
S: .
?
?
```

b. Suponha que você configure seu cliente de correio POP para funcionar no modo ler-e-guardar. Conclua a seguinte transação:

```
C: list
S: 1 498
```

```
S: 2 912
S: .
C: retr 1
S: blah blah ...
S: .........blah
S: .
?
?
```

c. Suponha que você configure seu cliente de correio POP para funcionar no modo ler-e-guardar. Usando sua solução na parte (b), suponha que você recupere as mensagens 1 e 2, saia do POP e então, 5 minutos mais tarde, acesse novamente o POP para obter um novo e-mail. Imagine que nenhuma outra mensagem foi enviada nesse intervalo. Elabore um transcript dessa segunda sessão POP.

13. a. O que é um banco de dados *whois*?

b. Use vários bancos de dados whois da Internet para obter os nomes de dois servidores DNS. Cite quais bancos de dados whois você utilizou.

c. Use nslookup em seu hospedeiro local para enviar consultas DNS a três servidores de nomes: seu servidor DNS local e os dois servidores DNS que encontrou na parte (b). Tente consultar registros dos tipos A, NS e MX. Faça um resumo do que encontrou.

d. Use nslookup para encontrar um servidor Web que tenha vários endereços IP. O servidor Web de sua instituição (escola ou empresa) tem vários endereços IP?

e. Use o banco de dados whois ARIN para determinar a faixa de endereços IP usados por sua universidade.

f. Descreva como um invasor pode usar bancos de dados whois e a ferramenta nslookup para fazer o reconhecimento de uma instituição antes de lançar um ataque.

g. Discuta por que bancos de dados whois devem ser disponíveis publicamente.

14. Suponha que você esteja baixando MP3 utilizando algum sistema de compartilhamento de arquivos P2P. O gargalo da Internet é o seu enlace de acesso residencial *full-duplex* de 128 kbps. Enquanto você está baixando MP3, repentinamente, dez outros usuários começam a carregar MP3 do seu computador. Admitindo que sua máquina seja de grande capacidade e que todas essas transferências e carregamentos não estão submetendo sua máquina (CPU, E/S de disco e assim por diante) a nenhum esforço extraordinário, os carregamentos simultâneos — que também estão passando pelo gargalo do seu enlace — retardam suas transferências? Por que sim ou por que não? Responda à mesma pergunta quando você dispuser de 128 kbps na direção do provedor e 512 na direção do usuário, como parte de uma conexão ADSL.

15. Suponha que haja N pares ativos na rede Gnutella e que cada dupla de pares tenha uma conexão TCP ativa. Suponha ainda que as conexões TCP passem por um total de *M* roteadores. Quantos nós e arestas há na rede de sobreposição correspondente?

16. Na Seção 2.6, descrevemos com certo grau de detalhe como um novo par se junta à rede Gnutella. Neste problema queremos explorar o que acontece quando um par sai da rede Gnutella. Suponha que cada nó participante mantenha conexões TCP com um número mínimo de quatro pares distintos o tempo todo. Imagine que o Par *X*, que tem cinco conexões TCP com outros pares, queira sair.

a. Em primeiro lugar, considere o caso de uma saída graciosa, isto é, o Par *X* fecha sua aplicação explicitamente e, por conseguinte, fecha graciosamente suas cinco conexões TCP. Que ações executariam cada um dos pares anteriormente conectados?

b. Agora, suponha que *X* desconecte-se abruptamente da Internet sem notificar seus cinco vizinhos que está fechando as conexões TCP. O que aconteceria?

17. Neste problema exploramos o roteamento pelo caminho inverso das mensagens QueryHit em Gnutella. Suponha que Alice emita uma mensagem Query. Admita ainda mais que Bob receba as mensagens Query (que podem ter sido transmitidas por vários pares intermediários) e que tem um arquivo que corresponde à mensagem.

a. Lembre-se de que, quando um par tem um arquivo correspondente, ele envia uma mensagem QueryHit pelo caminho inverso da mensagem Query correspondente. Um projeto alternativo seria Bob estabelecer uma conexão TCP direta com Alice e enviar a mensagem QueryHit por essa conexão. Quais são as vantagens e desvantagens desse projeto alternativo?

b. No protocolo Gnutella, quando o par Alice gera uma mensagem Query, insere um ID exclusivo no campo MessageID da mensagem. Sempre que o par Bob tem uma correspondência, gera uma mensagem QueryHit usando o mesmo MessageID da mensagem Query. Descreva como pares podem utilizar o campo MessageID e tabelas locais de roteamento para realizar o roteamento pelo caminho inverso.

c. Uma abordagem alternativa, que não utiliza identificadores de mensagens, é a seguinte: quando uma mensagem Query chega a um par, antes de retransmiti-la o par amplia a mensagem adicionando a ela o seu endereço IP. Descreva como pares podem usar esse mecanismo para executar roteamento pelo caminho inverso.

18. Repita o problema 17, mas agora responda às perguntas considerando as mensagens Ping e Pong da Gnutella (e não as mensagens Query e QueryHit).

19. Neste problema exploramos a elaboração de um sistema semelhante ao KaZaA que tem os costumeiros nós, líderes de grupo e líderes de supergrupo.

a. Suponha que cada líder de supergrupo seja responsável por aproximadamente 200 líderes de grupo e que cada líder de grupo seja responsável por aproximadamente 200 pares normais. Quantos líderes de supergrupo seriam necessários para uma rede de quatro milhões de pares?

b. Que informações cada líder de grupo armazenaria? Que informações cada líder de supergrupo armazenaria? Como uma busca poderia ser realizada nesse projeto de três níveis?

20. Considere uma inundação de consultas em compartilhamento de arquivos P2P, como discutido na Seção 2.6. Suponha que cada par esteja conectado a, no máximo, N vizinhos na rede de sobreposição. Imagine também que o campo de contagem de nós esteja inicialmente estabelecido em K. Suponha que Alice faça uma consulta. Determine um limite superior para o número de mensagens de consulta que é enviado para a rede de sobreposição.

21. Instale e compile os programas Java TCPClient e UDPClient em um hospedeiro e TCPServer e UDPServer em outro.

a. Suponha que você execute TCPClient antes de executar TCPServer. O que acontece? Por quê?

b. Imagine que você execute UDPClient antes de UDPServer. O que acontece? Por quê?

c. O que acontece se você usar números de porta diferentes para os lados cliente e servidor?

22. Reescreva TCPServer.java de modo que ele possa aceitar várias conexões. (*Dica*: você precisará usar threads.)

23. Suponha que, em UDPClient.java, a linha

```
DatagramSocket clientSocket = new DatagramSocket( );
```

seja substituída por

```
DatagramSocket clientSocket = new DatagramSocket(5432);
```

Será necessário mudar UDPServer.java? Quais são os números de porta para os sockets em UDPClient e UDPServer? Quais eram esses números antes dessa mudança?

Questões dissertativas

1. Na sua opinião, por que as aplicações de compartilhamento de arquivos P2P são tão populares? Será por que distribuem música e vídeo gratuitamente (o que é legalmente discutível) ou por que seu número imenso de servidores atende eficientemente uma demanda maciça por megabytes? Ou será pelas duas razões?
2. Há esforços em curso com a finalidade de construir um sistema de compartilhamento de arquivos P2P de código-fonte aberto que explore a heterogeneidade dos pares (como faz a tecnologia proprietária KaZaA)?
3. Sites de comércio eletrônico e outros sites Web freqüentemente têm bancos de dados "de apoio". Como servidores HTTP se comunicam com esses bancos de dados?
4. O que é HTML dinâmica? Dê exemplos de sites Web que usam HTML dinâmica.
5. Cite algumas linguagens de scripting populares do lado do servidor. Qual é a finalidade desse scripting?
6. Como você pode configurar seu browser para cache local? Que opções de cache você tem?
7. Você pode configurar seu browser para abrir várias conexões simultâneas com um site Web? Quais são as vantagens e as desvantagens de ter um grande número de conexões TCP simultâneas?
8. Na Seção 2.4.4, dissemos que um serviço de e-mail pela Web é freqüentemente implementado usando um servidor Web e um servidor IMAP. Como o servidor HTTP se comunica com o servidor IMAP?
9. Vimos que sockets TCP da Internet tratam os dados que estão sendo enviados como uma cadeia de bytes, mas que sockets UDP reconhecem fronteiras de mensagens. Cite uma vantagem e uma desvantagem da API orientada para bytes em relação à API que reconhece e preserva explicitamente as fronteiras das mensagens definidas por aplicações.
10. Suponha que você tenha um banco de dados que contém dados de astrologia e que queira disponibilizar os dados para browsers Web que funcionam em PCs, PDAs e telefones comuns (usando conversão de texto para voz). Como você faria isso?
11. Discuta a relação entre mensagem instantânea e o protocolo SIP.
12. Como funcionam listas de amigos em servidores de mensagem instantânea?
13. O que é o servidor Web Apache? Quanto custa? Que funcionalidade tem atualmente?
14. Imagine que as organizações responsáveis pela padronização da Web decidam modificar a convenção de nomeação de modo que cada objeto seja nomeado e referenciado por um nome exclusivo que independa de localização (um URN). Discuta algumas questões que envolveriam tal modificação.
15. Considere os esquemas usados correntemente pelo KaZaA para incentivar usuários a se tornarem carregadores ativos. De que outros modos um sistema de compartilhamento de arquivos P2P oferece incentivos a seus usuários?

Tarefas de programação de sockets

Detalhes completos destas tarefas, bem como trechos importantes em código Java, podem ser encontrados no site `www.aw.com/kurose_br`.

Tarefa 1: Servidor Web Multithread

Ao final desta tarefa de programação, você terá desenvolvido, em Java, um servidor Web multithread, que seja capaz de atender várias requisições em paralelo. Você implementará a versão 1.0 do HTTP como definida no RFC 1945.

Lembre-se de que o HTTP/1.0 cria uma conexão TCP separada para cada par requisição/resposta. Cada uma dessas conexões será manipulada por um thread. Haverá também um thread principal, no qual o servidor ficará à escuta de clientes que quiserem estabelecer conexões. Para simplificar o trabalho de programação, desenvolveremos a codificação em dois estágios. No primeiro estágio, você escreverá um servidor multithread que simplesmente apresenta o conteúdo da mensagem de requisição HTTP que recebe. Depois que esse programa estiver executando normalmente, você adicionará a codificação necessária para gerar uma resposta apropriada.

Ao desenvolver a codificação, você poderá testar seu servidor com um browser Web. Mas lembre-se de que você não estará atendendo através da porta padrão 80, portanto, precisará especificar o número de porta dentro do URL que der a seu browser. Por exemplo, se o nome de seu hospedeiro for `host.someschool.edu`, seu servidor estiver à escuta na porta 6789 e você quiser obter o arquivo `index.html`, então deverá especificar o seguinte URL dentro do browser:

```
http://host.someschool.edu:6789/index.html
```

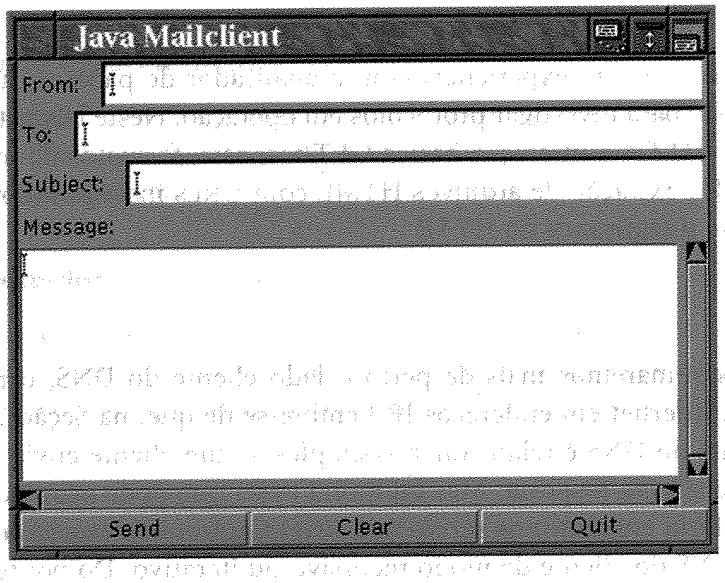
Quando seu servidor encontrar um erro, deverá enviar uma mensagem de resposta com uma fonte HTML adequada, de modo que a informação de erro seja apresentada na janela do browser.

Tarefa 2: Cliente de correio

Nesta tarefa, você desenvolverá um agente de usuário de correio em Java com as seguintes características:

- Que provê uma interface gráfica para o remetente com campos para o servidor de correio local, para o endereço de e-mail do remetente, para o endereço de e-mail do destinatário, para o assunto da mensagem e para a própria mensagem.
- Que estabelece uma conexão TCP entre o cliente de correio e o servidor de correio local. Que envia comandos SMTP para o servidor de correio local. Que recebe e processa comandos SMTP do servidor de correio local.

Esta será a aparência de sua interface:



Você desenvolverá o agente de usuário de modo que ele envie uma mensagem de e-mail para no máximo um destinatário por vez. Além disso, o agente de usuário admitirá que a parte de domínio do endereço de e-mail do destinatário será o nome canônico do servidor SMTP do destinatário. (O agente de usuá-

rio não realizará uma busca no DNS para um registro MX; portanto, o remetente deverá fornecer o nome real do servidor de correio.)

Tarefa 3: UDP Pinger Lab

Neste laboratório, você implementará um cliente e um servidor Ping simples, em UDP. A funcionalidade que esses programas oferecem é similar à do programa Ping padronizado, disponível em sistemas operacionais modernos. O Ping padronizado funciona enviando à Internet uma mensagem ECHO do Protocolo de Mensagens de Controle da Internet (ICMP), que a máquina remota devolve ao remetente como se fosse um eco. Então, o remetente pode determinar o tempo de viagem de ida e volta entre ele mesmo e o computador para o qual enviou o Ping.

O Java não provê nenhuma funcionalidade para enviar ou receber mensagens ICMP e é por isso que neste laboratório você implementará o Ping na camada de aplicação com sockets e mensagens UDP padronizados.

Tarefa 4: Servidor Proxy Web

Neste laboratório, você desenvolverá um servidor proxy Web simples que também poderá fazer cache de páginas Web. Esse servidor aceitará uma mensagem GET de um browser, transmitirá essa mensagem ao servidor Web destinatário, receberá a mensagem de resposta HTTP do servidor destinatário e transmitirá a mensagem de resposta ao browser. Esse servidor proxy é muito simples: entende apenas requisições GET simples. Contudo, pode manipular todos os tipos de objetos; não somente páginas HTML, mas também imagens.

Ethereal Labs

Você encontrará detalhes completos, em inglês, sobre os Ethereal Labs no site `www.aw.com/kurose_br`.

Ethereal Lab: HTTP

Como já tivemos uma primeira experiência com o analisador de pacotes Ethereal no Lab 1, estamos prontos para usar o Ethereal para investigar protocolos em operação. Neste laboratório, exploraremos diversos aspectos do protocolo HTTP: a interação básica GET/resposta, formatos de mensagens HTTP, extração de grandes arquivos HTML, extração de arquivos HTML com URLs inseridos, conexões persistentes e não persistentes e autenticação e segurança do HTTP.

Ethereal Lab: DNS

Neste laboratório, examinaremos mais de perto o lado cliente do DNS, o protocolo que transforma nomes de hospedeiros da Internet em endereços IP. Lembre-se de que, na Seção 2.5, dissemos que o papel desempenhado pelo cliente no DNS é relativamente simples — um cliente envia uma consulta a seu servidor de nomes local e recebe uma resposta. Mas há muita coisa que pode acontecer nos bastidores e que é invisível para os clientes DNS enquanto os servidores DNS hierárquicos se comunicam uns com os outros para resolver a consulta DNS do cliente de modo recursivo ou iterativo. Do ponto de vista do cliente DNS, todavia, o protocolo é bastante simples — é formulada uma consulta ao servidor de nomes local e é recebida uma resposta deste. Nesse laboratório observamos o DNS em ação.

Entrevista

Tim Berners-Lee

Tim Berners-Lee é diretor do World Wide Web Consortium (W3C) e chefe de pesquisa do Laboratório de Ciência da Computação do MIT. Em 1989, quando trabalhava no European Particle Physics Laboratory (Laboratório Europeu de Física das Partículas) do CERN, Tim inventou uma aplicação de hipermídia baseada na Internet para compartilhamento global de informações que se tornou conhecida como a World Wide Web. Um ano mais tarde, ele escreveu o primeiro programa cliente e o primeiro servidor para a Web. Formou-se em Física pela Universidade de Oxford, Inglaterra, em 1976.

Você estudou física. Quais são as semelhanças entre redes e física?

Quando estuda física, você imagina quais regras de comportamento em escala muito pequena poderiam gerar o mundo em grande escala como o percebemos. Quando se projeta um sistema global como a Web, tenta-se inventar regras de comportamento de páginas Web, enlaces e outras coisas que poderiam, no sentido lato, criar um mundo em grande escala da maneira que gostaríamos que ele fosse. Uma é análise e a outra é síntese, mas são muito semelhantes.

O que o fez se decidir pela especialização em redes?

Após ter me formado em física, as empresas de pesquisa em telecomunicações me pareciam ser os lugares mais interessantes. O microprocessador tinha acabado de sair e as telecomunicações estavam passando muito rapidamente da lógica fixa para sistemas baseados em microprocessadores. Era muito interessante.

Qual é a parte de seu trabalho que mais o desafia?

Quando dois grupos discordam seriamente sobre alguma coisa, mas querem, no final, alcançar um objetivo comum, descobrir exatamente o que cada um quer dizer e onde estão os mal-entendidos pode exigir um esforço muito grande. Qualquer chefe de grupo de trabalho sabe disso. Contudo, isso é necessário para se chegar a um consenso amplo.

Em sua opinião, qual é o futuro das redes/Internet?

Como disse em meu livro, eu tenho um sonho para a Web... e esse sonho tem duas partes.

Na primeira, espero que a Web se torne um meio muito mais poderoso de colaboração entre pessoas. Sempre imaginei o espaço da informação como algo ao qual todos tenham acesso imediato e intuitivo, e não apenas para consultar, mas também para criar... Além disso, o sonho da comunicação direta entre pessoas por meio de conhecimento compartilhado tem de ser possível para grupos de todos os tamanhos, interagindo eletronicamente com a mesma facilidade com que interagem pessoalmente.

Na segunda parte do sonho, as colaborações estendem-se a computadores, com as máquinas capacitadas a analisar todos os dados da Web — o conteúdo, os enlaces e as transações entre pessoas e computadores. Ainda está para surgir uma "Web Semântica" que torne isso possível, mas, quando surgir, os mecanismos cotidianos do comércio, da burocracia e de nossas vidas diárias serão manipulados por máquinas que falam com máquinas, deixando a cargo dos seres humanos a inspiração e a intuição... Esta Web inteligível para máquinas se tornará realidade por meio da implementação de uma série de avanços técnicos e contratos sociais que já estão começando.

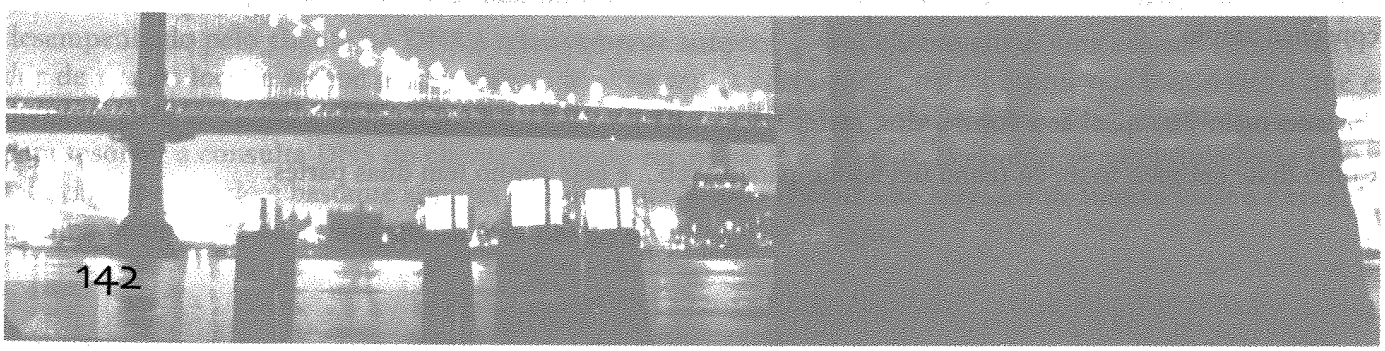
Quando as duas partes desse sonho se realizarem, a Web será um lugar onde a fantasia de um ser humano e o raciocínio de uma máquina coexistirão formando uma mistura ideal e poderosa.

Que pessoas o inspiraram profissionalmente?

Meus pais, que estiveram envolvidos nos primórdios da computação, me legaram uma fascinação por tudo o que se refere a esse assunto. Mike Sendall e Peggie Rimmer, para quem trabalhei em várias ocasiões no CERN, são duas das pessoas, entre outras, que me ensinaram e me estimularam. Mais tarde, aprendi a admirar pessoas como Vanevar Bush, Doug Englebart e Ted Nelson, que sonharam com coisas semelhantes em sua época, mas não tiveram o benefício da existência de PCs e da Internet para poder realizá-las.

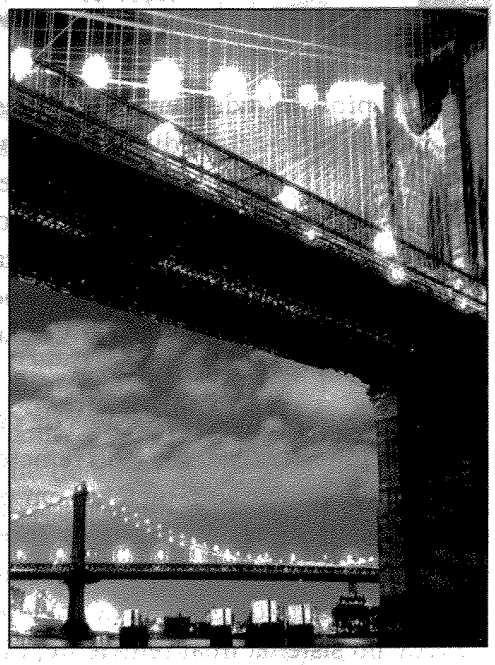
A missão do W3C é "conduzir a Web ao seu total potencial". Em sua opinião, qual é o potencial da Web? De que maneira esse potencial pode se tornar realidade?

Estamos tentando conduzi-la nas duas direções que mencionei: colaboração e Web semântica. Estamos sempre tentando aprimorar a universalidade da Web para que haja apenas uma Web, que, seja qual for seu browser, ela esteja sempre disponível e que qualquer pessoa possa acessá-la, não importando a marca da máquina, o fornecedor de software, a localização geográfica, a deficiência, o idioma ou a cultura.



Capítulo 3

Camada de Transporte



Posicionada entre as camadas de aplicação e de rede, a camada de transporte é uma peça central da arquitetura de rede em camadas. Ela desempenha o papel fundamental de fornecer serviços de comunicação diretamente aos processos de aplicação que rodam em hospedeiros diferentes. A abordagem pedagógica que adotamos neste capítulo é alternar entre discussões de princípios de camada de transporte e o modo como esses princípios são implementados em protocolos existentes; como de costume, daremos particular ênfase aos protocolos da Internet, em particular aos protocolos de camada de transporte TCP e UDP.

Começaremos discutindo a relação entre as camadas de transporte e de rede, preparando o cenário para o exame da primeira função importante da camada de transporte — ampliar o serviço de entrega da camada de rede entre dois sistemas finais para um serviço de entrega entre dois processos de camada de aplicação que rodam nos sistemas finais. Ilustraremos essa função quando abordarmos o UDP, o protocolo de transporte não orientado para conexão da Internet.

Então retornaremos aos princípios e enfrentaremos um dos problemas mais fundamentais de redes de computadores — como duas entidades podem se comunicar de maneira confiável por um meio que pode perder e corromper dados. Mediante uma série de cenários cada vez mais complicados (e realistas!), construiremos um conjunto de técnicas que os protocolos de transporte utilizam para resolver esse problema. Então, mostraremos como esses princípios estão incorporados no TCP, o protocolo de transporte orientado para conexão da Internet.

Em seguida, passaremos para um segundo problema fundamentalmente importante em redes — o controle da taxa de transmissão de entidades de camada de transporte para evitar ou se recuperar de congestionamentos dentro da rede. Consideraremos as causas e conseqüências do congestionamento, bem como técnicas de controle de congestionamento comumente usadas. Após adquirir um sólido entendimento das questões que estão por trás do controle de congestionamento, estudaremos como o TCP o aborda.

3.1 Introdução e serviços de camada de transporte

Nos dois capítulos anteriores, citamos o papel da camada de transporte e os serviços que ela fornece. Vamos revisar rapidamente o que já aprendemos sobre a camada de transporte.

Um protocolo de camada de transporte fornece **comunicação lógica** entre processos de aplicação que rodam em hospedeiros diferentes. *Comunicação lógica* nesse contexto significa que, do ponto de vista de uma aplicação, tudo se passa como se os hospedeiros que rodam os processos estivessem conectados diretamente; na verdade, esses hospedeiros poderão estar em lados opostos do planeta, conectados por numerosos roteadores e uma ampla variedade de tipos de enlace. Processos de aplicação usam a comunicação lógica provida pela camada de transporte para enviar mensagens entre si, livres da preocupação dos detalhes da infra-estrutura física utilizada para transportar essas mensagens. A Figura 3.1 ilustra a noção de comunicação lógica.

Nela, vemos que protocolos de camada de transporte são implementados nos sistemas finais, mas não em roteadores de rede. No lado remetente, a camada de transporte converte as mensagens que recebe de um processo de aplicação remetente em pacotes de camada de transporte, denominados segmentos de camada de transporte na terminologia da Internet. Isso é (possivelmente) feito fragmentando-se as mensagens da aplicação em pedaços menores e adicionando-se um cabeçalho de camada de transporte a cada pedaço para criar o segmento de camada de transporte. A camada de transporte, então, passa o segmento para a camada de rede no sistema final remetente, onde ele é encapsulado em um pacote de camada de rede (um datagrama) e enviado ao destinatário. É importante notar que roteadores de rede agem somente nos campos de camada de rede do datagrama; isto é, não examinam os campos do segmento de camada de transporte encapsulado com o datagrama. No lado destinatário, a camada de rede extrai do datagrama o segmento de camada de transporte e passa-o para a camada de transporte. Em seguida, essa camada processa o segmento recebido, disponibilizando os dados para a aplicação destinatária.

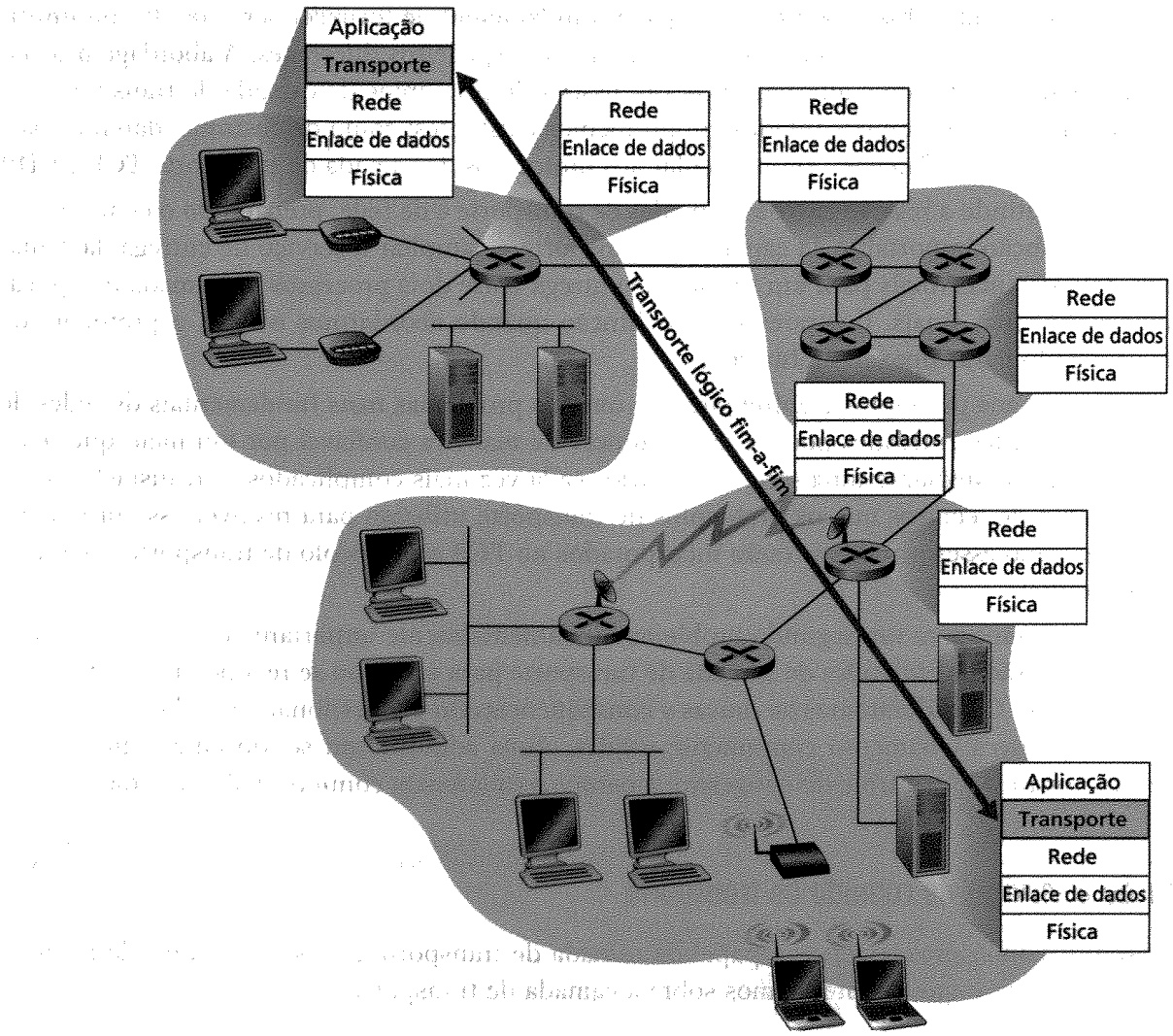


Figura 3.1 A camada de transporte fornece comunicação lógica, e não física, entre processos de aplicações

Pode haver mais de um protocolo de camada de transporte disponível para aplicações de rede. Por exemplo, a Internet tem dois protocolos — TCP e UDP. Cada um deles provê um conjunto diferente de serviços de camada de transporte à aplicação que o está chamando.

3.1.1 Relação entre as camadas de transporte e de rede

Lembre-se de que a camada de transporte se situa logo acima da camada de rede na pilha de protocolos. Enquanto um protocolo de camada de transporte fornece comunicação lógica entre *processos* que rodam em hospedeiros diferentes, um protocolo de camada de rede fornece comunicação lógica entre *hospedeiros*. Essa distinção é sutil, mas importante. Vamos examiná-la com o auxílio de uma analogia com moradias.

Considere duas casas, uma na Costa Leste e outra na Costa Oeste dos Estados Unidos e que, em cada uma delas, há uma dúzia de crianças. As crianças da Costa Leste são primas das crianças da Costa Oeste e todas adoram escrever cartas umas para as outras — cada criança escreve a cada primo uma vez por semana e cada carta é entregue pelo serviço de correio tradicional dentro de um envelope separado. Assim, cada moradia envia 144 cartas por semana para a outra. (Essas crianças economizariam muito dinheiro se tivessem e-mail!) Em cada moradia há uma criança responsável pela coleta e distribuição da correspondência — Ann, na casa da Costa Oeste, e Bill, na da Costa Leste. Toda semana, Ann coleta a correspondência de seus irmãos e irmãs e a coloca no correio. Quando as cartas chegam à casa da Costa Oeste, também é Ann quem tem a tarefa de distribuir a correspondência a seus irmãos e irmãs. Bill realiza o mesmo trabalho na casa da Costa Leste.

Nesse exemplo, o serviço postal provê uma comunicação lógica entre as duas casas — ele movimenta a correspondência de uma casa para outra, e não de uma pessoa para outra. Por outro lado, Ann e Bill provêem comunicação lógica entre os primos — eles coletam e entregam a correspondência de seus irmãos e irmãs. Note que, da perspectiva dos primos, Ann e Bill *são* o serviço postal, embora sejam apenas uma parte do sistema (a parte do sistema final) do processo de entrega fim-a-fim. Esse exemplo das moradias é uma analogia interessante para explicar como a camada de transporte se relaciona com a camada de rede:

mensagens de aplicação = cartas em envelopes
processos = primos
hospedeiros (também denominados sistemas finais) = casas
protocolo de camada de transporte = Ann e Bill
protocolo de camada de rede = serviço postal (incluindo os carteiros)

Continuando com essa analogia, observe que Ann e Bill fazem todo o seu trabalho dentro de suas respectivas casas; eles não estão envolvidos, por exemplo, com a classificação da correspondência em nenhuma central intermediária dos correios ou com o transporte da correspondência de uma central a outra. De maneira semelhante, protocolos de camada de transporte moram nos sistemas finais, onde movimentam mensagens de processos de aplicação para a borda da rede (isto é, para a camada de rede) e vice-versa, mas não interferem no modo como as mensagens são movimentadas dentro do núcleo da rede. Na verdade, como ilustrado na Figura 3.1, roteadores intermediários não reconhecem nenhuma informação que a camada de transporte possa ter anexado às mensagens da aplicação nem agem sobre ela.

Prosseguindo com nossa saga familiar, suponha agora que, quando Ann e Bill saem de férias, outro par de primos — digamos, Susan e Harvey — substitua-os e encarregue-se da coleta interna da correspondência e de sua entrega. Infelizmente para as duas famílias, eles não desempenham essa tarefa do mesmo modo que Ann e Bill. Por serem crianças mais novas, Susan e Harvey recolhem e entregam a correspondência com menos freqüência e, ocasionalmente, perdem cartas (que às vezes acabam mastigadas pelo cão da família). Assim, o par de primos Susan e Harvey não provê o mesmo conjunto de serviços (isto é, o mesmo modelo

de serviço) oferecido por Ann e Bill. De uma maneira análoga, uma rede de computadores pode disponibilizar vários protocolos de transporte, em que cada um oferece um modelo de serviço diferente às aplicações.

Os possíveis serviços que Ann e Bill podem fornecer são claramente limitados pelos possíveis serviços que os correios fornecem. Por exemplo, se o serviço postal não estipula um prazo máximo para entregar a correspondência entre as duas casas (digamos, três dias), então não há nenhuma possibilidade de Ann e Bill definirem um atraso máximo para a entrega da correspondência entre qualquer par de primos. De maneira semelhante, os serviços que um protocolo de transporte pode fornecer são freqüentemente limitados pelo modelo de serviço do protocolo subjacente da camada de rede. Se o protocolo de camada de rede não puder dar garantias contra atraso ou garantias de largura de banda para segmentos de camada de transporte enviados entre hospedeiros, então o protocolo de camada de transporte não poderá dar essas mesmas garantias para mensagens de aplicação enviadas entre processos.

No entanto, certos serviços *podem* ser oferecidos por um protocolo de transporte mesmo quando o protocolo de rede subjacente não oferece o serviço correspondente na camada de rede. Por exemplo, como veremos neste capítulo, um protocolo de transporte pode oferecer serviço confiável de transferência de dados a uma aplicação mesmo quando o protocolo subjacente da rede não é confiável, isto é, mesmo quando o protocolo de rede perde, embaralha ou duplica pacotes. Como outro exemplo (que exploraremos no Capítulo 8, quando discutirmos segurança de rede), um protocolo de transporte pode usar criptografia para garantir que as mensagens da aplicação não sejam lidas por intrusos mesmo quando a camada de rede não puder garantir o sigilo de segmentos de camada de transporte.

3.1.2 Visão geral da camada de transporte na Internet

Lembre-se de que a Internet — e, de maneira mais geral, a rede TCP/IP — disponibiliza dois protocolos de transporte distintos para a camada de aplicação. Um deles é o **UDP** (User Datagram Protocol — Protocolo de Datagrama de Usuário), que provê à aplicação solicitante um serviço não confiável, não orientado para conexão. O segundo desses protocolos é o **TCP** (Transmission Control Protocol — Protocolo de Controle de Transmissão), que provê à aplicação solicitante um serviço confiável, orientado para conexão. Ao projetar uma aplicação de rede, o criador da aplicação deve especificar um desses dois protocolos de transporte. Como vimos nas seções 2.7 e 2.8, o desenvolvedor da aplicação escolhe entre o UDP e o TCP ao criar sockets.

Para simplificar a terminologia, quando no contexto da Internet, faremos alusão ao pacote de camada de transporte como um *segmento*. Devemos mencionar, contudo, que a literatura da Internet (por exemplo, os RFCs) também se refere ao pacote de camada de transporte para TCP como um segmento, mas muitas vezes se refere ao pacote para UDP como um *datagrama*. Porém, essa mesma literatura também usa o termo *datagrama* para o pacote de camada de rede! Como este é um livro de introdução a redes de computadores, acreditamos que será menos confuso se nos referirmos a ambos os pacotes TCP e UDP como segmentos; reservaremos o termo *datagrama* para o pacote de camada de rede.

Antes de continuarmos com nossa breve apresentação do UDP e do TCP, é útil dizer algumas palavras sobre a camada de rede da Internet. (A camada de rede é examinada detalhadamente no Capítulo 4.) O protocolo de camada de rede da Internet tem um nome — IP, que quer dizer *Internet Protocol*. O IP provê comunicação lógica entre hospedeiros. O modelo de serviço do IP é um **serviço de entrega de melhor esforço**, o que significa que o IP faz o 'melhor esforço' para levar segmentos entre hospedeiros comunicantes, *mas não dá nenhuma garantia*. Em particular, o IP não garante a entrega de segmentos, a entrega ordenada de segmentos e tampouco a integridade dos dados nos segmentos. Por essas razões, ele é denominado um **serviço não confiável**. Mencionamos também neste livro que cada hospedeiro tem, no mínimo, um endereço de camada de rede, denominado endereço IP. Examinaremos endereçamento IP detalhadamente no Capítulo 4. Para este capítulo, precisamos apenas ter em mente que *cada hospedeiro tem um endereço IP*.

Agora que abordamos ligeiramente o modelo de serviço IP, vamos resumir os modelos de serviço providos por UDP e TCP. A responsabilidade fundamental do UDP e do TCP é ampliar o serviço de entrega IP

entre dois sistemas finais para um serviço de entrega entre dois processos que rodam nos sistemas finais. A ampliação da entrega hospedeiro a hospedeiro para entrega processo a processo é denominada **multiplexação/demultiplexação de camada de transporte**. Discutiremos multiplexação e demultiplexação de camada de transporte na próxima seção. O UDP e o TCP também fornecem verificação de integridade ao incluir campos de detecção de erros nos cabeçalhos de seus segmentos. Esses dois serviços mínimos de camada de transporte — entrega de dados processo a processo e verificação de erros — são os únicos que o UDP fornece! Em particular, como o IP, o UDP é um serviço não confiável — ele não garante que os dados enviados por um processo cheguem (quando chegam!) intactos ao processo destinatário. O UDP será discutido detalhadamente na Seção 3.3.

O TCP, por outro lado, oferece vários serviços adicionais às aplicações. Primeiramente, e mais importante, ele provê **transferência confiável de dados**. Usando controle de fluxo, números de seqüência, reconhecimentos e temporizadores (técnicas que exploraremos detalhadamente neste capítulo), o protocolo assegura que os dados sejam entregues do processo remetente ao processo destinatário corretamente e em ordem. Assim, o TCP converte o serviço não confiável do IP entre sistemas finais em um serviço confiável de transporte de dados entre processos. Ele também provê **controle de congestionamento**. O controle de congestionamento não é tanto um serviço fornecido à aplicação solicitante; é mais um serviço dirigido à Internet como um todo — para o bem geral. Em termos genéricos, o controle de congestionamento do TCP evita que qualquer outra conexão TCP abarrote os enlaces e comutadores entre hospedeiros comunicantes com uma quantidade excessiva de tráfego. Em princípio, o TCP permite que conexões TCP trafegando por um enlace de rede congestionado compartilhem em pé de igualdade a largura de banda daquele enlace. Isso é feito pela regulagem da taxa com a qual o lado remetente do TCP pode enviar tráfego para dentro da rede. O tráfego UDP, por outro lado, não é regulado. Uma aplicação que usa transporte UDP pode enviar tráfego à taxa que quiser, pelo tempo que quiser.

Um protocolo que fornece transferência confiável de dados e controle de congestionamento é, necessariamente, complexo. Precisaremos de várias seções para detalhar os princípios da transferência confiável de dados e do controle de congestionamento, bem como de seções adicionais para explicar o protocolo TCP. Esses tópicos são analisados nas seções 3.4 a 3.8. A abordagem escolhida neste capítulo é alternar entre princípios básicos e o protocolo TCP. Por exemplo, discutiremos primeiramente a transferência confiável de dados em âmbito geral e, em seguida, como o TCP fornece especificamente a transferência confiável de dados. De maneira semelhante, discutiremos inicialmente o controle de congestionamento em âmbito geral e, em seguida, como o TCP realiza o controle de congestionamento. Mas, antes de chegarmos a essa parte boa, vamos examinar, primeiramente, multiplexação/demultiplexação na camada de transporte.

3.2 Multiplexação e demultiplexação

Nesta seção, discutiremos multiplexação e demultiplexação na camada de transporte, isto é, a ampliação do serviço de entrega hospedeiro a hospedeiro provido pela camada de rede para um serviço de entrega processo a processo para aplicações que rodam nesses hospedeiros. Para manter a discussão em nível concreto, vamos examinar esse serviço básico da camada de transporte no contexto da Internet. Enfatizamos, contudo, que o serviço de multiplexação/demultiplexação é necessário para todas as redes de computadores.

No hospedeiro de destino, a camada de transporte recebe segmentos da camada de rede logo abaixo dela e tem a responsabilidade de entregar os dados desses segmentos ao processo de aplicação apropriado que roda no hospedeiro. Vamos examinar um exemplo. Suponha que você esteja sentado à frente de seu computador, descarregando páginas Web enquanto roda uma sessão FTP e duas sessões Telnet. Por conseguinte, você tem quatro processos de aplicação de rede em execução — dois processos Telnet, um processo FTP e um processo HTTP. Quando a camada de transporte em seu computador receber dados da camada de rede abaixo dela, precisará direcionar os dados recebidos a um desses quatro processos. Vamos ver agora como isso é feito.

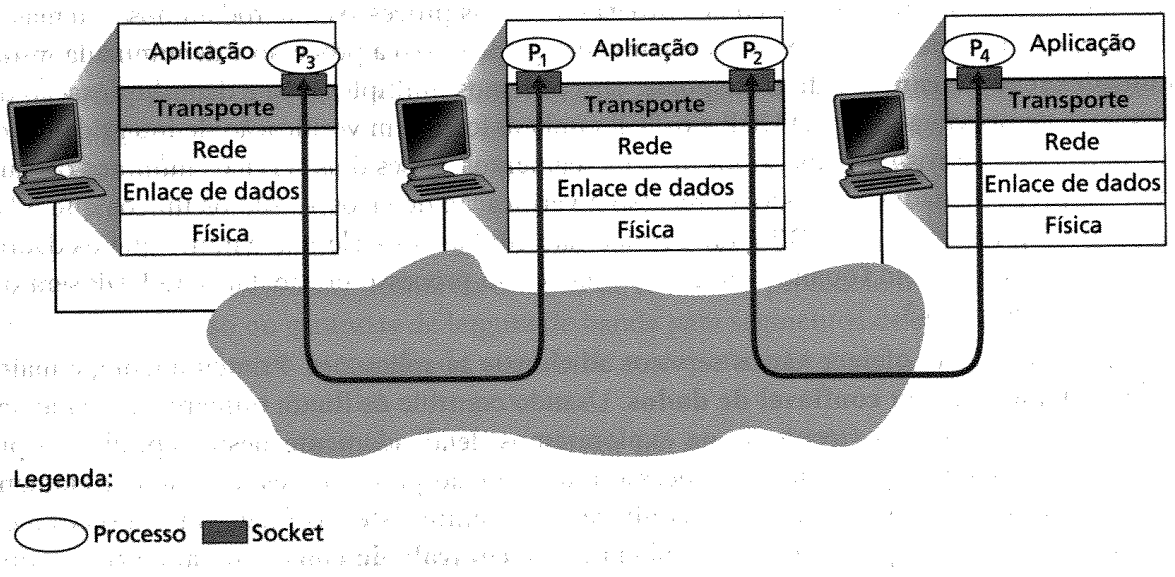


Figura 3.2 Multiplexação e demultiplexação na camada de transporte

Em primeiro lugar, lembre-se de que dissemos, nas seções 2.7 e 2.8, que um processo (como parte de uma aplicação de rede) pode ter um ou mais **sockets**, portas pelas quais dados passam da rede para o processo e do processo para a rede. Assim, como mostra a Figura 3.2, a camada de transporte do hospedeiro destinatário na verdade não entrega dados diretamente a um processo, mas a um socket intermediário. Como, a qualquer dado instante, pode haver mais de um socket no hospedeiro destinatário, cada um tem um identificador exclusivo. O formato do identificador depende de o socket ser UDP ou TCP, como discutiremos em breve.

Agora, vamos considerar como um hospedeiro destinatário direciona à porta correta um segmento de camada de transporte que chega. Cada segmento de camada de transporte tem um conjunto de campos para essa finalidade. Na extremidade receptora, a camada de transporte examina esses campos para identificar a porta receptora e direcionar o segmento a esse socket. A tarefa de entregar os dados contidos em um segmento da camada de transporte à porta correta é denominada **demultiplexação**. O trabalho de reunir, no hospedeiro de origem, porções de dados provenientes de diferentes portas, encapsular cada porção de dados com informações de cabeçalho (que mais tarde serão usadas na demultiplexação) para criar segmentos, e passar esses segmentos para a camada de rede é denominada **multiplexação**. Note que a camada de transporte do hospedeiro que está no meio da Figura 3.2 tem de demultiplexar segmentos que chegam da camada de rede abaixo para os processos P_1 ou P_2 acima; isso é feito direcionando à porta correspondente do processo os dados contidos no segmento que está chegando. A camada de transporte desse hospedeiro também tem de juntar dados de saída dessas portas, formar segmentos de camada de transporte e passá-los à camada de rede. Embora tenhamos apresentado multiplexação e demultiplexação no contexto dos protocolos de transporte da Internet, é importante entender que essas operações estarão presentes sempre que um único protocolo em uma camada (na de transporte ou em qualquer outra) for usado por vários protocolos na camada mais alta seguinte.*

Para ilustrar o serviço de demultiplexação, lembre-se da metáfora das moradias apresentada na seção anterior. Cada uma das crianças é identificada por seu nome próprio. Quando Bill recebe uma grande quantidade de correspondência do carteiro, realiza uma operação de demultiplexação ao examinar para quem as cartas estão endereçadas e, em seguida, entregar a correspondência a seus irmãos e irmãs. Ann realiza uma operação de multiplexação quando coleta as cartas de seus irmãos e irmãs e entrega a correspondência na agência do correio.

* O autor quer dizer várias entidades na camada mais alta seguinte (N. do R.T.).

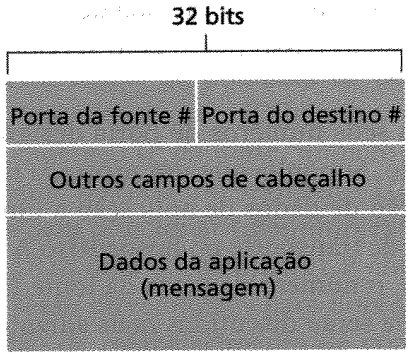


Figura 3.3 Campos de número de porta da fonte e do destino em um segmento de camada de transporte

Agora que entendemos os papéis da multiplexação e da demultiplexação na camada de transporte, vamos examinar como isso é feito em um hospedeiro. Sabemos, pela nossa discussão anterior, que multiplexação na camada de rede requer (1) que as portas tenham identificadores exclusivos e (2) que cada segmento tenha campos especiais que indiquem a porta para a qual o segmento deve ser entregue. Esses campos especiais, ilustrados na Figura 3.3, são o **campo de número de porta da fonte** e o **campo de número de porta do destino**. (Os segmentos UDP e TCP têm outros campos também, que serão examinados nas seções subseqüentes deste capítulo.) Cada número de porta é um número de 16 bits na faixa de 0 a 65535. Os números de porta entre 0 e 1023 são denominados **números de porta bem conhecidos**; eles são restritos, o que significa que estão reservados para utilização por protocolos de aplicação bem conhecidos, como HTTP (que usa a porta número 80) e FTP (que usa a porta número 21). A lista dos números de porta bem conhecidos é apresentada no RFC 1700 e atualizada em http://www.iana.org [RFC 3232]. Quando desenvolvemos uma nova aplicação (como as desenvolvidas nas seções 2.7 a 2.9), devemos atribuir a ela um número de porta.

Agora já deve estar claro como a camada de transporte *poderia* implementar o serviço de demultiplexação: cada porta do hospedeiro pode receber um número designado; quando um segmento chega ao hospedeiro, a camada de transporte examina seu número de porta de destino e o direciona à porta correspondente. Então, os dados do segmento passam através da porta e entram no processo ligado a ela. Como veremos, é assim que o UDP faz demultiplexação. Todavia, veremos também que multiplexação/demultiplexação em TCP é ainda mais sutil.

Multiplexação e demultiplexação não orientadas para conexão

Lembre-se, na Seção 2.8, de que um programa em Java que roda em um hospedeiro pode criar uma porta UDP com a linha

```
DatagramSocket mySocket = new DatagramSocket();
```

Quando uma porta UDP é criada dessa maneira, a camada de transporte automaticamente designa um número de porta ao socket. Em particular, a camada de transporte designa um número de porta na faixa de 1024 a 65535 que não esteja sendo usado naquele instante por qualquer outra porta UDP no hospedeiro. Alternativamente, um programa Java poderia criar uma porta com a linha

```
DatagramSocket mySocket = new DatagramSocket(19157);
```

Nesse caso, a aplicação designa um número de porta específico — a saber, 19157 — à porta UDP. Se o desenvolvedor responsável por escrever o código da aplicação estivesse implementando o lado servidor de um 'protocolo bem conhecido', ele teria de designar o número de porta bem conhecido correspondente. O

lado cliente da aplicação normalmente permite que a camada de transporte designe o número de porta automaticamente — e transparentemente — ao passo que o lado servidor da aplicação designa um número de porta específico.

Agora que os sockets UDP já têm seus números de porta designados, podemos descrever multiplexação/demultiplexação UDP com precisão. Suponha que um processo no hospedeiro A, cujo número de porta UDP é 19157, queira enviar uma porção de dados de aplicação a um processo cujo número de porta UDP seja 46428 no hospedeiro B. A camada de transporte no hospedeiro A cria um segmento de camada de transporte que inclui os dados de aplicação, o número da porta da fonte (19157), o número da porta de destino (46428) e mais outros dois valores (que serão discutidos mais adiante, mas que não são importantes para a discussão em curso). Então, a camada de transporte passa o segmento resultante para a camada de rede. Essa camada encapsula o segmento em um datagrama IP e faz uma tentativa de melhor esforço para entregar o segmento ao hospedeiro destinatário. Se o segmento chegar à máquina de destino B, esta examinará o número da porta de destino no segmento (46428) e o entregará a seu socket identificado pelo número 46428. Note que a máquina B poderia estar rodando vários processos, cada um com sua própria porta UDP e número de porta associado. À medida que segmentos UDP chegassem da rede, a máquina B direcionaria (demultiplexaria) cada segmento à porta apropriada examinando o número de porta de destino do segmento.

É importante notar que um socket UDP é totalmente identificado por uma tupla com dois elementos, consistindo em um endereço IP de destino e um número de porta de destino. Conseqüentemente, se dois segmentos UDP tiverem endereços IP de fonte e/ou números de porta de fonte diferentes, porém o mesmo endereço IP de *destino* e o mesmo número de porta de *destino*, eles serão direcionados ao mesmo processo de destino por meio do mesmo socket de destino.

É possível que agora você esteja imaginando qual é a finalidade do número de porta da fonte. Como mostra a Figura 3.4, no segmento A-B, o número de porta da fonte serve como parte de um "endereço de retorno" — quando B quer enviar um segmento de volta para A, a porta de destino no segmento B-A tomará seu valor do valor da porta de fonte do segmento A-B. (O endereço de retorno completo é o endereço IP e o número de porta de fonte de A). Como exemplo, lembre-se do programa servidor UDP que estudamos na Seção 2.8. Em `UDPServer.java`, o servidor usa um método para extrair o número de porta da fonte do segmento que recebe do cliente; então envia um novo segmento ao cliente, com o número de porta que extraiu servindo como o número de porta de destino desse novo segmento.

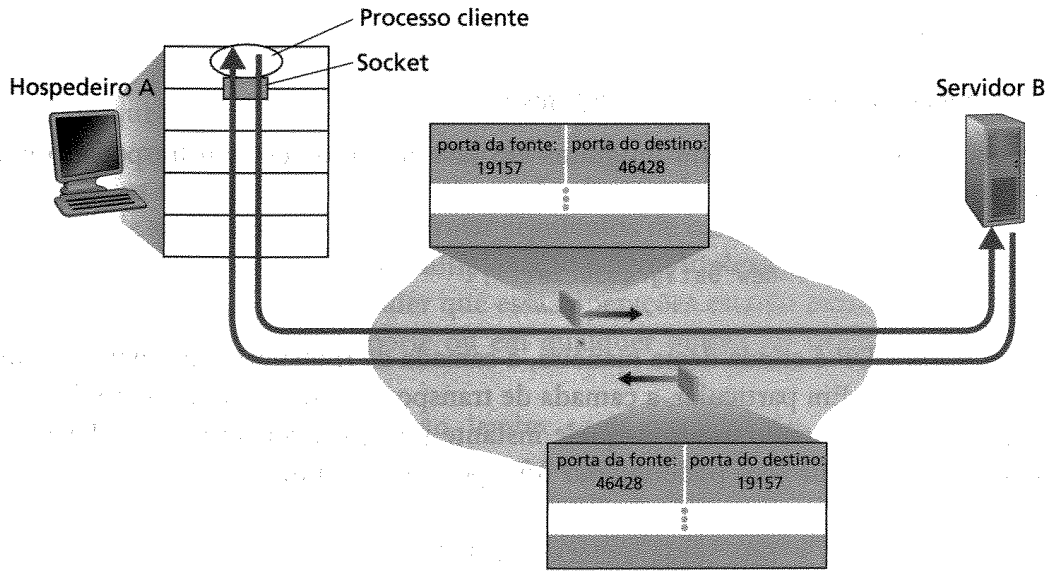


Figura 3.4 A inversão de números de porta de fonte e de destino

Multiplexação e demultiplexação orientadas para conexão

Para entender demultiplexação TCP, temos de examinar de perto sockets TCP e estabelecimento de conexão TCP. Há uma diferença sutil entre um socket UDP e um socket TCP: o socket TCP é identificado por uma tupla de quatro elementos: (endereço IP da fonte, número de porta da fonte, endereço IP de destino, número de porta do destino). Assim, quando um segmento TCP que vem da rede chega a um hospedeiro, este usa todos os quatro valores para direcionar (demultiplexar) o segmento para o socket apropriado. Em particular, e ao contrário do UDP, dois segmentos TCP chegando com endereços IP de fonte ou números de porta de fonte diferentes serão direcionados para dois sockets diferentes (com exceção de um TCP que esteja carregando a requisição de estabelecimento de conexão original). Para perceber melhor, vamos considerar novamente o exemplo de programação cliente-servidor TCP apresentado na Seção 2.7:

- A aplicação servidor TCP tem um socket de entrada que espera requisições de estabelecimento de conexão vindas de clientes TCP (ver Figura 2.27) na porta número 6789.
- O cliente TCP gera um segmento de estabelecimento de conexão com a linha

```
Socket clientSocket = new Socket("serverHostName", 6789);
```

- Uma requisição de estabelecimento de conexão nada mais é do que um segmento TCP com número de porta de destino 6789 e um bit especial de estabelecimento de conexão ajustado no cabeçalho TCP (que será discutido na Seção 3.5). O segmento inclui também um número de porta de fonte que foi escolhido pelo cliente. A linha acima cria ainda um socket TCP para o processo cliente, através do qual dados podem entrar e sair do processo cliente.
- Quando o sistema operacional do computador que está rodando o processo servidor recebe o segmento de requisição de conexão que está chegando e cuja porta de destino é 6789, ele localiza o processo servidor que está esperando para aceitar uma conexão na porta número 6789. Então, o processo servidor cria uma nova conexão:

```
Socket connectionSocket = welcomeSocket.accept();
```

- O servidor também nota os quatro valores seguintes no segmento de requisição de conexão: (1) o número de porta da fonte no segmento, (2) o endereço IP do hospedeiro da fonte, (3) o número de porta do destino no segmento e (4) seu próprio endereço IP. O socket de conexão recém-criado é identificado por esses quatro valores; todos os segmentos subseqüentes que chegarem, cuja porta da fonte, endereço IP da fonte, porta de destino e endereço IP de destino combinarem com esses quatro valores, serão demultiplexados para essa porta. Com a conexão TCP agora ativa, o cliente e o servidor podem enviar dados um para o outro.

O hospedeiro servidor pode suportar vários sockets TCP simultâneos, sendo cada um ligado a um processo e identificado por sua própria tupla de quatro elementos. Quando um segmento TCP chega ao hospedeiro, todos os quatro campos (endereço IP da fonte, porta da fonte, endereço IP de destino, porta de destino) são usados para direcionar (demultiplexar) o segmento para o socket apropriado.

A situação é ilustrada na Figura 3.5, na qual o hospedeiro C inicia duas sessões HTTP para o servidor B, e o hospedeiro A inicia uma sessão HTTP para o servidor B.

Os hospedeiros A e C e o servidor B possuem, cada um, seu próprio endereço IP exclusivo: A, C e B, respectivamente. O hospedeiro C atribui dois números diferentes (26145 e 7532) de porta da fonte às suas duas conexões HTTP. Como o hospedeiro A está escolhendo números de porta independentemente de C, ele poderia também atribuir um número de porta da fonte 26145 à sua conexão HTTP. Apesar disso, o servidor B ainda será capaz de demultiplexar corretamente as duas conexões que têm o mesmo número de porta de fonte, já que elas têm endereços IP de fonte diferentes.

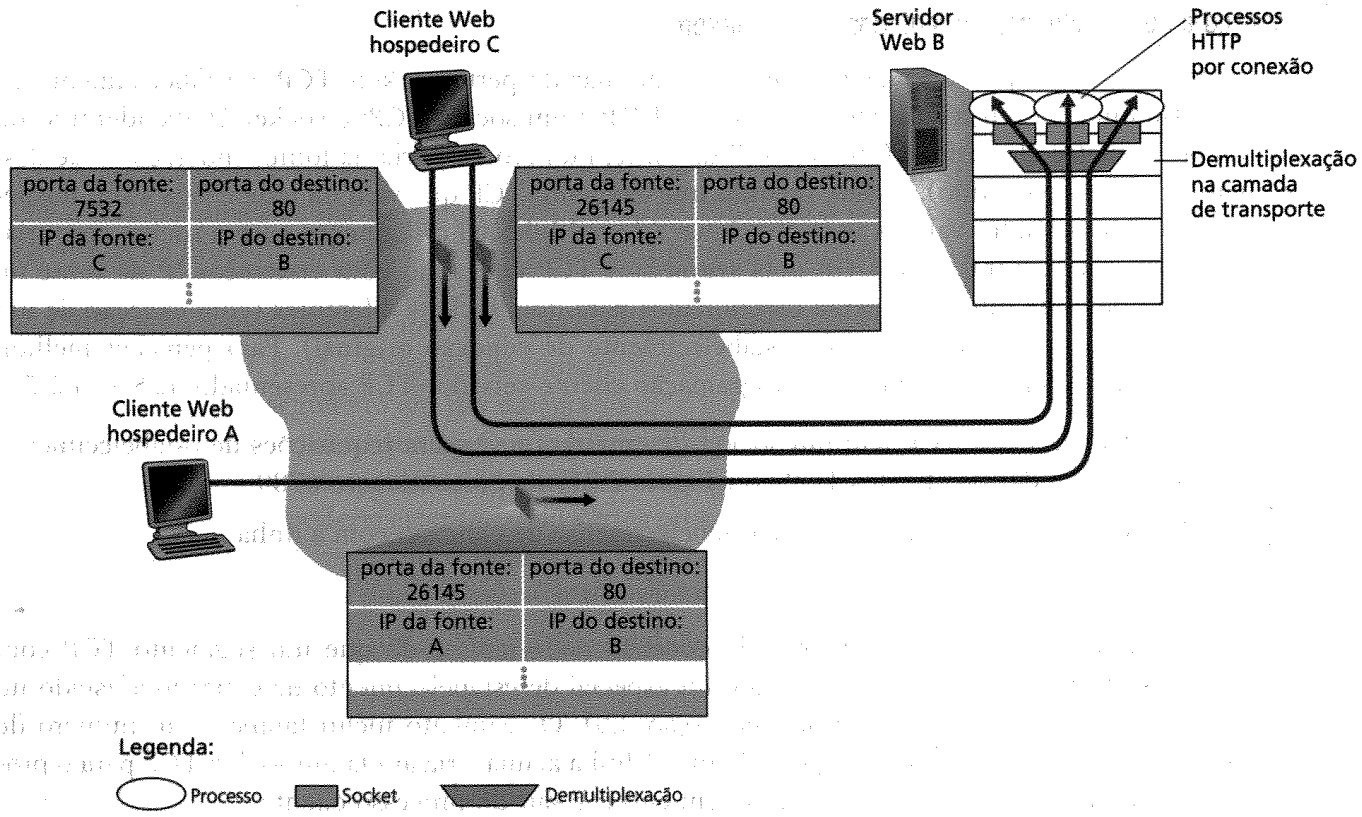


Figura 3.5 Dois clientes que usam o mesmo número de porta de destino (80) para se comunicar com a mesma aplicação de servidor Web

Servidores Web e TCP

Antes de encerrar essa discussão, é instrutivo falar um pouco mais sobre servidores Web e como eles usam números de porta. Considere um hospedeiro rodando um servidor Web, tal como um Apache, na porta 80. Quando clientes (por exemplo, browsers) enviam segmentos ao servidor, *todos* os segmentos terão a porta de destino 80. Em particular, os segmentos que estabelecem a conexão inicial e os segmentos que carregam as mensagens de requisição HTTP, ambos terão a porta de destino 80. Como acabamos de descrever, o servidor distingue os segmentos dos diferentes clientes pelos endereços IP e pelos números de porta da fonte.

Servidores Web normalmente geram um novo processo ou criam uma nova thread para cada nova conexão cliente. Como mostra a Figura 3.5, cada um desses processos tem seu próprio socket de conexão através do qual chegam requisições HTTP e são enviadas respostas HTTP. Mencionamos, contudo, que nem sempre existe uma correspondência unívoca entre sockets de conexão e processos. Na verdade, os servidores Web de alto desempenho atuais muitas vezes utilizam somente um processo, mas criam uma nova thread com um novo socket de conexão para cada nova conexão cliente. (Uma thread pode ser considerada um subprocesso leve.) Se você fez a primeira tarefa de programação do Capítulo 2, construiu um servidor Web que faz exatamente isso. Para um servidor desses, a qualquer dado instante podem haver muitos sockets de conexão (com identificadores diferentes) ligados ao mesmo processo.

Se o cliente e o servidor estiverem usando HTTP persistente, então, durante toda a conexão persistente, eles trocarão mensagens HTTP através do mesmo socket do servidor. Todavia, se o cliente e o servidor usarem HTTP não persistente, então uma nova conexão TCP é criada e encerrada para cada requisição/resposta e, portanto, um novo socket é criado e mais tarde encerrado para cada requisição/resposta. Essa criação e encerramento freqüentes de sockets podem causar sério impacto sobre o desempenho de um servidor Web movimentado (embora o sistema operacional possa usar vários estratagemas para atenuar o problema).

Aconselhamos o leitor interessado em questões de sistema operacional referentes a HTTP persistente e não persistente a consultar [Nielsen, 1997; Nahum, 2002].

Agora que já discutimos multiplexação e demultiplexação na camada de transporte, passemos à discussão de um dos protocolos da Internet, o UDP. Na próxima seção, veremos que a contribuição desse protocolo não é muito mais do que um serviço de multiplexação/demultiplexação.

3.3 Transporte não orientado para conexão: UDP

Nesta seção, examinaremos o UDP mais de perto, como ele funciona e o que ele faz. Aconselhamos o leitor a rever o material apresentado na Seção 2.1, que inclui uma visão geral do modelo de serviço UDP, e o da Seção 2.8, que discute a programação de portas por UDP.

Para motivar nossa discussão sobre UDP, suponha que você esteja interessado em projetar um protocolo de transporte simples, bem básico. Como você faria isso? De início, você deve considerar a utilização de um protocolo de transporte vazio. Em particular, do lado do remetente, considere pegar as mensagens do processo de aplicação e passá-las diretamente para a camada de rede; do lado do destinatário, considere pegar as mensagens que chegam da camada de rede e passá-las diretamente ao processo da aplicação. Mas, como aprendemos na seção anterior, o que teremos de fazer é praticamente nada. No mínimo, a camada de transporte tem de fornecer um serviço de multiplexação/demultiplexação para passar os dados da camada de rede ao processo de aplicação correto.

O UDP, definido no RFC 768, faz apenas quase tão pouco quanto um protocolo de transporte pode fazer. À parte sua função de multiplexação/demultiplexação e de alguma verificação de erros, ele nada adiciona ao IP. Na verdade, se o criador da aplicação escolher o UDP, em vez do TCP, a aplicação estará 'falando' quase diretamente com o IP. O UDP pega as mensagens do processo de aplicação, anexa os campos de número de porta da fonte e do destino para o serviço de multiplexação/demultiplexação, adiciona dois outros pequenos campos e passa o segmento resultante à camada de rede, que encapsula o segmento dentro de um datagrama IP e, em seguida, faz uma tentativa de melhor esforço para entregar o segmento ao hospedeiro receptor. Se o segmento chegar ao hospedeiro receptor, o UDP usará o número de porta de destino para entregar os dados do segmento ao processo de aplicação correto. Note que, com o UDP, não há apresentação entre as entidades remetente e destinatária da camada de transporte antes de enviar um segmento. Por essa razão, dizemos que o UDP é não orientado para conexão.

O DNS é um exemplo de protocolo de camada de aplicação que usa o UDP. Quando a aplicação DNS em um hospedeiro quer fazer uma consulta, constrói uma mensagem de consulta DNS e passa a mensagem para o UDP. Sem realizar nenhuma apresentação com a entidade UDP que está funcionando no sistema final de destino, o UDP do lado do hospedeiro adiciona campos de cabeçalho à mensagem e passa o segmento resultante à camada de rede, que encapsula o segmento UDP em um datagrama e o envia a um servidor de nomes. A aplicação DNS no hospedeiro requisitante então espera por uma resposta à sua consulta. Se não receber uma resposta (possivelmente porque a rede subjacente perdeu a consulta ou a resposta), ela tentará enviar a consulta a outro servidor de nomes ou informará à aplicação consultante que não pode obter uma resposta.

É possível que agora você esteja imaginando por que um criador de aplicação escolheria construir uma aplicação sobre UDP, em vez de sobre TCP. O TCP não é sempre preferível ao UDP, já que fornece serviço confiável de transferência de dados e o UDP não? A resposta é 'não', pois muitas aplicações se adaptam melhor ao UDP pelas seguintes razões:

- *Melhor controle no nível da aplicação sobre quais dados são enviados e quando.* Com UDP, tão logo um processo de aplicação passe dados ao UDP, o protocolo empacotará esses dados dentro de um segmento UDP e os passará imediatamente à camada de rede. O TCP, por outro lado, tem um mecanismo de controle de congestionamento que limita o remetente TCP da camada de transporte quando um ou mais enlaces entre os hospedeiros da fonte e do destinatário ficam excessivamente

congestionados. O TCP também continuará a reenviar um segmento até que o hospedeiro destinatário reconheça a recepção desse segmento, independentemente do tempo que a entrega confiável levar. Visto que aplicações de tempo real requerem uma taxa mínima de envio, não querem atrasar excessivamente a transmissão de segmentos e podem tolerar uma certa perda de dados, o modelo de serviço do TCP não é particularmente compatível com essas necessidades das aplicações. Como discutiremos mais adiante, essas aplicações podem usar UDP e implementar, como parte da aplicação, qualquer funcionalidade adicional necessária além do serviço de entrega de segmentos simples e básico do UDP.

- *Não há estabelecimento de conexão.* Como discutiremos mais adiante, o TCP usa uma apresentação de três vias antes de começar a transferir dados. O UDP simplesmente envia mensagens sem nenhuma preliminar formal e, assim, não introduz nenhum atraso para estabelecer uma conexão. Provavelmente esta é a principal razão pela qual o DNS roda sobre UDP, e não sobre TCP — o DNS seria muito mais lento se rodasse em TCP. O HTTP usa o TCP, e não o UDP, porque a confiabilidade é fundamental para páginas Web com texto. Mas, como discutimos brevemente na Seção 2.2, o atraso de estabelecimento de uma conexão TCP é uma contribuição importante aos atrasos associados à recepção de documentos Web.
- *Não há estados de conexão.* O TCP mantém o estado de conexão nos sistemas finais. Esse estado inclui buffers de envio e recebimento, parâmetros de controle de congestionamento e parâmetros numéricos de seqüência e de reconhecimento. Veremos na Seção 3.5 que essa informação de estado é necessária para implementar o serviço de transferência confiável de dados do TCP e para prover controle de congestionamento. O UDP, por sua vez, não mantém o estado de conexão e não monitora nenhum desses parâmetros. Por essa razão, um servidor devotado a uma aplicação específica pode suportar um número muito maior de clientes ativos quando a aplicação roda sobre UDP e não sobre TCP.
- *Pequena sobrecarga de cabeçalho de pacote.* O segmento TCP tem 20 bytes de sobrecarga de cabeçalho além dos dados para cada segmento, enquanto o UDP tem somente 8 bytes de sobrecarga.

A Figura 3.6 relaciona aplicações populares da Internet e os protocolos de transporte que elas usam. Como era de esperar, o e-mail, o acesso a terminal remoto, a Web e a transferência de arquivos rodam sobre TCP — todas essas aplicações necessitam do serviço confiável de transferência de dados do TCP. Não obstante, muitas aplicações importantes executam sobre UDP, e não sobre TCP. O UDP é usado para atualização das tabelas de roteamento com o protocolo RIP (*routing information protocol* — protocolo de informação de roteamento) (veja Seção 4.6.1). Como as atualizações RIP são enviadas periodicamente (em geral, a cada 5 minutos), atualizações perdidas serão substituídas por atualizações mais recentes, tornando inútil a recuperação das atualizações perdidas. O UDP também é usado para levar dados de gerenciamento de rede (SNMP; veja o Capítulo 9). Nesse caso, o UDP é preferível ao TCP, já que aplicações de gerenciamento de rede freqüentemente devem funcionar quando a rede está em estado sobrecarregado — exatamente quando é difícil conseguir transferência confiável de dados com congestionamento controlado. E também, como mencionamos anteriormente, o DNS roda sobre UDP, evitando, desse modo, atrasos de estabelecimento de conexões TCP.

Como mostra a Figura 3.6, hoje o UDP também é comumente usado para aplicações de multimídia, como telefone por Internet, videoconferência em tempo real e recepção de áudio e vídeo armazenados. Examinaremos essas aplicações mais de perto no Capítulo 7. No momento, mencionamos apenas que todas essas aplicações podem tolerar uma pequena quantidade de perda de pacotes, de modo que a transferência confiável de dados não é absolutamente crítica para o sucesso da aplicação. Além disso, aplicações em tempo real, como telefone por Internet e videoconferência, reagem muito mal ao controle de congestionamento do TCP. Por essas razões, os desenvolvedores de aplicações de multimídia muitas vezes optam por rodar suas aplicações sobre UDP em vez de sobre TCP.

Aplicação	Protocolo de camada de aplicação	Protocolo de transporte subjacente
Correio eletrônico	SMTP	TCP
Acesso a terminal remoto	Telnet	TCP
Web	HTTP	TCP
Transferência de arquivo	FTP	TCP
Servidor remoto de arquivo	NFS	tipicamente UDP
Recepção de multimídia	tipicamente proprietária	tipicamente UDP
Telefonia por Internet	tipicamente proprietária	tipicamente UDP
Gerenciamento de rede	SNMP	tipicamente UDP
Protocolo de roteamento	RIP	tipicamente UDP
Tradução de nome	DNS	tipicamente UDP

Figura 3.6 Aplicações populares da Internet e seus protocolos de transporte subjacentes

Embora atualmente seja comum rodar aplicações de multimídia sobre UDP, isso é, no mínimo, controvertido. Como já mencionamos, o UDP não tem controle de congestionamento. Mas esse controle é necessário para evitar que a rede entre em um estado no qual pouquíssimo trabalho útil é realizado. Se todos começassem a enviar vídeo com alta taxa de bits sem usar nenhum controle de congestionamento, haveria tamanho transbordamento de pacotes nos roteadores que poucos pacotes UDP conseguiriam atravessar com sucesso o caminho da fonte ao destino. Além do mais, as altas taxas de perda induzidas pelos remetentes UDP sem controle fariam com que os remetentes TCP (que, como veremos mais adiante, *reduzem* suas taxas de envio em face de congestionamento) reduzissem drasticamente suas taxas. Assim, a falta de controle de congestionamento no UDP pode resultar em altas taxas de perda entre um remetente e um destinatário UDP e no acúmulo de sessões TCP — um problema potencialmente sério [Floyd, 1999]. Muitos pesquisadores propuseram novos mecanismos para forçar todas as fontes, inclusive as fontes UDP, a realizar um controle de congestionamento adaptativo [Mahdavi, 1997; Floyd, 2000; Kohler, 2004].

Antes de discutirmos a estrutura do segmento UDP, mencionaremos que *é* possível que uma aplicação tenha transferência confiável de dados usando UDP. Isso pode ser feito se a confiabilidade for embutida na própria aplicação (por exemplo, adicionando mecanismos de reconhecimento e de retransmissão, tais como os que estudaremos na próxima seção). Mas esta é uma tarefa não trivial que manteria o desenvolvedor ocupado com a depuração por um longo tempo. Não obstante, embutir confiabilidade diretamente na aplicação permite que ela tire proveito de ambas as alternativas. Em outras palavras, os processos de aplicação podem se comunicar de maneira confiável sem ter de se sujeitar às limitações da taxa de transmissão impostas pelo mecanismo de controle de congestionamento do TCP. Muitas das atuais aplicações proprietárias de áudio e vídeo fazem exatamente isso — rodam sobre UDP, mas dispõem de reconhecimentos e retransmissões embutidos na aplicação para reduzir a perda de pacotes. Veja, por exemplo, [Rhee, 1998] e a discussão de transporte de multimídia no Capítulo 7.

3.3.1 Estrutura do segmento UDP

A estrutura do segmento UDP, mostrada na Figura 3.7, é definida no RFC 768. Os dados da aplicação ocupam o campo de dados do segmento UDP. Por exemplo, para o DNS, o campo de dados contém uma mensagem de consulta ou uma mensagem de resposta. Para uma aplicação de recepção de áudio, parcelas de áudio preenchem o campo de dados. O cabeçalho UDP tem apenas quatro campos, cada um consistindo em 2 bytes. Como já discutido na seção anterior, os números de porta permitem que o hospedeiro destinatário passe os dados da aplicação ao processo correto que está funcionando na máquina destinatária (isto é, realize a função de demultiplexação). A soma de verificação é usada pelo hospedeiro receptor para

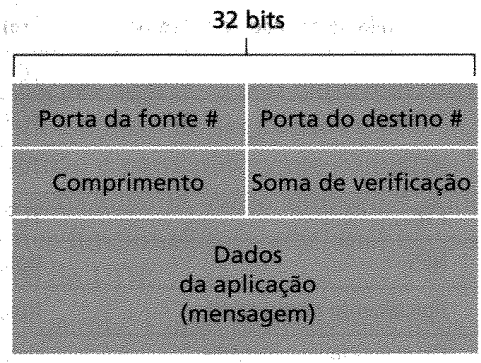


Figura 3.7 Estrutura do segmento UDP

verificar se foram introduzidos erros no segmento. Na verdade, a soma de verificação também é calculada para alguns dos campos no cabeçalho IP, além do segmento UDP. Mas ignoramos esse detalhe para podermos enxergar a floresta por entre as árvores. Discutiremos o cálculo da soma de verificação mais adiante. Os princípios básicos da detecção de erros estão descritos na Seção 5.2. O campo de comprimento especifica o comprimento do segmento UDP, incluindo o cabeçalho, em bytes.

3.3.2 Soma de verificação UDP

A soma de verificação UDP serve para detectar erros. Em outras palavras, é usada para determinar se bits dentro do segmento UDP foram alterados (por exemplo, por ruído nos enlaces ou enquanto armazenados em um roteador) durante sua movimentação da fonte até o destino. O UDP no lado remetente realiza o complemento de 1 da soma de todas as palavras de 16 bits do segmento levando em conta o "vai um" em toda a soma. Esse resultado é colocado no campo de soma de verificação no segmento UDP. Damos aqui um exemplo simples do cálculo da soma de verificação. Se quiser saber detalhes sobre a implementação eficiente do algoritmo de cálculo e sobre o desempenho com dados reais, consulte o RFC 1071 e [Stone, 1998 e 2000], respectivamente. Como exemplo, suponha que tenhamos as seguintes três palavras de 16 bits:

0110011001100000
0101010101010101
1000111100001100

A soma das duas primeiras dessas palavras de 16 bits é:

0110011001100000
0101010101010101
1011101110110101

Adicionando a terceira palavra à soma acima, temos:

1011101110110101
1000111100001100
0100101011000010

Note que essa última adição teve "vai um" no bit mais significativo que foi somado ao bit menos significativo. O complemento de 1 é obtido pela conversão de todos os 0 em 1 e de todos os 1 em 0. Desse modo, o complemento de 1 da soma 0100101011000010 é 1011010100111101, que passa a ser a soma de verificação. No destinatário, todas as quatro palavras de 16 bits são somadas, inclusive a soma de verificação. Se nenhum erro for introduzido no pacote, a soma no destinatário será, obviamente, 1111111111111111. Se um dos bits for um zero, saberemos então que houve introdução de erro no pacote.

Provavelmente você está imaginando por que o UDP fornece uma soma de verificação em primeiro lugar, visto que muitos protocolos de camada de enlace (dentre os quais, o popular protocolo Ethernet) também fornecem verificação de erros. A razão é que não há garantia de que todos os enlaces entre a origem e o destino forneçam verificação de erros — um dos enlaces pode usar um protocolo de camada de enlace que não forneça verificação de erros. Além disso, mesmo que os segmentos sejam corretamente transmitidos por um enlace, é possível haver introdução de erros de bits quando um segmento é armazenado na memória de um roteador. Dado que não são garantidas nem a confiabilidade enlace a enlace, nem a detecção de erro na memória, o UDP deve prover detecção de erro *fim-a-fim* na camada de transporte se quisermos que o serviço de transferência de dados fim-a-fim forneça detecção de erro. Esse é um exemplo do famoso **princípio fim-a-fim** do projeto de sistemas [Saltzer, 1984]. Esse princípio afirma que, uma vez que é dado como certo que funcionalidades (detecção de erro, neste caso) devem ser implementadas fim-a-fim, "funções colocadas nos níveis mais baixos podem ser redundantes ou de pouco valor em comparação com o custo de fornecê-las no nível mais alto".

Como se pretende que o IP rode sobre qualquer protocolo de camada 2, é útil que a camada de transporte forneça verificação de erros como medida de segurança. Embora o UDP forneça verificação de erros, ele nada faz para recuperar um erro. Algumas implementações do UDP simplesmente descartam o segmento danificado; outras passam o segmento errado à aplicação acompanhado de um aviso.

Isso encerra nossa discussão sobre o UDP. Logo veremos que o TCP oferece transferência confiável de dados às suas aplicações, bem como outros serviços que o UDP não oferece. Naturalmente, o TCP também é mais complexo do que o UDP. Contudo, antes de discutirmos o TCP, primeiramente devemos examinar os princípios subjacentes da transferência confiável de dados.

3.4 Princípios da transferência confiável de dados

Nesta seção, consideramos o problema conceitual da transferência confiável de dados. Isso é apropriado, já que o problema de implementar transferência confiável de dados ocorre não somente na camada de transporte, mas também na camada de enlace e na camada de aplicação. Assim, o problema geral é de importância central para o trabalho em rede. Na verdade, se tivéssemos de fazer uma lista dos dez maiores problemas fundamentalmente importantes para o trabalho em rede, o da transferência confiável de dados seria o candidato número um da lista. Na seção seguinte, examinaremos o TCP e mostraremos, em particular, que ele utiliza muitos dos princípios que descreveremos aqui.

A Figura 3.8 ilustra a estrutura para nosso estudo de transferência confiável de dados. A abstração do serviço fornecido às entidades das camadas superiores é a de um canal confiável através do qual dados podem ser transferidos. Com um canal confiável, nenhum dos dados transferidos é corrompido (trocado de 0 para 1 ou vice-versa) nem perdido, e todos são entregues na ordem em que foram enviados. Este é exatamente o modelo de serviço oferecido pelo TCP às aplicações de Internet que recorrem a ele.

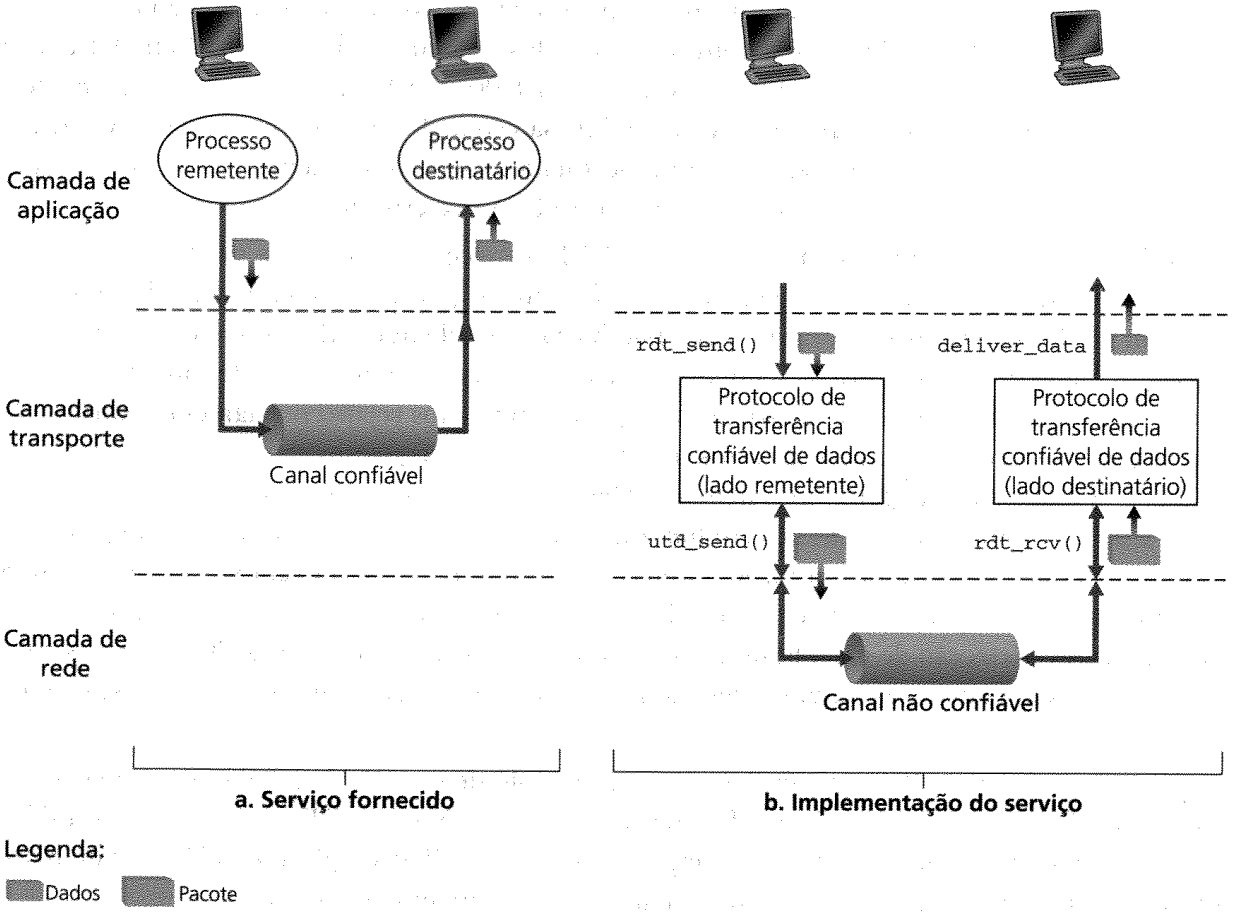


Figura 3.8 Transferência confiável de dados: modelo do serviço e implementação do serviço

É responsabilidade de um **protocolo de transferência confiável de dados** implementar essa abstração de serviço. A tarefa é dificultada pelo fato de que a camada *abaixo* do protocolo de transferência confiável de dados pode ser não confiável. Por exemplo, o TCP é um protocolo confiável de transferência de dados que é implementado sobre uma camada de rede fim-a-fim não confiável (IP). De modo mais geral, a camada abaixo das duas extremidades que se comunicam confiavelmente pode consistir em um único enlace físico (como é o caso, por exemplo, de um protocolo de transferência de dados na camada de enlace) ou em uma rede global interligada (como é o caso de um protocolo de camada de transporte). Para nossa finalidade, contudo, podemos considerar essa camada mais baixa simplesmente como um canal ponto a ponto não confiável.

Nesta seção, desenvolveremos gradualmente os lados remetente e destinatário de um protocolo confiável de transferência de dados, considerando modelos progressivamente mais complexos do canal subjacente. A Figura 3.8(b) ilustra as interfaces de nosso protocolo de transferência de dados. O lado remetente do protocolo de transferência de dados será invocado de cima, por uma chamada a `rdt_send( )`. Ele passará os dados a serem entregues à camada superior no lado destinatário. (Aqui, `rdt` representa o protocolo de transferência confiável de dados [*reliable data transfer*] e `_send` indica que o lado remetente do `rdt` está sendo chamado. O primeiro passo no desenvolvimento de qualquer protocolo é dar-lhe um bom nome!) Do lado destinatário, `rdt_rcv( )` será chamado quando um pacote chegar do lado destinatário do canal. Quando o protocolo `rdt` quiser entregar dados à camada superior, ele o fará chamando `deliver_data( )`. No que se segue, usamos a terminologia 'pacote' em vez de 'segmento' de camada de transporte. Como a teoria desenvolvida nesta seção se aplica a redes de computadores em geral, e não somente à camada de transporte da Internet, o termo genérico 'pacote' talvez seja mais apropriado aqui.

Nesta seção, consideramos apenas o caso de **transferência unidirecional de dados**, isto é, transferência de dados do lado remetente ao lado destinatário. O caso de **transferência bidirecional** confiável de dados (isto é, *full-duplex*) não é conceitualmente mais difícil, mas é bem mais tedioso de explicar. Embora consideremos apenas a transferência unidirecional de dados, é importante notar que, apesar disso, os lados remetente e destinatário de nosso protocolo terão de transmitir pacotes em ambas as direções, como mostra a Figura 3.8. Logo veremos que, além de trocar pacotes contendo os dados a transferir, os lados remetente e destinatário do `rdt` também precisarão trocar pacotes de controle entre si. Ambos os lados de envio e destino do `rdt` enviam pacotes para o outro lado por meio de uma chamada a `udt_send( )` (em que `udt` representa transferência não confiável de dados — *unreliable data transfer*).

3.4.1 Construindo um protocolo de transferência confiável de dados

Vamos percorrer agora uma série de protocolos que vão se tornando cada vez mais complexos, até chegar a um protocolo de transferência confiável de dados impecável.

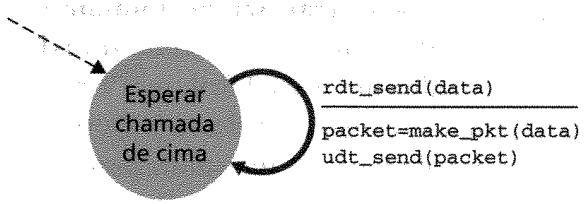
Transferência confiável de dados sobre um canal perfeitamente confiável: `rdt1.0`

Vamos considerar primeiramente o caso mais simples, em que o canal subjacente é completamente confiável. O protocolo em si, que denominaremos `rdt1.0`, é trivial. As definições de **máquina de estado finito** (*finite-state machine* — FSM) para o remetente e o destinatário `rdt1.0` são apresentadas na Figura 3.9. A FSM da Figura 3.9(a) define a operação do remetente, enquanto a FSM da Figura 3.9(b) define a operação do destinatário. É importante notar que há FSM *separadas* para o remetente e o destinatário. Ambas as FSM da Figura 3.9 têm apenas um estado. As setas na descrição da FSM indicam a transição do protocolo de um estado para outro. (Uma vez que cada FSM da Figura 3.9 tem apenas um estado, uma transição é, necessariamente, de um dado estado para ele mesmo; examinaremos diagramas de estados mais complicados em breve.) O evento que causou a transição é mostrado acima da linha horizontal que a rotula, e as ações realizadas quando ocorre o evento são mostradas abaixo da linha horizontal. Quando nenhuma ação é realizada em um evento, ou quando não ocorre nenhum evento e uma ação é realizada, usaremos o símbolo Λ, acima ou abaixo da linha horizontal, para indicar explicitamente a falta de uma ação ou de um evento, respectivamente. O estado inicial da FSM é indicado pela seta tracejada. Embora as FSMs da Figura 3.9 tenham somente um estado, as outras que veremos em breve têm vários estados, portanto, será importante identificar o estado inicial de cada FSM.

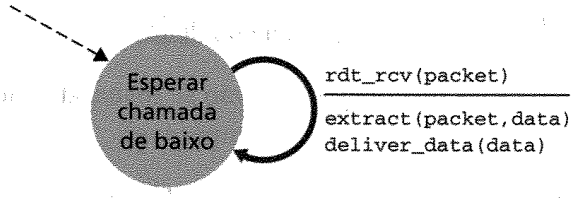
O lado remetente do `rdt` simplesmente aceita dados da camada superior pelo evento `rdt_send(data)`, cria um pacote que contém os dados (pela ação `make_pkt(data)`) e envia-o para dentro do canal. Na prática, o evento `rdt_send(data)` resultaria de uma chamada de procedimento (por exemplo, para `rdt_send()`) pela aplicação da camada superior.

Do lado destinatário, `rdt` recebe um pacote do canal subjacente pelo evento `rdt_rcv(packet)`, extrai os dados do pacote (pela ação `extract(packet, data)`) e os passa para a camada superior (pela ação `deliver_data(data)`). Na prática, o evento `rdt_ rcv(packet)` resultaria de uma chamada de procedimento (por exemplo, para `rdt_rcv( )`) do protocolo da camada inferior.

Nesse protocolo simples, não há diferença entre a unidade de dados e um pacote. E, também, todo o fluxo de pacotes corre do remetente para o destinatário; com um canal perfeitamente confiável, não há necessidade de o lado destinatário fornecer qualquer informação ao remetente, já que nada pode dar errado! Note que também admitimos que o destinatário está capacitado a receber dados seja qual for a velocidade em que o remetente os envie. Assim, não há necessidade de pedir para o remetente desacelerar!



a. rdt1.0: lado remetente



b. rdt1.0: lado destinatário

Figura 3.9 `rdt1.0` — Um protocolo para um canal completamente confiável

Transferência confiável de dados por um canal com erros de bits: `rdt2.0`

Um modelo mais realista de canal subjacente é um canal em que os bits de um pacote podem ser corrompidos. Esses erros de bits ocorrem normalmente nos componentes físicos de uma rede enquanto o pacote é transmitido, propagado ou armazenado. Continuaremos a admitir, por enquanto, que todos os pacotes transmitidos sejam recebidos (embora seus bits possam estar corrompidos) na ordem em que foram enviados.

Antes de desenvolver um protocolo para se comunicar de maneira confiável com esse canal, considere primeiramente como as pessoas enfrentariam uma situação como essa. Considere como você ditaria uma mensagem longa pelo telefone. Em um cenário típico, quem estivesse anotando a mensagem diria 'o.k.' após cada sentença que ouvisse, entendesse e anotasse. Se a pessoa ouvisse uma mensagem truncada, pediria que você a repetisse. Esse protocolo de ditado de mensagem usa **reconhecimentos positivos** ('o.k.') e **reconhecimentos negativos** ('Repita, por favor'). Essas mensagens de controle permitem que o destinatário faça o remetente saber o que foi recebido corretamente e o que foi recebido com erro e, portanto, exige repetição. Em um arranjo de rede de computadores, protocolos de transferência confiável de dados baseados nesse tipo de retransmissão são conhecidos como **protocolos ARQ (Automatic Repeat reQuest — solicitação automática de repetição)**.

Essencialmente, são exigidas três capacitações adicionais dos protocolos ARQ para manipular a presença de erros de bits:

- *Detecção de erros.* Primeiramente, é preciso um mecanismo que permita ao destinatário detectar quando ocorrem erros. Lembre-se de que dissemos na seção anterior que o UDP usa o campo de soma de verificação da Internet exatamente para essa finalidade. No Capítulo 5, examinaremos, com mais detalhes, técnicas de detecção e de correção de erros. Essas técnicas permitem que o destinatário detecte e possivelmente corrija erros de bits de pacotes. Por enquanto, basta saber que essas técnicas exigem que bits extras (além dos bits dos dados originais a serem transferidos) sejam enviados do remetente ao destinatário. Esses bits são colocados no campo de soma de verificação do pacote de dados do protocolo `rdt2.0`.
- *Realimentação do destinatário.* Uma vez que remetente e destinatário normalmente estejam rodando em sistemas finais diferentes, possivelmente separados por milhares de quilômetros, o único modo de o remetente saber qual é a visão de mundo do destinatário (neste caso, se um pacote foi

recebido corretamente ou não) é o destinatário fornecer realimentação explícita ao remetente. As respostas de reconhecimento positivo (ACK) ou negativo (NAK) no cenário do ditado da mensagem são exemplos dessa realimentação. Nosso protocolo `rdt2.0` devolverá, dessa mesma maneira, pacotes ACK e NAK do destinatário ao remetente. Em princípio, esses pacotes precisam apenas ter o comprimento de um bit; por exemplo, um valor 0 poderia indicar um NAK e um valor 1 poderia indicar um ACK.

- *Retransmissão.* Um pacote que é recebido com erro no destinatário será retransmitido pelo remetente.

A Figura 3.10 mostra a representação por FSM do `rdt2.0`, um protocolo de transferência de dados que emprega detecção de erros, reconhecimentos positivos e reconhecimentos negativos.

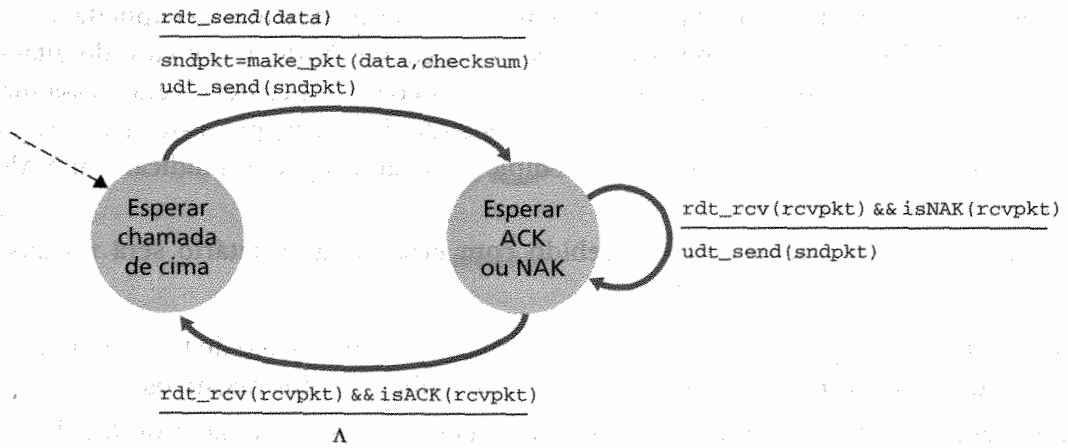
O lado remetente do `rdt2.0` tem dois estados. No estado mais à esquerda, o protocolo do lado remetente está esperando que os dados sejam passados pela camada superior. Quando o evento `rdt_send(data)` ocorrer, o remetente criará um pacote (`sndpkt`) contendo os dados a serem enviados, juntamente com uma soma de verificação do pacote (por exemplo, como discutimos na Seção 3.3.2 para o caso de um segmento UDP) e, então, enviará o pacote pela operação `udt_send(sndpkt)`. No estado mais à direita, o protocolo remetente está esperando por um pacote ACK ou NAK da parte do destinatário. Se um pacote ACK for recebido (a notação `rdt_rcv(rcvpkt) && isACK (rcvpkt)` na Figura 3.10 corresponde a esse evento), o remetente saberá que o pacote transmitido mais recentemente foi recebido corretamente. Assim, o protocolo volta ao estado de espera por dados vindos da camada superior. Se for recebido um NAK, o protocolo retransmitirá o último pacote e esperará por um ACK ou NAK a ser devolvido pelo destinatário em resposta ao pacote de dados retransmitido. É importante notar que, quando o destinatário está no estado de espera por ACK ou NAK, não pode receber mais dados da camada superior; isto é, o evento `rdt_send()` não pode ocorrer; isso somente acontecerá após o remetente receber um ACK e sair desse estado. Assim, o remetente não enviará novos dados até ter certeza de que o destinatário recebeu corretamente o pacote em questão. Devido a esse comportamento, protocolos como o `rdt2.0` são conhecidos como protocolos **pare e espere** (*stop-and-wait*).

A FSM do lado destinatário para o `rdt2.0` tem um único estado. Quando o pacote chega, o destinatário responde com um ACK ou um NAK, dependendo de o pacote recebido estar ou não corrompido. Na Figura 3.10, a notação `rdt_rcv(rcvpkt) && corrupt(rcvpkt)` corresponde ao evento em que um pacote é recebido e existe um erro.

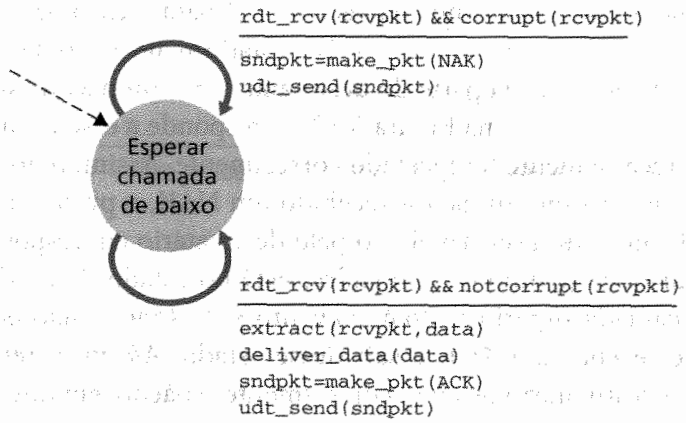
Pode parecer que o protocolo `rdt2.0` funciona, mas infelizmente ele tem um defeito fatal. Em particular, ainda não tratamos da possibilidade de o pacote ACK ou NAK estar corrompido! (Antes de continuar, é bom você começar a pensar em como esse problema pode ser resolvido.) Lamentavelmente, nossa pequena omissão não é tão inofensiva quanto possa parecer. No mínimo, precisaremos adicionar aos pacotes ACK/NAK bits de soma de verificação para detectar esses erros. A questão mais difícil é como o protocolo deve se recuperar de erros em pacotes ACK ou NAK. Nesse caso, a dificuldade é que, se um ACK ou um NAK estiver corrompido, o remetente não terá como saber se o destinatário recebeu ou não corretamente a última parcela de dados transmitidos.

Considere três possibilidades para manipular ACKs ou NAKs corrompidos:

- Para a primeira possibilidade, imagine o que um ser humano faria no cenário do ditado da mensagem. Se quem estiver ditando não entender o 'o.k.' ou o 'Repita, por favor' do destinatário, provavelmente perguntará: "O que foi que você disse?" (introduzindo assim um novo tipo de pacote remetente–destinatário em nosso protocolo). O locutor então repetiria a resposta. Mas e se a frase "O que foi que você disse?" estivesse corrompida? O destinatário, sem ter nenhuma noção se a sentença corrompida era parte do ditado ou um pedido para repetir a última resposta, provavelmente responderia: "O que foi que você disse?" E então, é claro, essa resposta também poderia estar truncada. É óbvio que estamos entrando em um caminho difícil.



a. rdt2.0: lado remetente



b. rdt2.0: lado destinatário

Figura 3.10 `rdt2.0` — Um protocolo para um canal com erros de bits

- Uma segunda alternativa é adicionar um número suficiente de bits de soma de verificação para permitir que o remetente não somente detecte, mas também se recupere de erros de bits. Isso resolve o problema imediato para um canal que pode corromper pacotes, mas não perdê-los.
- Uma terceira abordagem é o remetente simplesmente reenviar o pacote de dados corrente quando receber um pacote ACK ou NAK truncado. Esse método, no entanto, introduz pacotes duplicados no canal remetente–destinatário. A dificuldade fundamental com pacotes duplicados é que o destinatário não sabe se o último ACK ou NAK que enviou foi recebido corretamente no remetente. Assim, ele não pode saber *a priori* se um pacote que chega contém novos dados ou se é uma retransmissão!

Uma solução simples para esse novo problema (e que é adotada em quase todos os protocolos de transferência de dados existentes, inclusive o TCP) é adicionar um novo campo ao pacote de dados e fazer com que o remetente numere seus pacotes de dados colocando um número de seqüência nesse campo. O destinatário então teria apenas de verificar esse número de seqüência para determinar se o pacote recebido é ou não uma retransmissão. Para esse caso simples de protocolo pare e espere, um número de seqüência de um bit é suficiente, já que permitirá que o destinatário saiba se o remetente está reenviando o pacote previamente transmitido (o número de seqüência do pacote recebido é o mesmo do pacote recebido mais recentemente) ou um novo pacote (o número de seqüência muda, indo "para a frente" em progressão aritmética de módulo 2). Como estamos admitindo que este é um canal que não perde pacotes, os pacotes ACK e NAK em si não precisam indicar o número de seqüência do pacote que estão reconhecendo. O remetente sabe que um pacote ACK ou NAK recebido (truncado ou não) foi gerado em resposta ao seu pacote de dados transmitidos mais recentemente.

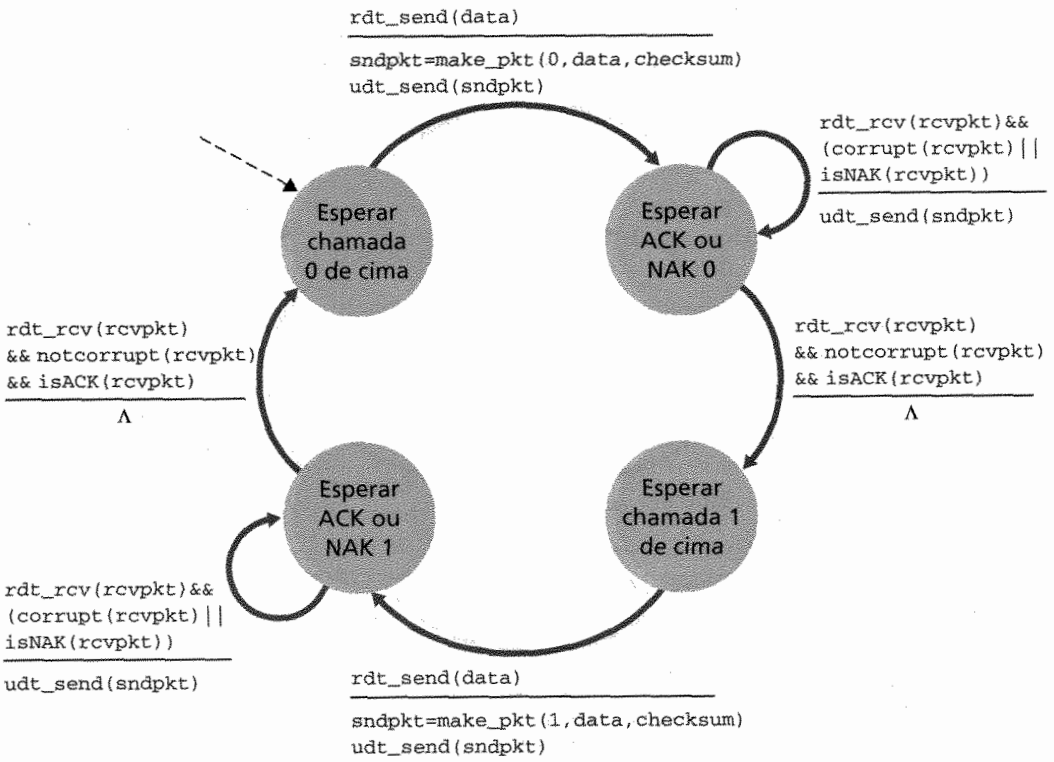


Figura 3.11 rdt2.1 remetente

As figuras 3.11 e 3.12 mostram a descrição da FSM para o rdt2.1, nossa versão corrigida do rdt2.0. Cada um dos rdt2.1 remetente e destinatário da FSM agora tem um número duas vezes maior de estados do que antes. Isso acontece porque o estado do protocolo deve agora refletir se o pacote que está sendo correntemente enviado (pelo remetente) ou aguardado (no destinatário) deveria ter um número de seqüência 0 ou 1. Note que as ações nos estados em que um pacote numerado com 0 está sendo enviado ou aguardado são imagens especulares daquelas que devem funcionar quando estiver sendo enviado ou aguardado um pacote numerado com 1; as únicas diferenças têm a ver com a manipulação do número de seqüência.

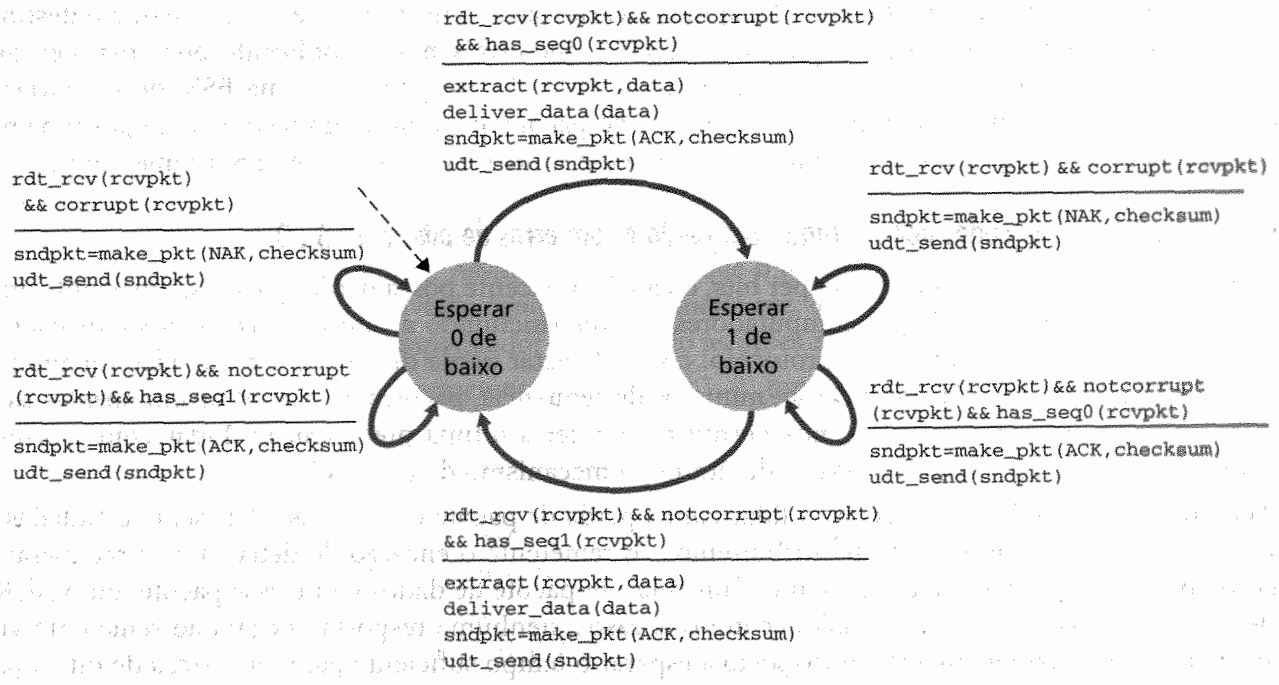


Figura 3.12 rdt2.1 destinatário

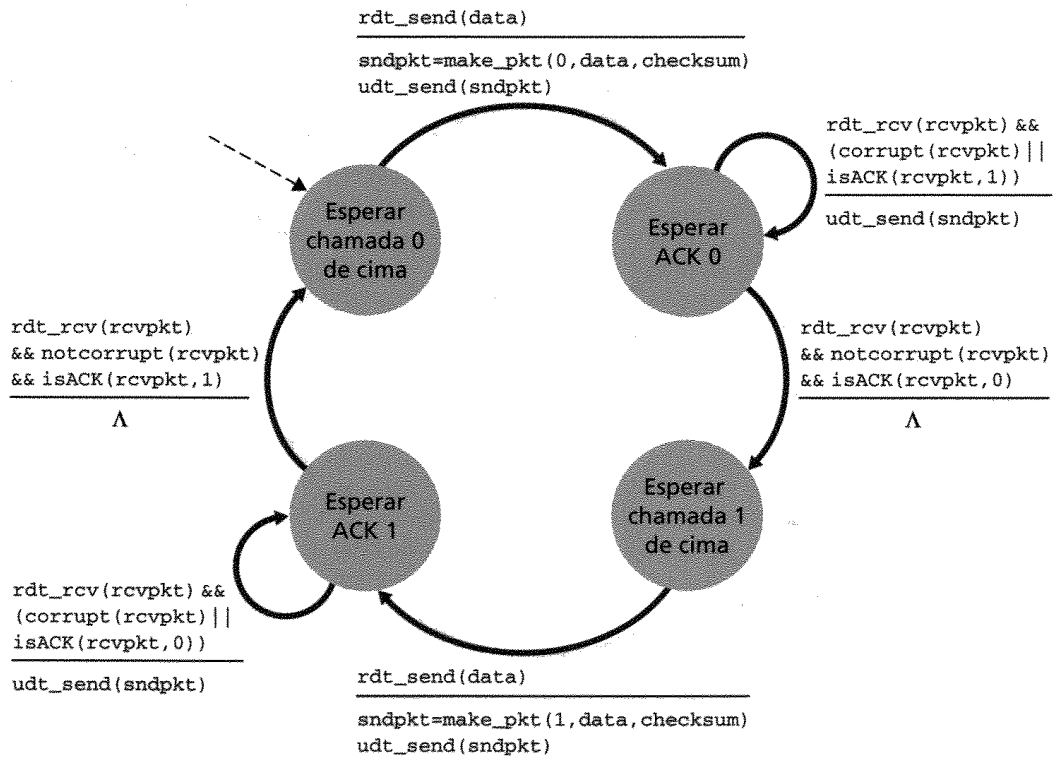


Figura 3.13 `rdt2.2` remetente

O protocolo `rdt2.1` usa tanto o reconhecimento positivo como o negativo do remetente ao destinatário. Quando um pacote fora de ordem é recebido, o destinatário envia um reconhecimento positivo para o pacote que recebeu; quando um pacote corrompido é recebido, ele envia um reconhecimento negativo. Podemos conseguir o mesmo efeito de um pacote NAK se, em vez de enviarmos um NAK, enviarmos um ACK em seu lugar para o último pacote corretamente recebido. Um remetente que recebe dois ACKs para o mesmo pacote (isto é, **ACKs duplicados**) sabe que o destinatário não recebeu corretamente o pacote seguinte àquele para o qual estão sendo dados dois ACKs.

Nosso protocolo de transferência confiável de dados sem NAK para um canal com erros de bits é o `rdt2.2`, mostrado nas figuras 3.13 e 3.14. Uma modificação sutil entre `rdt2.1` e `rdt2.2` é que o destinatário agora deve incluir o número de seqüência do pacote que está sendo reconhecido por uma mensagem ACK (o que é feito incluindo o argumento ACK, 0 ou ACK, 1 em `make_pkt()` na FSM destinatária) e o remetente agora deve verificar o número de seqüência do pacote que está sendo reconhecido por uma mensagem ACK recebida (o que é feito incluindo o argumento 0 ou 1 em `isACK()` na FSM remetente).

Transferência confiável de dados por um canal com perda e com erros de bits: `rdt3.0`

Suponha agora que, além de corromper bits, o canal subjacente possa perder pacotes, um acontecimento que não é incomum nas redes de computadores de hoje (incluindo a Internet). Duas preocupações adicionais devem agora ser tratadas pelo protocolo: como detectar perda de pacote e o que fazer quando isso ocorre. A utilização de soma de verificação, números de seqüência, pacotes ACK e retransmissões — as técnicas já desenvolvidas em `rdt2.2` — nos permitirá atender a última preocupação. Lidar com a primeira preocupação, por sua vez, exigirá a adição de um novo mecanismo de protocolo.

Há muitas abordagens possíveis para lidar com a perda de pacote (e diversas delas serão estudadas nos exercícios ao final do capítulo). Aqui, atribuiremos ao remetente o encargo de detectar e se recuperar das perdas de pacote. Suponha que o remetente transmita um pacote de dados e que esse pacote, ou o ACK do seu destinatário, seja perdido. Em qualquer um dos casos, nenhuma resposta chegará ao remetente vinda do destinatário. Se o remetente estiver disposto a esperar o tempo suficiente para ter *certeza* de que o pacote foi perdido, ele poderá simplesmente retransmitir o pacote de dados. É preciso que você se convença de que esse protocolo funciona mesmo.

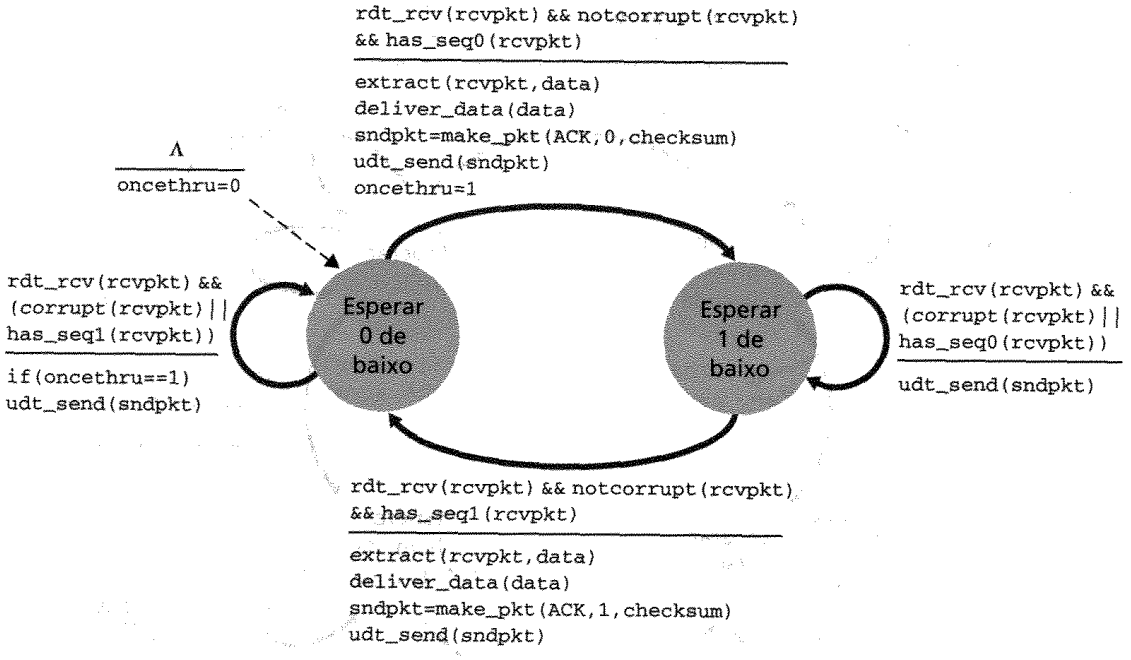


Figura 3.14 rdt2.2 destinatário

Mas quanto tempo o remetente precisa esperar para ter certeza de que algo foi perdido? É claro que deve aguardar no mínimo o tempo de um atraso de ida e volta entre ele e o destinatário (o que pode incluir buffers em roteadores ou equipamentos intermediários) e mais o tempo que for necessário para processar um pacote no destinatário. Em muitas redes, o atraso máximo para esses piores casos é muito difícil até de estimar, quanto mais saber com certeza. Além disso, o ideal seria que o protocolo se recuperasse da perda de pacotes logo que possível; esperar pelo atraso do pior dos casos pode significar um longo tempo até que a recuperação do erro seja iniciada. Assim, a abordagem adotada na prática é a seguinte: o remetente faz uma escolha ponderada de um valor de tempo dentro do qual seria provável, mas não garantido, que a perda tivesse acontecido. Se não for recebido um ACK nesse período, o pacote é retransmitido. Note que, se um pacote sofrer um atraso particularmente longo, o remetente poderá retransmiti-lo mesmo que, nem o pacote de dados, nem o seu ACK tenham sido perdidos. Isso introduz a possibilidade de **pacotes de dados duplicados** no canal remetente–destinatário. Felizmente, o protocolo rdt2.2 já dispõe de funcionalidade suficiente (isto é, números de seqüência) para tratar dos casos de pacotes duplicados.

Do ponto de vista do remetente, a retransmissão é uma panacéia. O remetente não sabe se um pacote de dados foi perdido, se um ACK foi perdido ou se o pacote ou o ACK simplesmente estavam muito atrasados. Em todos os casos, a ação é a mesma: retransmitir. Para implementar um mecanismo de retransmissão com base no tempo, é necessário um **temporizador de contagem regressiva** que interrompa o processo remetente após ter decorrido um dado tempo. Assim, será preciso que o remetente possa (1) acionar o temporizador todas as vezes que um pacote for enviado (quer seja a primeira vez, quer seja uma retransmissão), (2) responder a uma interrupção feita pelo temporizador (realizando as ações necessárias) e (3) parar o temporizador.

A Figura 3.15 mostra a FSM remetente para o rdt3.0, um protocolo que transfere confiavelmente dados por um canal que pode corromper ou perder pacotes; nos "Exercícios de fixação" pediremos a você que estabeleça a FSM destinatária para rdt3.0. A Figura 3.16 mostra como o protocolo funciona sem pacotes perdidos ou atrasados e como manipula pacotes de dados perdidos. Nessa figura, a passagem do tempo ocorre do topo do diagrama para baixo. Note que o instante de recebimento de um pacote é necessariamente posterior ao instante de envio de um pacote, como resultado de atrasos de transmissão e de propagação. Nas figuras 3.16(b-d), os colchetes do lado remetente indicam os instantes em que o temporizador foi acionado e, mais tarde, os instantes em que ele parou. Vários dos aspectos mais sutis desse protocolo são examinados nos exercícios ao final deste capítulo. Como os números de seqüência se alternam entre 0 e 1, o protocolo rdt3.0 às vezes é conhecido como **protocolo bit alternante**.

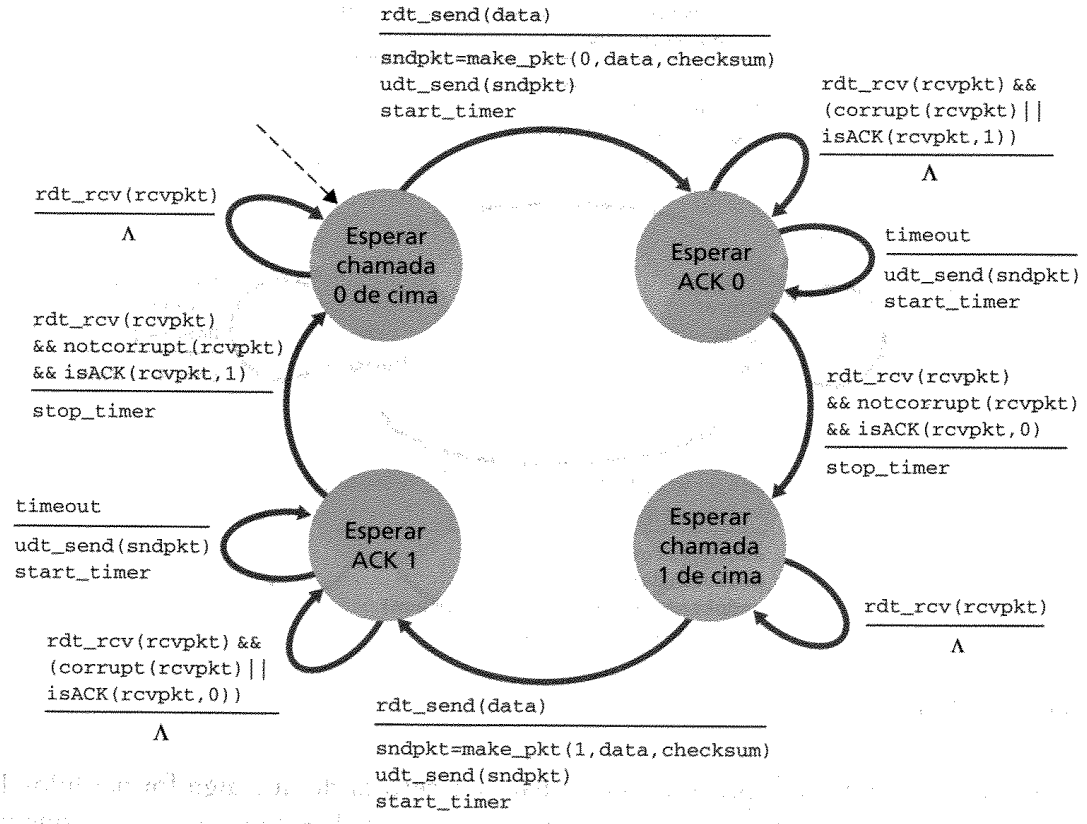


Figura 3.15 `rdt3.0` remetente

Agora já reunimos os elementos fundamentais de um protocolo de transferência de dados. Somas de verificação, números de seqüência, temporizadores e pacotes de reconhecimento negativo e positivo — cada um desempenha um papel crucial e necessário na operação do protocolo. Temos agora em funcionamento um protocolo de transferência confiável de dados!

3.4.2 Protocolos de transferência confiável de dados com paralelismo

O protocolo `rdt3.0` é correto em termos funcionais, mas é pouco provável que alguém fique contente com o desempenho dele, particularmente nas redes de alta velocidade de hoje. No coração do problema do desempenho do `rdt3.0` está o fato de ele ser um protocolo do tipo pare e espere.

Para avaliar o impacto sobre o desempenho causado pelo comportamento pare e espere, considere um caso ideal de dois hospedeiros, um localizado na Costa Oeste dos Estados Unidos e outro na Costa Leste, como mostra a Figura 3.17.

O atraso de propagação de ida e volta à velocidade da luz, T_{prop}, entre esses dois sistemas finais é de aproximadamente 30 milissegundos. Suponha que eles estejam conectados por um canal com capacidade de transmissão, R, de 1 gigabit (10^9 bits) por segundo. Para um tamanho de pacote, L, de 1 kbyte (8 mil bits) por pacote, incluindo o campo de cabeçalho e também o de dados, o tempo necessário para realmente transmitir o pacote para o enlace de 1 Gbps é:

$$t_{trans} = \frac{L}{R} = \frac{8 \text{ kbits/pacote}}{10^9 \text{ bits/seg}} = 8 \text{ microssegundos}$$

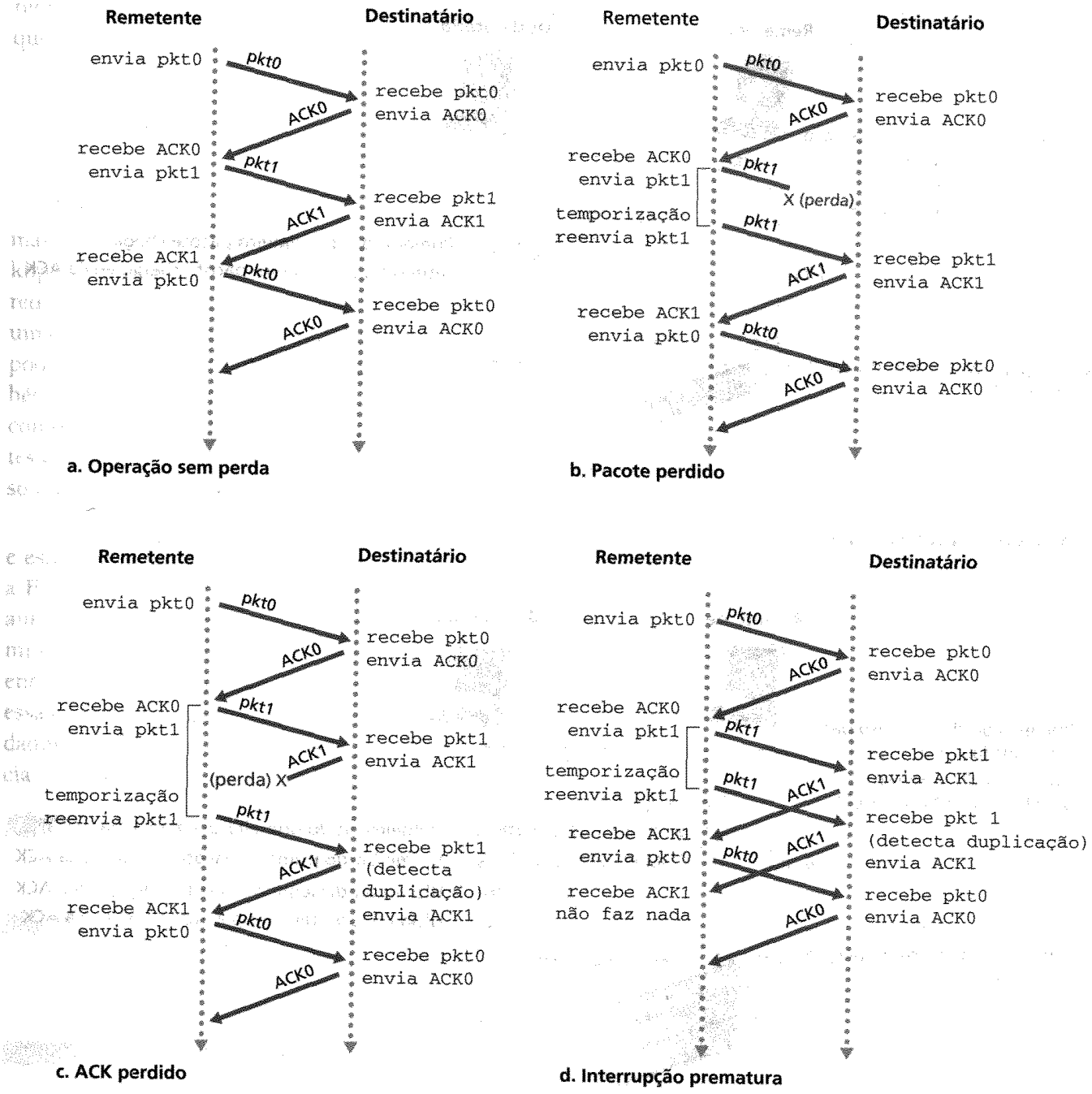
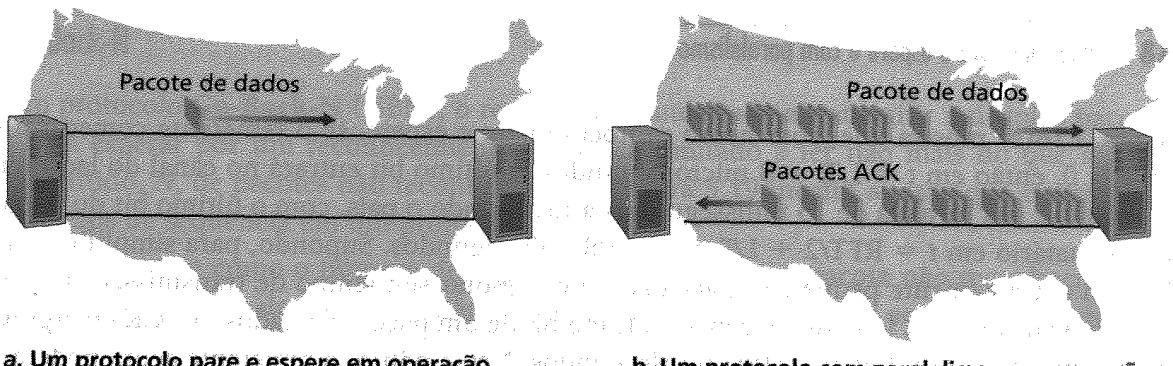


Figura 3.16 Operação do `rdt3.0`, o protocolo bit alternante



a. Um protocolo pare e espere em operação

b. Um protocolo com paralelismo em operação

Figura 3.17 Protocolo pare e espere *versus* protocolo com paralelismo

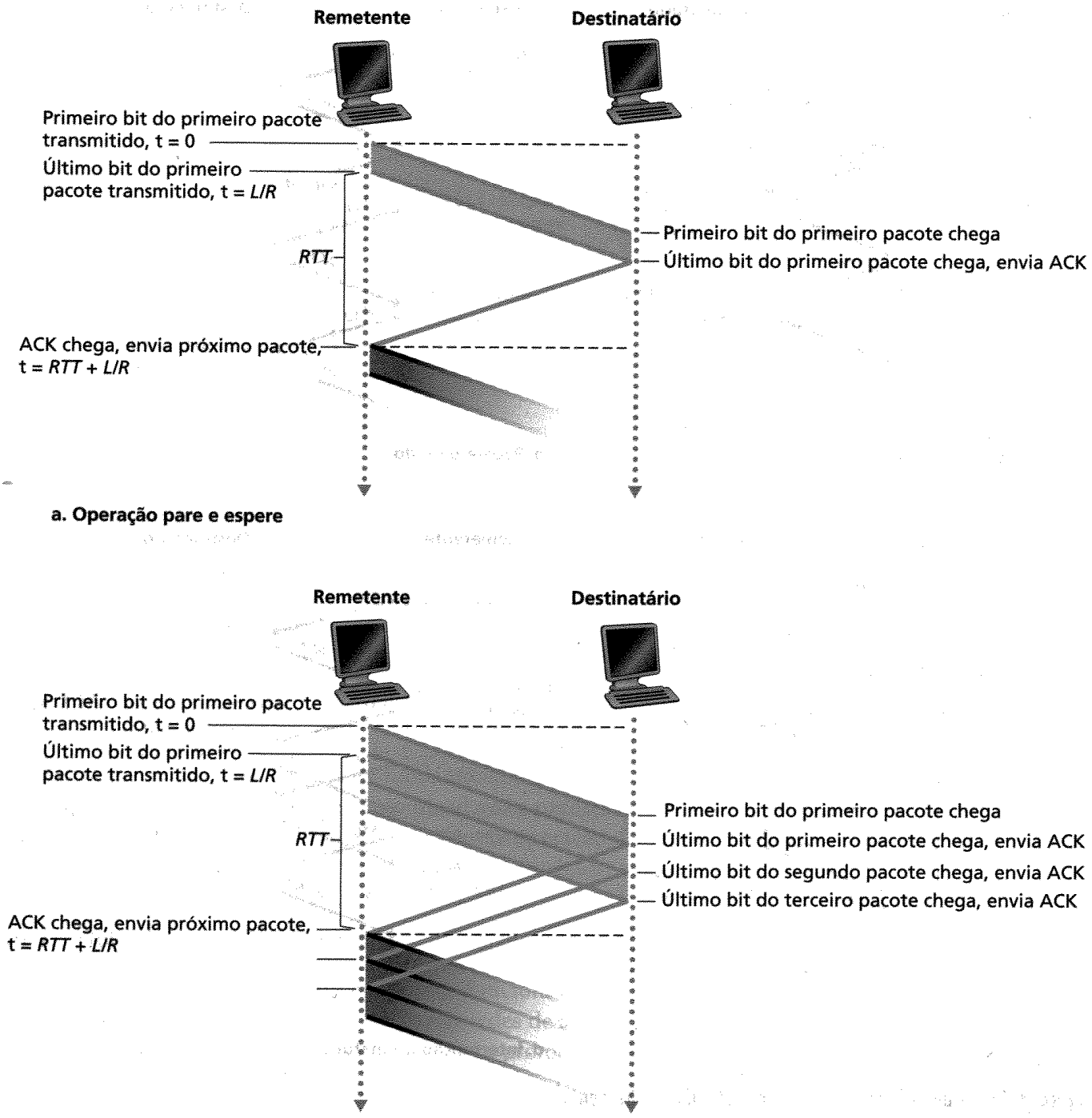


Figura 3.18 Envio com pare e espere e com paralelismo

A Figura 3.18(a) mostra que, com nosso protocolo pare e espere, se o remetente começar a enviar o pacote em $t = 0$, então em $t = L/R = 8$ microssegundos, o último bit entrará no canal do lado remetente. O pacote então faz sua jornada de 15 milissegundos atravessando o país, com o último bit do pacote emergindo no destinatário em $t = RTT/2 + L/R = 15{,}008$ milissegundos. Supondo, para simplificar, que pacotes ACK sejam extremamente pequenos (para podermos ignorar seu tempo de transmissão) e que o destinatário pode enviar um ACK logo que receber o último bit de um pacote de dados, o ACK emergirá de volta no remetente em $t = RTT + L/R = 30{,}008$ milissegundos. Nesse ponto, o remetente agora poderá transmitir a próxima mensagem. Assim, em 30,008 milissegundos, o remetente esteve enviando por apenas 0,008 milissegundo. Se definirmos a **utilização** do remetente (ou do canal) como a fração de tempo em que o

remetente está realmente ocupado enviando bits para dentro do canal, a análise da Figura 3.18(a) mostra que o protocolo pare e espere tem uma utilização do remetente U_{remet} bastante desanimadora, de:

$$U_{remet} = \frac{L/R}{RTT + L/R} = \frac{0{,}008}{30{,}008} = 0{,}00027$$

Portanto, o remetente ficou ocupado apenas 2,7 centésimos de 1 por cento do tempo! Visto de outra maneira, ele só foi capaz de enviar 1.000 bytes em 30,008 milissegundos, uma vazão efetiva de apenas 267 kbps — mesmo estando disponível um enlace de 1 gigabit por segundo! Imagine o infeliz gerenciador de rede que acabou de pagar uma fortuna para ter capacidade de enlace da ordem de gigabits, mas consegue uma vazão de apenas 267 quilobits por segundo! Este é um exemplo gráfico de como protocolos de rede podem limitar as capacidades oferecidas pelo hardware subjacente de rede. Além disso, desprezamos também os tempos de processamento de protocolo das camadas inferiores no remetente e no destinatário, bem como os atrasos de processamento e de fila que ocorreriam em quaisquer roteadores intermediários existentes entre o remetente e o destinatário. Incluir esses efeitos serviria apenas para aumentar ainda mais o atraso e piorar ainda mais o fraco desempenho.

A solução para esse problema de desempenho em particular é simples: em vez de operar em modo pare e espere, o remetente é autorizado a enviar vários pacotes sem esperar por reconhecimentos, como mostra a Figura 3.17(b). A Figura 3.18(b) mostra que, se um remetente for autorizado a transmitir três pacotes antes de ter de esperar por reconhecimentos, sua utilização será essencialmente triplicada. Uma vez que os muitos pacotes em trânsito entre remetente e destinatário podem ser visualizados como se estivessem enchendo uma tubulação, essa técnica é conhecida, em inglês, como ***pipelining*** (tubulação). Porém, como essa expressão é difícil de traduzir para o português, preferimos usar 'paralelismo', embora a transmissão de dados seja de fato seqüencial. O paralelismo gera as seguintes conseqüências para protocolos de transferência confiável de dados:

- A faixa de números de seqüência tem de ser ampliada, uma vez que cada pacote em trânsito (sem contar as retransmissões) precisa ter um número de seqüência exclusivo e pode haver vários pacotes não reconhecidos em trânsito.
- Os lados remetente e destinatário dos protocolos podem ter de reservar buffers para mais de um pacote. No mínimo, o remetente terá de providenciar buffers para pacotes que foram transmitidos, mas que ainda não foram reconhecidos. O buffer de pacotes corretamente recebidos pode também ser necessário no destinatário, como discutiremos a seguir.
- A faixa de números de seqüência necessária e as necessidades de buffer dependerão da maneira como um protocolo de transferência de dados responde a pacotes perdidos, corrompidos e demasiadamente atrasados. Duas abordagens básicas em relação à recuperação de erros com paralelismo podem ser identificadas: **Go-Back-N** e **repetição seletiva.**

3.4.3 Go-Back-N

Em um protocolo **Go-Back-N** (GBN), o remetente é autorizado a transmitir múltiplos pacotes (se disponíveis) sem esperar por um reconhecimento, mas fica limitado a ter não mais do que algum número máximo permitido, *N*, de pacotes não reconhecidos. A Figura 3.19 mostra a visão que o remetente tem da faixa de números de seqüência em um protocolo GBN.

Se definirmos `base` como o número de seqüência do mais antigo pacote não reconhecido e `nextseqnum` como o menor número de seqüência não utilizado (isto é, o número de seqüência do próximo pacote a ser enviado), então quatro intervalos na faixa de números de seqüência poderão ser identificados. Os números de seqüência no intervalo `[0,base-1]` correspondem aos pacotes que já foram transmitidos e reconhecidos. O intervalo `[base,nextseqnum-1]` corresponde aos pacotes que foram enviados, mas ainda

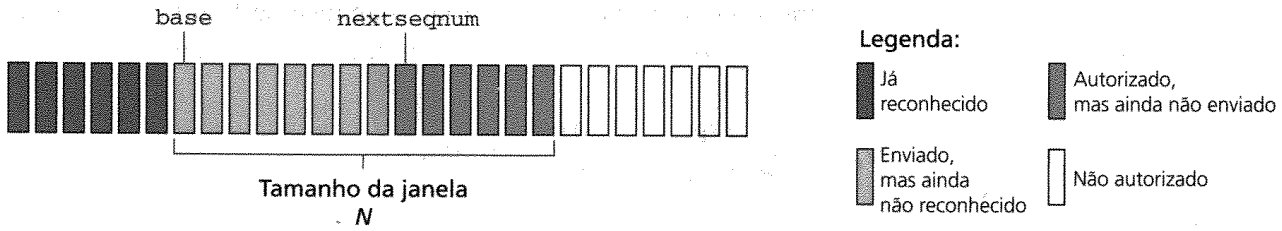


Figura 3.19 Visão do remetente para os números de seqüência no protocolo Go-Back-N

não foram reconhecidos. Os números de seqüência no intervalo `[nextseqnum,base+N-1]` podem ser usados para pacotes que podem ser enviados imediatamente, caso cheguem dados vindos da camada superior. Finalmente, números de seqüência maiores ou iguais a `base+N` não podem ser usados até que um pacote não reconhecido que esteja pendente seja reconhecido (especificamente, o pacote cujo número de seqüência é `base`).

Como sugere a Figura 3.19, a faixa de números de seqüência permitidos para pacotes transmitidos mas ainda não reconhecidos pode ser vista como uma 'janela' de *tamanho N* sobre a faixa de números de seqüência. À medida que o protocolo opera, essa janela se desloca para a frente sobre o espaço de números de seqüência. Por essa razão, *N* é freqüentemente denominado **tamanho de janela** e o protocolo GBN em si, **protocolo de janela deslizante** (*sliding-window protocol*). É possível que você esteja pensando que razão teríamos, em primeiro lugar, para limitar o número de pacotes pendentes não reconhecidos a um valor N. Por que não permitir um número ilimitado desses pacotes? Veremos na Seção 3.5 que o controle de fluxo é uma das razões para impor um limite ao remetente. Examinaremos outra razão para isso na Seção 3.7, quando estudarmos o controle de congestionamento do TCP.

Na prática, o número de seqüência de um pacote é carregado em um campo de comprimento fixo no cabeçalho do pacote. Se k for o número de bits no campo de número de seqüência do pacote, a faixa de números de seqüência será então $[0,2^k - 1]$. Com uma faixa finita de números de seqüência, toda a aritmética que envolver números de seqüência deverá ser feita usando aritmética de módulo 2^k. (Em outras palavras, o espaço do número de seqüência pode ser imaginado como um anel de tamanho 2^k, em que o número de seqüência $2^k - 1$ é imediatamente seguido pelo número de seqüência 0.) Lembre-se de que `rdt3.0` tem um número de seqüência de 1 bit e uma faixa de números de seqüência de [0,1]. Vários problemas ao final deste capítulo tratam das conseqüências de uma faixa finita de números de seqüência. Veremos na Seção 3.5 que o TCP tem um campo de número de seqüência de 32 bits, onde os números de seqüência do TCP contam bytes na cadeia de bytes, em vez de pacotes.

As figuras 3.20 e 3.21 descrevem uma FSM estendida dos lados remetente e destinatário de um protocolo GBN baseado em ACK, mas sem NAK.

Referimo-nos a essa descrição de FSM como *FSM estendida* porque adicionamos variáveis (semelhantes às variáveis de linguagem de programação) para `base` e `nextseqnum`; também adicionamos operações sobre essas variáveis e ações condicionais que as envolvem. Note que a especificação da FSM estendida agora está começando a parecer um pouco com uma especificação de linguagem de programação. [Bochman, 1984] fornece um excelente levantamento sobre extensões adicionais às técnicas FSM, bem como sobre outras técnicas para especificação de protocolos baseadas em linguagens.

O remetente GBN deve responder a três tipos de eventos:

- *Chamada vinda de cima.* Quando `rdt_send()` é chamado de cima, o remetente primeiramente verifica se a janela está cheia, isto é, se há N pacotes pendentes não reconhecidos. Se a janela não estiver cheia, um pacote é criado e enviado e as variáveis são adequadamente atualizadas. Se estiver cheia, o remetente apenas devolve os dados à camada superior — uma indicação implícita de que a janela está cheia. Presumivelmente, a camada superior então teria de tentar outra vez mais tarde. Em uma implementação real, o remetente muito provavelmente teria colocado esses dados

em um buffer (mas não os teria enviado imediatamente) ou teria um mecanismo de sincronização (por exemplo, um semáforo ou uma flag) que permitiria que a camada superior chamasse `rdt_send()` somente quando as janelas não estivessem cheias.

- *Recebimento de um ACK.* Em nosso protocolo GBN, um reconhecimento de pacote com número de seqüência *n* seria tomado como um **reconhecimento cumulativo**, indicando que todos os pacotes com número de seqüência até e inclusive *n* tinham sido corretamente recebidos no destinatário. Voltaremos a esse assunto em breve, quando examinarmos o lado destinatário do GBN.
- *Um esgotamento de temporização.* O nome 'Go-Back-N' deriva do comportamento do remetente em relação a pacotes perdidos ou demasiadamente atrasados. Como no protocolo pare e espere, um temporizador é usado para recuperar a perda de dados ou reconhecer pacotes. Se ocorrer o esgotamento da temporização, o remetente reenvia *todos* os pacotes que tinham sido previamente enviados mas que ainda não tinham sido reconhecidos. Nosso remetente da Figura 3.20 usa apenas um único temporizador, que pode ser imaginado como um temporizador para o mais antigo pacote já transmitido mas que ainda não foi reconhecido. Se for recebido um ACK e ainda houver pacotes adicionais transmitidos mas ainda não reconhecidos, o temporizador será reiniciado. Se não houver nenhum pacote pendente não reconhecido, o temporizador será desligado.

As ações do destinatário no GBN também são simples. Se um pacote com número de seqüência *n* for recebido corretamente e estiver na ordem (isto é, os últimos dados entregues à camada superior vierem de um pacote com número de seqüência $n - 1$), o destinatário enviará um ACK para o pacote *n* e entregará a porção dos dados do pacote à camada superior. Em todos os outros casos, o destinatário descarta o pacote e reenvia um ACK para o pacote mais recente que foi recebido na ordem correta. Dado que os pacotes são entregues à camada superior um por vez, se o pacote *k* tiver sido recebido e entregue, então todos os pacotes com número de seqüência menores do que *k* também terão sido entregues. Assim, o uso de reconhecimentos cumulativos é uma escolha natural para o GBN.

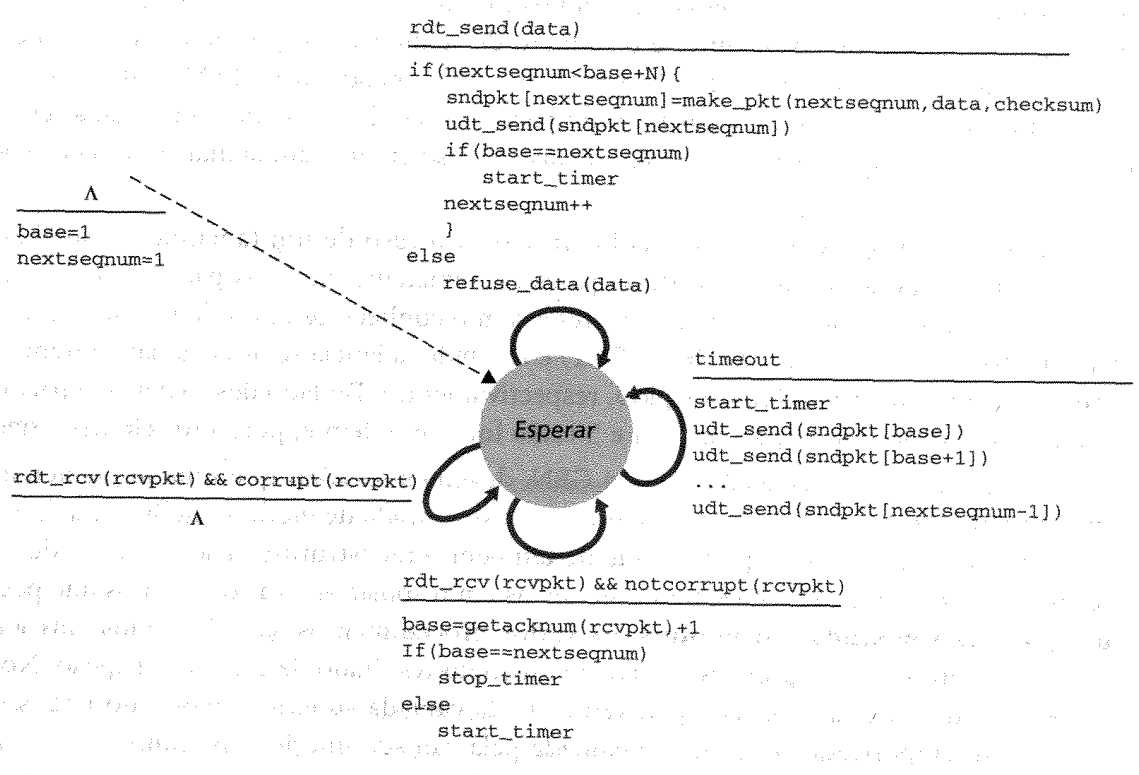


Figura 3.20 Descrição da FSM estendida do remetente GBN

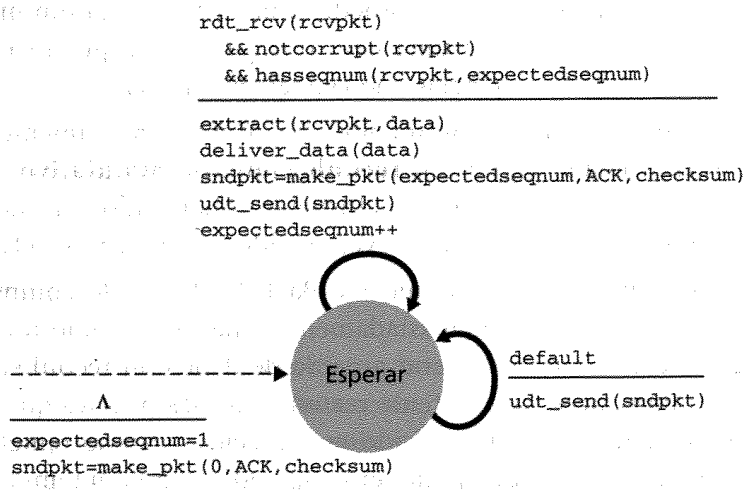


Figura 3.21 Descrição da FSM estendida do destinatário GBN

Em nosso protocolo GBN, o destinatário descarta os pacotes que chegam fora de ordem. Embora pareça bobagem e perda de tempo descartar um pacote corretamente recebido (mas fora de ordem), existem justificativas para isso. Lembre-se de que o destinatário deve entregar dados na ordem certa à camada superior. Suponha agora que o pacote n esteja sendo esperado, mas quem chega é o pacote $n + 1$. Como os dados devem ser entregues na ordem certa, o destinatário *poderia* conservar o pacote $n + 1$ no buffer (salvá-lo) e entregar esse pacote à camada superior mais tarde, após ter recebido o pacote n. Contudo, se o pacote n for perdido, os pacotes n e $n + 1$ serão ambos eventualmente retransmitidos como resultado da regra de retransmissão do GBN no remetente. Assim, o destinatário pode simplesmente descartar o pacote $n + 1$. A vantagem dessa abordagem é a simplicidade da manipulação de buffers no destinatário — o destinatário não precisa colocar no buffer nenhum pacote que esteja fora de ordem. Desse modo, enquanto o remetente deve manter os limites superior e inferior de sua janela e a posição de `nextseqnum` dentro dessa janela, a única informação que o destinatário precisa manter é o número de seqüência do próximo pacote esperado conforme a ordem. Esse valor é retido na variável `expectedseqnum` mostrada na FSM destinatária da Figura 3.21. Evidentemente, a desvantagem de jogar fora um pacote recebido corretamente é que a retransmissão subseqüente desse pacote pode ser perdida ou ficar truncada, caso em que ainda mais retransmissões seriam necessárias.

A Figura 3.22 mostra a operação do protocolo GBN para o caso de um tamanho de janela de quatro pacotes. Por causa da limitação do tamanho dessa janela, o remetente envia os pacotes de 0 a 3, mas, em seguida, tem de esperar que um ou mais desses pacotes sejam reconhecidos antes de prosseguir. E, à medida que cada ACK sucessivo (por exemplo, ACK0 e ACK1) é recebido, a janela se desloca para a frente e o remetente pode transmitir um novo pacote (pkt4 e pkt5, respectivamente). Do lado destinatário, o pacote 2 é perdido. Desse modo, verifica-se que os pacotes 3, 4 e 5 estão fora de ordem e, portanto, são descartados.

Antes de encerrarmos nossa discussão sobre o GBN, devemos ressaltar que uma implementação desse protocolo em uma pilha de protocolo provavelmente seria estruturada de modo semelhante à da FSM estendida da Figura 3.20. A implementação provavelmente também seria estruturada sob a forma de vários procedimentos que implementam as ações a serem executadas em resposta aos vários eventos que podem ocorrer. Nessa **programação baseada em eventos**, os vários procedimentos são chamados (invocados) por outros procedimentos presentes na pilha de protocolo ou como resultado de uma interrupção. No remetente, esses eventos seriam: (1) uma chamada pela entidade da camada superior invocando `rdt_send()`, (2) uma interrupção pelo temporizador e (3) uma chamada pela camada inferior invocando `rdt_rcv()` quando chega um pacote. Os exercícios de programação ao final deste capítulo lhe darão a chance de implementar de verdade essas rotinas em um ambiente de rede simulado, mas realista.

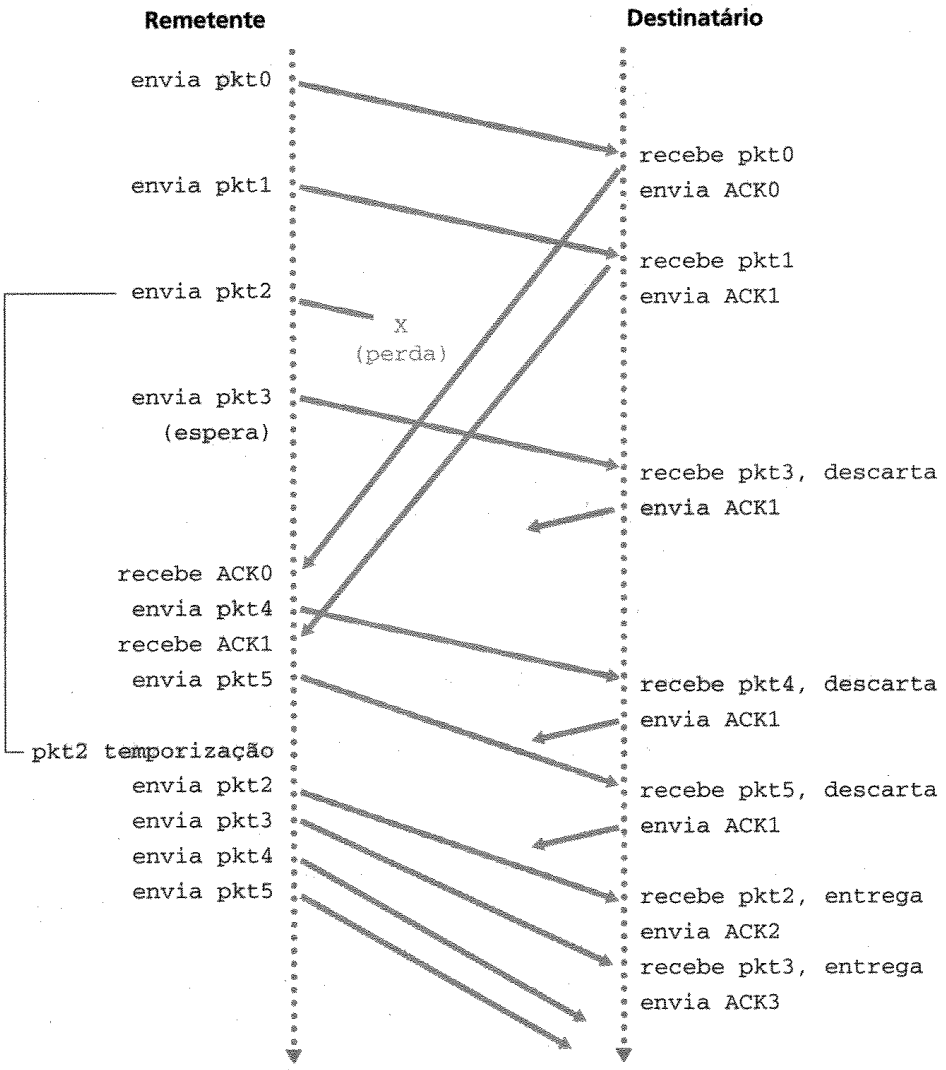


Figura 3.22 GBN em operação

Salientamos que o protocolo GBN incorpora quase todas as técnicas que encontraremos quando estudarmos, na Seção 3.5, os componentes de transferência confiável de dados do TCP. Essas técnicas incluem a utilização de números de seqüência, reconhecimentos cumulativos, somas de verificação e uma operação de esgotamento de temporização/retransmissão.

3.4.4 Repetição seletiva

O protocolo GBN permite que o remetente potencialmente "encha a rede" com pacotes na Figura 3.17, evitando, assim, os problemas de utilização de canal observados em protocolos do tipo pare e espere. Há, contudo, casos em que o próprio GBN sofre com problemas de desempenho. Em particular, quando o tamanho da janela e o produto entre o atraso e a largura de banda são grandes, pode haver muitos pacotes pendentes na rede. Assim, um único erro de pacote pode fazer com que o GBN retransmita um grande número de pacotes — muitos deles desnecessariamente. À medida que aumenta a probabilidade de erros no canal, a rede pode ficar lotada com essas retransmissões desnecessárias. Imagine se, em uma conversa, toda vez que uma palavra fosse pronunciada de maneira truncada as outras mil que a circundam (por exemplo, um tamanho de janela de mil palavras) tivessem de ser repetidas. A conversa sofreria atrasos devido a todas essas palavras reiteradas.

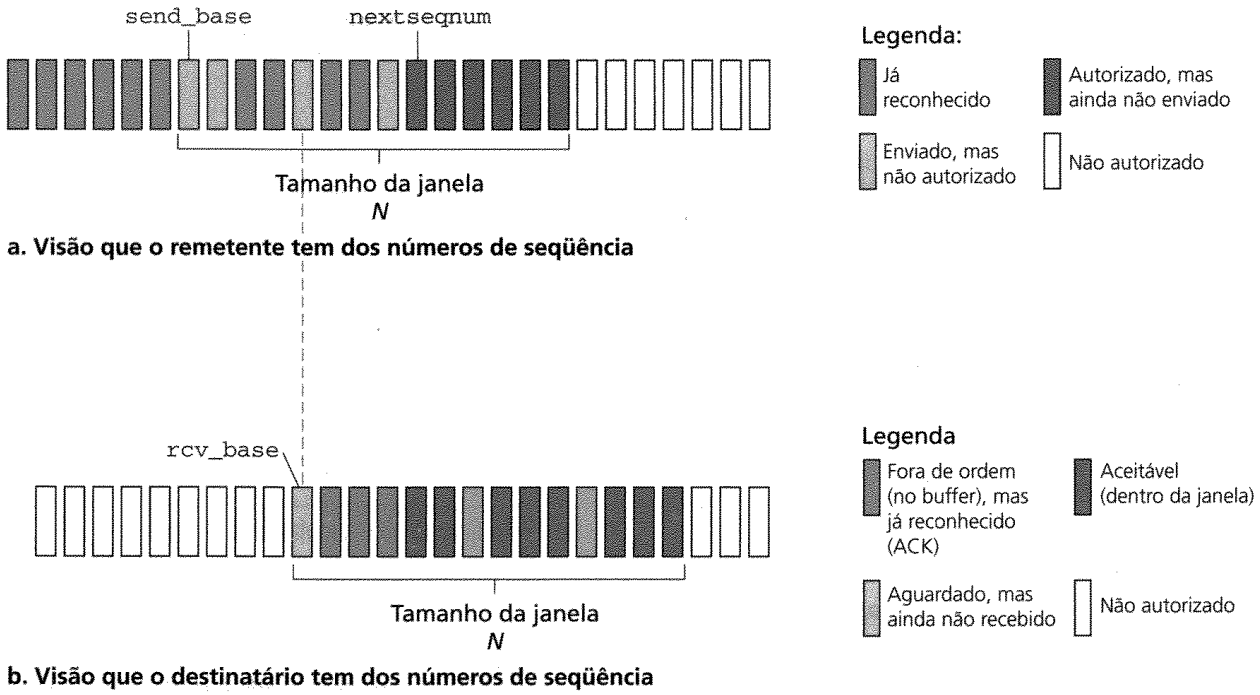


Figura 3.23 Visões que os protocolos SR remetente e destinatário têm do espaço de número de seqüência

Como o próprio nome sugere, protocolos de repetição seletiva (*selective repeat* — SR) evitam retransmissões desnecessárias porque fazem o remetente retransmitir somente os pacotes suspeitos de terem sido recebidos com erro (isto é, que foram perdidos ou corrompidos) no destinatário. Essa retransmissão individual, somente quando necessária, exige que o destinatário reconheça *individualmente* os pacotes recebidos de modo correto. Uma janela de tamanho *N* será usada novamente para limitar o número de pacotes pendentes não reconhecidos dentro da rede. Contudo, ao contrário do GBN, o remetente já terá recebido ACKs para alguns dos pacotes na janela. A Figura 3.23 mostra a visão que o protocolo de SR remetente tem do espaço do número de seqüência. A Figura 3.24 detalha as várias ações executadas pelo protocolo SR remetente.

1. *Dados recebidos de cima.* Quando são recebidos dados de cima, o protocolo SR remetente verifica o próximo número de seqüência disponível para o pacote. Se o número de seqüência está dentro da janela do remetente, os dados são empacotados e enviados; do contrário, eles são armazenados ou devolvidos à camada superior para transmissão posterior, como acontece no GBN.
2. *Esgotamento de temporização.* Novamente são usados temporizadores para proteção contra perda de pacotes. Contudo, cada pacote agora deve ter seu próprio temporizador lógico, já que apenas um pacote será transmitido quando a temporização se esgotar. Um único hardware de temporizador pode ser usado para emular a operação de múltiplos temporizadores lógicos [Varghese, 1997].
3. ACK recebido. Se for recebido um ACK, o SR remetente marcará aquele pacote como recebido, contanto que esteja na janela. Se o número de seqüência do pacote for igual a send_base, a base da janela se deslocará para a frente até o pacote não reconhecido que tiver o menor número de seqüência. Se a janela se deslocar e houver pacotes não transmitidos com números de seqüência que agora caem dentro da janela, esses pacotes serão transmitidos.

Figura 3.24 Eventos e ações do protocolo SR remetente

1. *Pacote com número de seqüência no intervalo* `[rcv_base, rcv_base+N-1]` *foi corretamente recebido.* Nesse caso, o pacote recebido cai dentro da janela do destinatário e um pacote ACK seletivo é devolvido ao remetente. Se o pacote não tiver sido recebido anteriormente, irá para o buffer. Se esse pacote tiver um número de seqüência igual à base da janela destinatária (`rcv_base` na Figura 3.22), então ele, e quaisquer outros pacotes anteriormente armazenados no buffer e numerados consecutivamente (começando com `rcv_base`), serão entregues à camada superior. A janela destinatária é então deslocada para a frente de acordo com o número de pacotes entregues à camada superior. Como exemplo, considere a Figura 3.26. Quando um pacote com número de seqüência `rcv_base=2` é recebido, ele e os pacotes 3, 4 e 5 podem ser entregues à camada superior.
2. *Pacote com número de seqüência em* `[rcv_base-N, rcv_base-1]` *é recebido.* Nesse caso, um ACK deve ser gerado mesmo que esse pacote já tenha sido reconhecido anteriormente pelo destinatário.
3. *Qualquer outro.* Ignore o pacote.

Figura 3.25 Eventos e ações do protocolo de SR destinatário

O protocolo SR destinatário reconhecerá um pacote corretamente recebido esteja ele ou não na ordem certa. Pacotes fora de ordem ficam no buffer até que todos os pacotes faltantes (isto é, os que têm números de seqüência menores) sejam recebidos, quando então um conjunto de pacotes poderá ser entregue à camada superior na ordem correta. A Figura 3.25 apresenta as várias ações realizadas pelo protocolo SR destinatário.

A Figura 3.26 apresenta um exemplo de operação do protocolo SR quando ocorre perda de pacotes. Note que, nessa figura, o destinatário inicialmente armazena os pacotes 3, 4 e 5 e os entrega juntamente com o pacote 2 à camada superior, quando o pacote 2 é finalmente recebido.

É importante notar que na etapa 2 da Figura 3.25 o destinatário reconhece novamente (em vez de ignorar) pacotes já recebidos com certos números de seqüência que estão *abaixo* da atual base da janela. É bom que você se convença de que esse reconhecimento duplo é de fato necessário. Dados os espaços dos números de seqüência do remetente e do destinatário na Figura 3.23, por exemplo, se não houver ACK para pacote com número `send_base` propagando-se do destinatário ao remetente, este acabará retransmitindo o pacote `send_base`, embora esteja claro (para nós, e não para o remetente!) que o destinatário já recebeu esse pacote. Caso o destinatário não reconhecesse esse pacote, a janela do remetente jamais se deslocaria para a frente! Esse exemplo ilustra um importante aspecto dos protocolos SR (e também de muitos outros). O remetente e o destinatário nem sempre têm uma visão idêntica do que foi recebido corretamente e do que não foi. Para protocolos SR, isso significa que as janelas do remetente e do destinatário nem sempre coincidirão.

A falta de sincronização entre as janelas do remetente e do destinatário tem importantes conseqüências quando nos defrontamos com a realidade de uma faixa finita de números de seqüência. Considere o que poderia acontecer, por exemplo, com uma faixa finita de quatro números de seqüência de pacotes (0, 1, 2, 3) e um tamanho de janela de três. Suponha que os pacotes de 0 a 2 sejam transmitidos, recebidos e reconhecidos corretamente no destinatário. Nesse ponto, a janela do destinatário está sobre o quarto, o quinto e o sexto pacotes, que têm os números de seqüência 3, 0 e 1, respectivamente. Agora, considere dois cenários. No primeiro, mostrado na Figura 3.27(a), os ACKs para os três primeiros pacotes foram perdidos e o remetente retransmite esses pacotes. Assim, o que o destinatário recebe em seguida é um pacote com o número de seqüência 0 — uma cópia do primeiro pacote enviado.

No segundo cenário, mostrado na Figura 3.27(b), os ACKs para os três primeiros pacotes foram entregues corretamente. Assim, o remetente desloca sua janela para a frente e envia o quarto, o quinto e o sexto pacotes com os números de seqüência 3, 0 e 1, respectivamente. O pacote com o número de seqüência 3 é perdido, mas o pacote com o número de seqüência 0 chega — um pacote que contém dados novos.

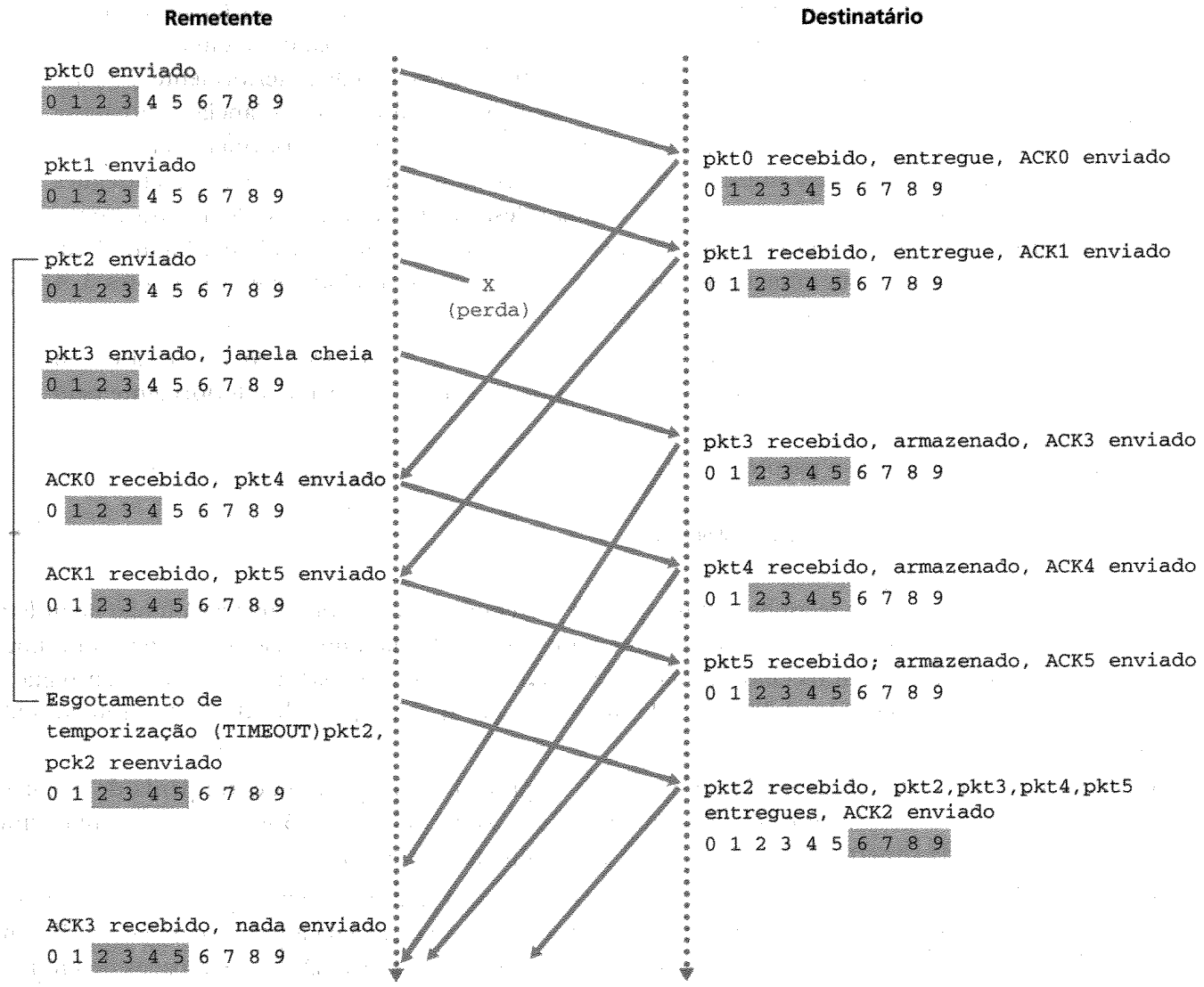


Figura 3.26 Operação SR

Agora, na Figura 3.27, considere o ponto de vista do destinatário, que tem uma cortina imaginária entre o remetente e ele, já que o destinatário não pode 'ver' as ações executadas pelo remetente. Tudo o que o destinatário observa é a seqüência de mensagens que ele recebe do canal e envia para o canal. No que concerne a ele, os dois cenários da Figura 3.27 são *idênticos*. Não há nenhum modo de distinguir a retransmissão do primeiro pacote da transmissão original do quinto pacote. Fica claro que um tamanho de janela que seja igual ao tamanho do espaço de numeração seqüencial menos 1 não vai funcionar. Mas qual deve ser o tamanho da janela? Um problema ao final deste capítulo pede que você demonstre que o tamanho da janela pode ser menor ou igual à metade do tamanho do espaço de numeração seqüencial para os protocolos SR.

Com isso, concluímos nossa discussão sobre protocolos de transferência confiável de dados. Percorremos um longo caminho e apresentamos numerosos mecanismos que, juntos, provêem transferência confiável de dados. A Tabela 3.1 resume esses mecanismos. Agora que já vimos todos eles em operação e podemos enxergar 'o quadro geral', aconselhamos que você leia novamente esta seção para perceber como esses mecanismos foram adicionados pouco a pouco, de modo a abordar modelos (realistas) de complexidade crescente do canal que conecta o remetente ao destinatário ou para melhorar o desempenho dos protocolos.

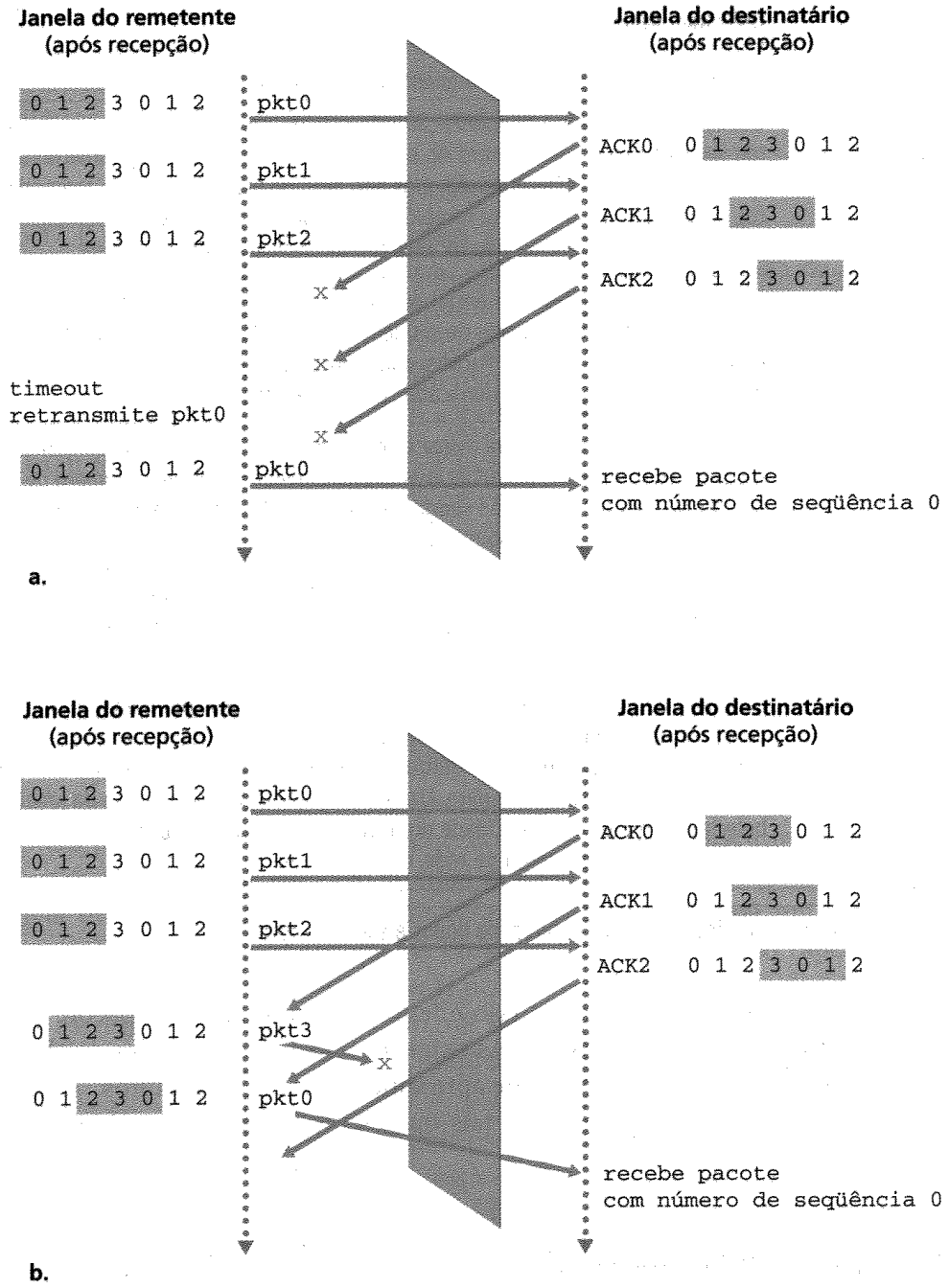


Figura 3.27 Dilema do remetente SR com janelas muito grandes: um novo pacote ou uma retransmissão?

Encerraremos nossa explanação considerando uma premissa remanescente em nosso modelo de canal subjacente. Lembre-se de que admitimos que pacotes não podem ser reordenados dentro do canal entre o remetente e o destinatário. Esta é uma premissa em geral razoável quando o remetente e o destinatário estão conectados por um único fio físico. Contudo, quando o 'canal' que está conectando os dois é uma rede, pode ocorrer reordenação de pacotes. Uma manifestação da reordenação de pacotes é que podem aparecer cópias antigas de um pacote com número de seqüência ou de reconhecimento x, mesmo que nem a janela do remetente nem a do destinatário contenha x. Com a reordenação de pacotes, podemos considerar que o canal essencialmente usa armazenamento de pacotes e emite-os espontaneamente em algum momento qualquer do futuro. Como números de seqüência podem ser reutilizados, devemos tomar um certo cuidado para nos prevenir contra esses pacotes duplicados. A abordagem adotada na prática é garantir que um número de seqüência não seja reutilizado até que o remetente esteja 'certo' de que nenhum pacote enviado anterior-

Mecanismo	Utilização, Comentários
Soma de verificação	Usada para detectar erros de bits em um pacote transmitido.
Temporizador	Usado para controlar a temporização/retransmitir um pacote, possivelmente porque o pacote (ou seu ACK) foi perdido dentro do canal. Como pode ocorrer esgotamento de temporização quando um pacote está atrasado, mas não perdido (esgotamento de temporização prematuro), ou quando um pacote foi recebido pelo destinatário mas o ACK remetente-destinatário foi perdido, um destinatário pode receber cópias duplicadas de um pacote.
Número de seqüência	Usado para numeração seqüencial de pacotes de dados que transitam do remetente ao destinatário. Lacunas nos números de seqüência de pacotes recebidos permitem que o destinatário detecte um pacote perdido. Pacotes com números de seqüência duplicados permitem que o destinatário detecte cópias duplicadas de um pacote.
Reconhecimento	Usado pelo destinatário para avisar o remetente que um pacote ou conjunto de pacotes foi recebido corretamente. Reconhecimentos normalmente portam o número de seqüência do pacote, ou pacotes, que estão sendo reconhecidos. Reconhecimentos podem ser individuais ou cumulativos, dependendo do protocolo.
Reconhecimento negativo	Usado pelo destinatário para avisar o remetente que um pacote não foi recebido corretamente. Reconhecimentos negativos normalmente portam o número de seqüência do pacote que não foi recebido corretamente.
Janela, paralelismo	O remetente pode ficar restrito a enviar somente pacotes com números de seqüência que caiam dentro de uma determinada faixa. Permitindo que vários pacotes sejam transmitidos, ainda que não reconhecidos, a utilização do remetente pode ser aumentada em relação ao modo de operação pare e espere. Em breve veremos que o tamanho da janela pode ser estabelecido com base na capacidade do destinatário receber e fazer buffer de mensagens ou no nível de congestionamento na rede, ou em ambos.

Tabela 3.1 Resumo de mecanismos de transferência confiável de dados e sua utilização

mente com número de seqüência *x* está na rede. Isso é feito admitindo que um pacote não pode 'viver' na rede mais do que um certo tempo máximo fixado. As extensões do TCP para redes de alta velocidade [RFC 1323] usam um tempo de vida máximo de pacote de aproximadamente três minutos. [Sunshine, 1978] descreve um método para usar números de seqüência tais que os problemas de reordenação podem ser completamente evitados.

3.5 Transporte orientado para conexão: TCP

Agora que já vimos os princípios subjacentes à transferência confiável de dados, vamos voltar para o TCP — o protocolo de transporte confiável da camada de transporte, orientado para conexão, da Internet. Nesta seção, veremos que, para poder fornecer transferência confiável de dados, o TCP conta com muitos dos princípios subjacentes discutidos na seção anterior, incluindo detecção de erro, retransmissões, reconhecimentos cumulativos, temporizadores e campos de cabeçalho para números de seqüência e de reconhecimento. O TCP está definido nos RFCs 793, 1122, 1323, 2018 e 2581.

3.5.1 A conexão TCP

Dizemos que o TCP é **orientado para conexão** porque, antes que um processo de aplicação possa começar a enviar dados a outro, os dois processos precisam primeiramente se 'apresentar' — isto é, devem enviar alguns segmentos preliminares um ao outro para estabelecer os parâmetros da transferência de dados em questão. Como parte do estabelecimento da conexão TCP, ambos os lados da conexão iniciarão muitas "variáveis de estado" (muitas das quais serão discutidas nesta seção e na Seção 3.7) associadas com a conexão TCP.

A 'conexão' TCP não é um circuito TDM ou FDM fim-a-fim, como acontece em uma rede de comutação de circuitos. Tampouco é um circuito virtual (veja o Capítulo 1), pois o estado de conexão reside inteiramente nos dois sistemas finais. Como o protocolo TCP roda somente nos sistemas finais, e não nos elementos intermediários da rede (roteadores e comutadores de camada de enlace), os elementos intermediários não mantêm estado de conexão TCP. Na verdade, os roteadores intermediários são completamente alheios às conexões TCP; eles enxergam datagramas, e não conexões.

Uma conexão TCP provê um **serviço full-duplex**: se houver uma conexão TCP entre o processo A em um hospedeiro e o processo B em outro hospedeiro, então os dados da camada de aplicação poderão fluir de A para B ao mesmo tempo em que os dados da camada de aplicação fluem de B para A. A conexão TCP é sempre **ponto a ponto**, isto é, entre um único remetente e um único destinatário. O chamado 'multicast' (veja a Seção 4.7) — a transferência de dados de um remetente para vários destinatários em uma única operação de envio — não é possível com o TCP. Com o TCP, dois hospedeiros é bom; três é demais!

Vamos agora examinar como uma conexão TCP é estabelecida. Suponha que um processo que roda em um hospedeiro queira iniciar a conexão com outro processo em outro hospedeiro. Lembre-se de que o processo que está iniciando a conexão é denominado *processo cliente*, enquanto o outro processo é denominado *processo servidor*. O processo de aplicação cliente primeiramente informa à camada de transporte no cliente que ele quer estabelecer uma conexão com um processo no servidor. Lembre-se (Seção 2.7) de que um programa cliente em Java faz isso emitindo o comando

```
Socket clientSocket = new Socket ("hostname", portNumber);
```

em que `hostname` é o nome do servidor e `portNumber` identifica o processo no servidor. A camada de transporte no cliente então passa a estabelecer uma conexão TCP-servidor. Discutiremos com algum detalhe o procedimento de estabelecimento de conexão ao final desta seção. Por enquanto, basta saber que o cliente primeiramente envia um segmento TCP especial; o servidor responde com um segundo segmento TCP especial e, por fim, o cliente responde novamente com um terceiro segmento especial. Os primeiros dois segmentos não contêm nenhuma "carga útil", isto é, nenhum dado da camada de aplicação; o terceiro desses segmentos pode carregar uma carga útil. Como três segmentos são enviados entre dois hospedeiros, esse procedimento de estabelecimento de conexão é freqüentemente denominado **apresentação de três vias** (*3-way handshake*).

Uma vez estabelecida uma conexão TCP, os dois processos de aplicação podem enviar dados um para o outro. Vamos considerar o envio de dados do processo cliente para o processo servidor. O processo cliente passa uma cadeia de dados através do socket (a porta do processo), como descrito na Seção 2.7. Tão logo passem pelo socket, os dados estão nas mãos do TCP que está rodando no cliente. Como mostra a Figura 3.28, o TCP direciona seus dados para o **buffer de envio** da conexão, que é um dos buffers reservados durante a apresentação de três vias inicial. De quando em quando, o TCP arranca grandes pedaços de dados do buffer de envio. O interessante é que a especificação do TCP [RFC 793] é muito lacônica ao indicar quando o TCP deve realmente enviar dados que estão nos buffers, determinando apenas que o TCP "deve enviar aqueles dados em segmentos segundo sua própria conveniência". A quantidade máxima de dados que pode ser arrancada e colocada em um segmento é limitada pelo **tamanho máximo do segmento** (*maximum segment size* — MSS). O MSS normalmente é estabelecido primeiramente determinando o tamanho do maior quadro de camada de enlace que pode ser enviado pelo hospedeiro remetente local (denominado **unidade máxima de transmissão** — *maximum transmission unit* — MTU) e, em seguida, estabelecendo um MSS que garanta que um segmento TCP (quando encapsulado em um datagrama IP) caberá em um único quadro de camada de enlace. Valores comuns da MTU são 1.460 bytes, 536 bytes e 512 bytes. Também foram propostas abordagens para descobrir a MTU de Caminho (Path MTU) — o maior quadro de camada de enlace que pode ser enviado por todos os enlaces desde a fonte até o destino [RFC 1191]. Note que o MSS é a máxima quantidade de dados de camada de aplicação no segmento, e não o tamanho máximo do segmento TCP incluindo cabeçalhos. (Essa terminologia é confusa, mas temos de conviver com ela, pois já está arraigada.)

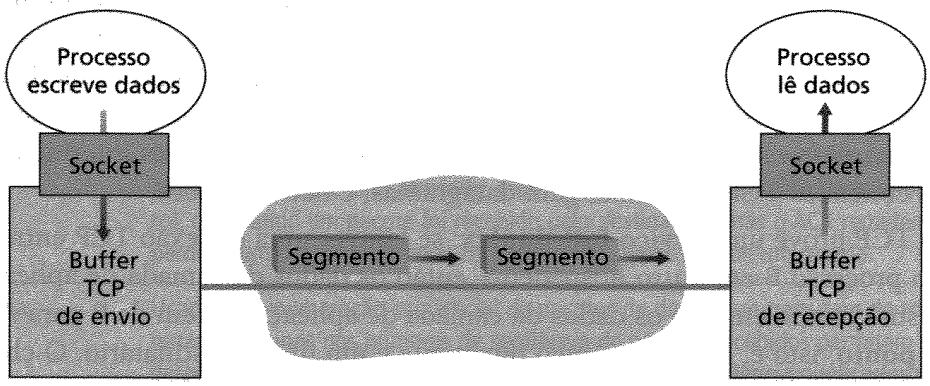


Figura 3.28 Buffers TCP de envio e de recepção

O TCP combina cada porção de dados do cliente com um cabeçalho TCP, formando, assim, **segmentos TCP**. Os segmentos são passados para baixo, para a camada de rede, onde são encapsulados separadamente dentro dos datagramas IP de camada de rede. Os datagramas IP são então enviados para dentro da rede. Quando o TCP recebe um segmento na outra extremidade, os dados do segmento são colocados no buffer de recepção da conexão, como mostra a Figura 3.28. A aplicação lê a cadeia de dados desse buffer. Cada lado da conexão tem seus próprios buffers de envio e seu próprio buffer de recepção. (Você pode ver o applet de controle de fluxo on-line em `http://www.aw.com/kurose_br`, que oferece uma animação dos buffers de envio e de recepção.)

Entendemos, dessa discussão, que uma conexão TCP consiste em buffers, variáveis e um socket de conexão de um processo em um hospedeiro e outro conjunto de buffers, variáveis e um socket de conexão de um processo em outro hospedeiro. Como mencionamos anteriormente, nenhum buffer nem variáveis são alocados à conexão nos elementos da rede (roteadores, comutadores e repetidores) existentes entre os hospedeiros.

História

Vinton Cerf, Robert Kahn e TCP/IP

No início da década de 1970, as redes de comutação de pacotes começaram a proliferar. A ARPAnet — precursora da Internet — era apenas mais uma dentre tantas redes, que tinham, cada uma, seu próprio protocolo. Dois pesquisadores, Vinton Cerf e Robert Kahn, reconheceram a importância de interconectar essas redes e inventaram um protocolo inter-redes denominado TCP/IP, que quer dizer Transmission Control Protocol/Internet Protocol (protocolo de controle de transmissão/protocolo da Internet). Embora no começo Cerf e Kahn considerassem o protocolo como uma entidade única, mais tarde ele foi dividido em duas partes, TCP e IP, que operavam separadamente. Cerf e Kahn publicaram um artigo sobre o TCP/IP em maio de 1974 em *IEEE transactions on communication technology* [Cerf, 1974].

O protocolo TCP/IP, que é o 'feijão-com-arroz' da Internet de hoje, foi elaborado antes dos PCs e das estações de trabalho, antes da proliferação das Ethernets e de outras tecnologias de redes locais, antes da Web, da recepção de vídeo e do bate-papo virtual. Cerf e Kahn perceberam a necessidade de um protocolo de rede que, de um lado, fornecesse amplo suporte para aplicações ainda a serem definidas e que, de outro, permitisse a interoperação de hospedeiros arbitrários e protocolos de camada de enlace.

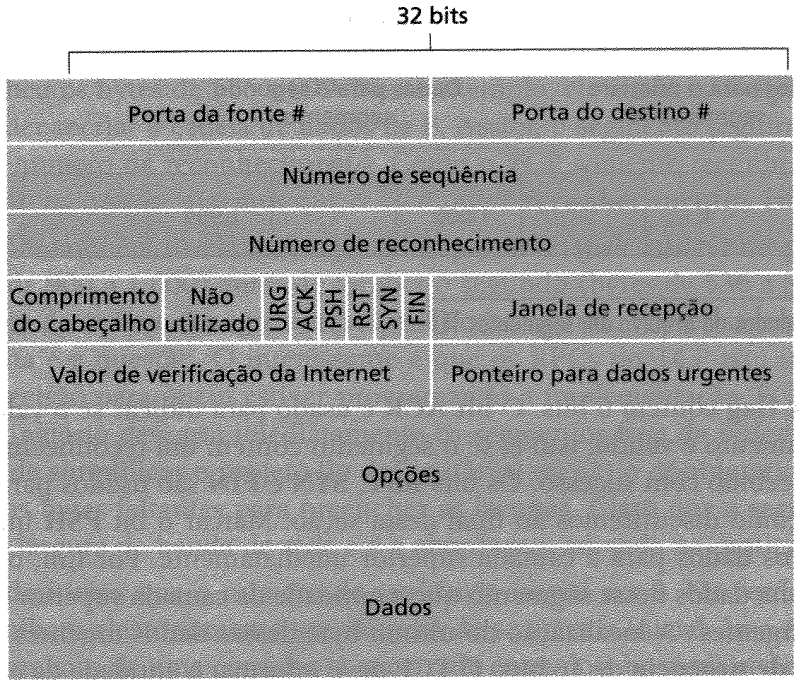


Figura 3.29 Estrutura do segmento TCP

3.5.2 Estrutura do segmento TCP

Agora que examinamos brevemente a conexão TCP, vamos verificar a estrutura do segmento TCP, que consiste em campos de cabeçalho e um campo de dados. O campo de dados contém uma quantidade de dados de aplicação. Como citado antes, o MSS limita o tamanho máximo do campo de dados de um segmento. Quando o TCP envia um arquivo grande, tal como uma imagem de uma página Web, ele comumente fragmenta o segmento em pedaços de tamanho MSS (exceto o último pedaço, que muitas vezes é menor do que o MSS). Aplicações interativas, contudo, muitas vezes transmitem quantidades de dados que são menores do que o MSS. Por exemplo, com aplicações de login remoto como Telnet, o campo de dados do segmento TCP é, muitas vezes, de apenas 1 byte. Como o cabeçalho TCP tem tipicamente 20 bytes (12 bytes mais do que o cabeçalho UDP), o comprimento dos segmentos enviados por Telnet pode ser de apenas 21 bytes.

A Figura 3.29 mostra a estrutura do segmento TCP. Como acontece com o UDP, o cabeçalho inclui **números de porta de fonte e de destino**, que são usados para multiplexação e demultiplexação de dados de/para aplicações de camadas superiores e, assim como no UDP, inclui um **campo de soma de verificação**. Um cabeçalho de segmento TCP também contém os seguintes campos:

- O **campo de número de seqüência** de 32 bits e o **campo de número de reconhecimento** de 32 bits são usados pelos TCPs remetente e destinatário na implementação de um serviço confiável de transferência de dados, como discutido a seguir.
- O campo de **janela de recepção** de 16 bits é usado para controle de fluxo. Veremos em breve que esse campo é usado para indicar o número de bytes que um destinatário está disposto a aceitar.
- O **campo de comprimento de cabeçalho** de 4 bits especifica o comprimento do cabeçalho TCP em palavras de 32 bits. O cabeçalho TCP pode ter comprimento variável devido ao campo de opções TCP. (O campo de opções TCP normalmente está vazio, de modo que o comprimento do cabeçalho TCP típico é 20 bytes.)
- O **campo de opções**, opcional e de comprimento variável, é usado quando um remetente e um destinatário negociam o MSS, ou como um fator de aumento de escala da janela para utilização em redes de alta velocidade. Uma opção de marca de tempo é também definida. Consulte o RFC 854 e o RFC 1323 para detalhes adicionais.

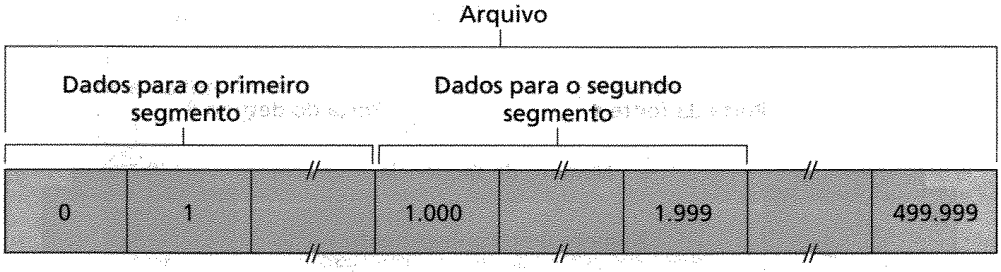


Figura 3.30 Dividindo os dados do arquivo em segmentos TCP

O **campo de flag** contém 6 bits. O **bit ACK** é usado para indicar se o valor carregado no campo de reconhecimento é válido, isto é, se o segmento contém um reconhecimento para um segmento que foi recebido com sucesso. Os bits **RST**, **SYN** e **FIN** são usados para estabelecer e encerrar a conexão, como discutiremos ao final desta seção. Marcar o bit **PSH** indica que o destinatário deve passar os dados para a camada superior imediatamente. Por fim, o bit **URG** é usado para mostrar que há dados nesse segmento que a entidade da camada superior do lado remetente marcou como 'urgentes'. A localização do último byte desses dados urgentes é indicada pelo **campo de ponteiro de urgência** de 16 bits. O TCP deve informar à entidade da camada superior do lado destinatário quando existem dados urgentes e passar a ela um ponteiro para o final desses dados. (Na prática, o PSH, o URG e o ponteiro de dados urgentes não são usados. Contudo, mencionamos esses campos para descrever todos.)

Números de seqüência e números de reconhecimento

Dois dos mais importantes campos do cabeçalho do segmento TCP são o campo de número de seqüência e o campo de número de reconhecimento. Esses campos são parte fundamental do serviço de transferência confiável de dados do TCP. Mas, antes de discutirmos como esses campos são utilizados, vamos explicar exatamente o que o TCP coloca nesses campos.

O TCP vê os dados como uma cadeia de bytes não estruturada, mas ordenada. O uso que o TCP faz dos números de seqüência reflete essa visão, pois esses números são aplicados sobre a cadeia de bytes transmitidos, e *não* sobre a série de segmentos transmitidos. O **número de seqüência para um segmento** é o número do primeiro byte do segmento. Vamos ver um exemplo. Suponha que um processo no hospedeiro A queira enviar uma cadeia de dados para um processo no hospedeiro B por uma conexão TCP. O TCP do hospedeiro A vai implicitamente numerar cada byte da cadeia de dados. Suponha que a cadeia de dados consista em um arquivo composto de 500.000 bytes, que o MSS seja de 1.000 bytes e que seja atribuído o número 0 ao primeiro byte da cadeia de dados. Como mostra a Figura 3.30, o TCP constrói 500 segmentos a partir da cadeia de dados. O primeiro segmento recebe o número de seqüência 0; o segundo, o número de seqüência 1.000; o terceiro, o número de seqüência 2.000, e assim por diante. Cada número de seqüência é inserido no campo de número de seqüência no cabeçalho do segmento TCP apropriado.

Vamos agora considerar os números de reconhecimento. Esses números são um pouco mais complicados do que os números de seqüência. Lembre-se de que o TCP é *full-duplex*, portanto o hospedeiro A pode estar recebendo dados do hospedeiro B enquanto envia dados ao hospedeiro B (como parte da mesma conexão TCP). Cada um dos segmentos que chegam do hospedeiro B tem um número de seqüência para os dados que estão fluindo de B para A. *O número de reconhecimento que o hospedeiro A atribui a seu segmento é o número de seqüência do próximo byte que ele estiver aguardando do hospedeiro B.* É bom examinarmos alguns exemplos para entendermos o que está acontecendo aqui. Suponha que o hospedeiro A tenha recebido do hospedeiro B todos os bytes numerados de 0 a 535 e também que esteja prestes a enviar um segmento ao hospedeiro B. O hospedeiro A está esperando pelo byte 536 e por todos os bytes subseqüentes da corrente de dados do hospedeiro B. Assim, ele coloca o número 536 no campo de número de reconhecimento do segmento que envia para o hospedeiro B.

Como outro exemplo, suponha que o hospedeiro A tenha recebido um segmento do hospedeiro B contendo os bytes de 0 a 535 e outro segmento contendo os bytes de 900 a 1.000. Por alguma razão, o hospedeiro A ainda não recebeu os bytes de 536 a 899. Nesse exemplo, ele ainda está esperando pelo byte 536 (e os superiores) para poder recriar a cadeia de dados de B. Assim, o segmento seguinte que A envia a B conterá 536 no campo de número de reconhecimento. Como o TCP somente reconhece bytes até o primeiro byte que estiver faltando na cadeia, dizemos que o TCP provê **reconhecimentos cumulativos**.

Esse último exemplo também revela uma questão importante, mas sutil. O hospedeiro A recebeu o terceiro segmento (bytes de 900 a 1.000) antes de receber o segundo (bytes de 536 a 899). Portanto, o terceiro segmento chegou fora de ordem. E o que um hospedeiro faz quando recebe segmentos fora de ordem em uma conexão TCP? Esta é a questão. O interessante é que os RFCs do TCP não impõem nenhuma regra para isso e deixam a decisão para quem estiver programando a implementação TCP. Há basicamente duas alternativas: (1) o destinatário descarta imediatamente os segmentos fora de ordem (o que, como discutimos anteriormente, pode simplificar o projeto do destinatário) ou (2) o destinatário conserva os bytes fora de ordem e espera pelos bytes faltantes para preencher as lacunas. Claro que a segunda alternativa é mais eficiente em termos de largura de banda de rede e é a abordagem adotada na prática.

Na Figura 3.30, admitimos que o número de seqüência inicial era 0. Na verdade, ambos os lados de uma conexão TCP escolhem aleatoriamente um número de seqüência inicial. Isso é feito para minimizar a possibilidade de um segmento de uma conexão já encerrada entre dois hospedeiros e ainda presente na rede ser tomado por um segmento válido em uma conexão posterior entre esses dois mesmos hospedeiros (que também podem estar usando os mesmos números de porta da conexão antiga) [Sunshine, 1978].

Telnet: um estudo de caso para números de seqüência e números de reconhecimento

O Telnet, definido no RFC 854, é um protocolo popular de camada de aplicação utilizado para fazer login remoto. Ele roda sobre TCP e é projetado para trabalhar entre qualquer par de hospedeiros. Diferentemente das aplicações de transferência de dados em grandes blocos, que foram discutidas no Capítulo 2, o Telnet é uma aplicação interativa. Discutiremos, agora, um exemplo de Telnet, pois ele ilustra muito bem números de seqüência e de reconhecimento do TCP. Observamos que muitos usuários agora preferem usar o protocolo 'ssh' em vez do Telnet, visto que dados enviados por uma conexão Telnet (incluindo senhas!) não são criptografados, o que torna essa aplicação vulnerável a ataques de bisbilhoteiros (como discutiremos na Seção 8.7).

Suponha que o hospedeiro A inicie uma sessão Telnet com o hospedeiro B. Como o hospedeiro A inicia a sessão, ele é rotulado de cliente, enquanto o hospedeiro B é rotulado de servidor. Cada caractere digitado pelo usuário (no cliente) será enviado ao hospedeiro remoto; este devolverá uma cópia ('eco') de cada caractere, que será apresentada na tela Telnet do usuário. Esse eco é usado para garantir que os caracteres vistos pelo usuário do Telnet já foram recebidos e processados no local remoto. Assim, cada caractere atravessa a rede duas vezes entre o momento em que o usuário aperta o teclado e o momento em que o caractere é apresentado em seu monitor.

Suponha agora que o usuário digite a letra 'C' e saia para tomar um café. Vamos examinar os segmentos TCP que são enviados entre o cliente e o servidor. Como mostra a Figura 3.31, admitamos que os números de seqüência iniciais sejam 42 e 79 para cliente e servidor, respectivamente. Lembre-se de que o número de seqüência de um segmento será o número de seqüência do primeiro byte do campo de dados. Assim, o primeiro segmento enviado do cliente terá número de seqüência 42; o primeiro segmento enviado do servidor terá número de seqüência 79. Note que o número de reconhecimento será o número de seqüência do próximo byte de dados que o hospedeiro estará aguardando. Após o estabelecimento da conexão TCP, mas antes de quaisquer dados serem enviados, o cliente ficará esperando pelo byte 79 e o servidor, pelo byte 42.

Como mostra a Figura 3.31, são enviados três segmentos. O primeiro é enviado do cliente ao servidor, contendo, em seu campo de dados, um byte com a representação ASCII para a letra 'C'. O primeiro segmento também tem 42 em seu campo de número de seqüência, como acabamos de descrever. E mais, como o cliente ainda não recebeu nenhum dado do servidor, esse segmento terá o número 79 em seu campo de número de reconhecimento.

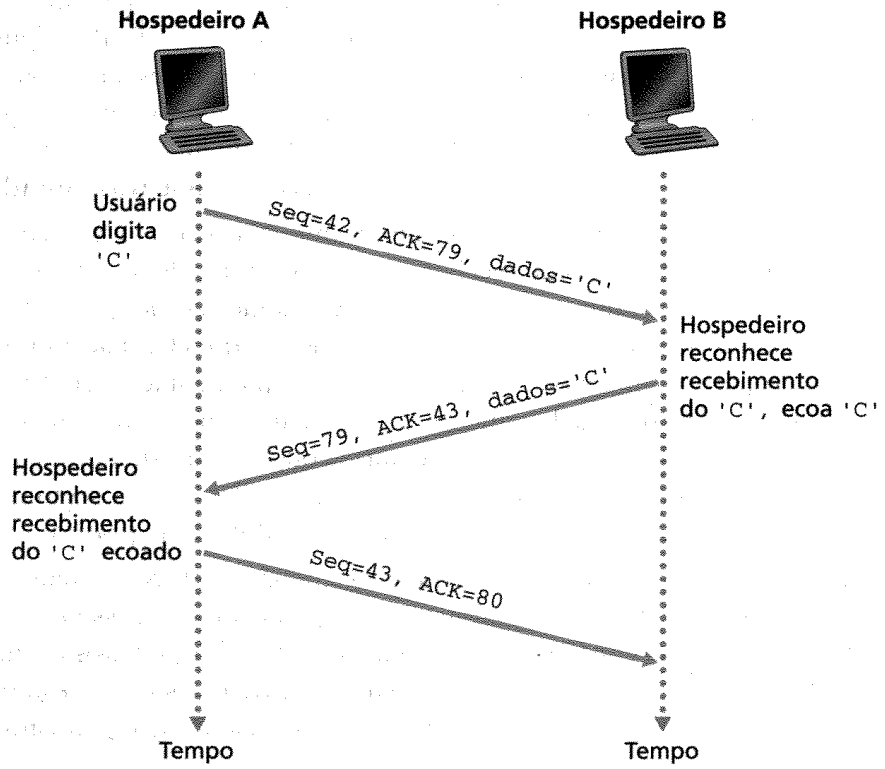


Figura 3.31 Números de seqüência e de reconhecimento para uma aplicação Telnet simples sobre TCP

O segundo segmento é enviado do servidor ao cliente. Esse segmento tem dupla finalidade. A primeira finalidade é fornecer um reconhecimento para os dados que o servidor recebeu. Ao colocar 43 no campo de reconhecimento, o servidor está dizendo ao cliente que recebeu com sucesso tudo até o byte 42 e agora está aguardando os bytes de 43 em diante. A segunda finalidade desse segmento é ecoar a letra 'C'. Assim, o segundo segmento tem a representação ASCII de 'C' em seu campo de dados. Ele tem o número de seqüência 79, que é o número de seqüência inicial do fluxo de dados de servidor para cliente dessa conexão TCP, pois este é o primeiríssimo byte de dados que o servidor está enviando. Note que o reconhecimento para dados do cliente para o servidor é levado em um segmento que carrega dados do servidor para o cliente. Dizemos que este reconhecimento **pegou uma carona** no segmento de dados do servidor para o cliente. Tecnicamente, essa 'carona' recebe o nome de *piggyback*.

O terceiro segmento é enviado do cliente ao servidor. Seu único propósito é reconhecer os dados que recebeu do servidor. (Lembre-se de que o segundo segmento continha dados — a letra 'C' — do servidor para o cliente.) Esse segmento tem um campo de dados vazio (isto é, o reconhecimento não está pegando carona com nenhum dado do cliente para o servidor), tem o número 80 no campo do número de reconhecimento porque o cliente recebeu a cadeia de dados até o byte com número de seqüência 79 e agora está aguardando os bytes de 80 em diante. É possível que você esteja pensando que é estranho que esse segmento também tenha um número de seqüência, já que não contém dados. Mas, como o TCP tem um campo de número de seqüência, o segmento precisa apresentar algum número para preenchê-lo.

3.5.3 Estimativa do tempo de viagem de ida e volta e de esgotamento de temporização

O TCP, assim como o nosso protocolo `rdt` da Seção 3.4, utiliza um mecanismo de controle de temporização/retransmissão para recuperar segmentos perdidos. Embora conceitualmente simples, surgem muitas questões sutis quando implementamos um mecanismo de controle de temporização/retransmissão em um protocolo real como o TCP. Talvez a pergunta mais óbvia seja a duração dos intervalos de controle. Evidentemente, esse intervalo deve ser maior do que o tempo de viagem de ida e volta da conexão (RTT),

isto é, o tempo decorrido entre o envio de um segmento e seu reconhecimento. Se não fosse assim, seriam enviadas retransmissões desnecessárias. Mas quão maior deve ser o intervalo e, antes de mais nada, como o RTT deve ser estimado? Deve-se associar um temporizador a cada segmento não reconhecido? São tantas perguntas! Nesta seção, nossa discussão se baseia no trabalho de [Jacobson, 1988] sobre TCP e nas recomendações da IETF vigentes para o gerenciamento de temporizadores TCP [RFC 2988].

Estimativa do tempo de viagem de ida e volta

Vamos iniciar nosso estudo do gerenciamento do temporizador TCP considerando como esse protocolo estima o tempo de viagem de ida e volta entre remetente e destinatário, o que apresentaremos a seguir. O RTT para um segmento, denominado `SampleRTT` no exemplo, é a quantidade de tempo transcorrido entre o momento em que o segmento é enviado (isto é, passado ao IP) e o momento em que é recebido um reconhecimento para o segmento. Em vez de medir um `SampleRTT` para cada segmento transmitido, a maioria das implementações de TCP executa apenas uma medição de `sampleRTT` por vez. Isto é, em qualquer instante, o `SampleRTT` estará sendo estimado para apenas um dos segmentos transmitidos mas ainda não reconhecidos, o que resulta em um novo valor de `SampleRTT` para aproximadamente cada RTT. E mais, o TCP nunca computa um `SampleRTT` para um segmento que foi retransmitido; apenas mede-o para segmentos que foram transmitidos uma vez. (Um dos problemas ao final do capítulo perguntará por quê.)

Obviamente, os valores de `SampleRTT` sofrerão variação de segmento para segmento devido a congestionamento nos roteadores e a variações de carga nos sistemas finais. Por causa dessa variação, qualquer dado valor de `SampleRTT` pode ser atípico. Portanto, para estimar um RTT típico, é natural tomar alguma espécie de média dos valores de `SampleRTT`. O TCP mantém uma média, denominada `EstimatedRTT`, dos valores de `SampleRTT`. Ao obter um novo `SampleRTT`, o TCP atualiza `EstimatedRTT` de acordo com a seguinte fórmula:

$$\texttt{EstimatedRTT} = (1 - \alpha) \cdot \texttt{EstimatedRTT} + \alpha \cdot \texttt{SampleRTT}.$$

Essa fórmula está escrita sob a forma de um comando de linguagem de programação — o novo valor de `EstimatedRTT` é uma combinação ponderada entre o valor anterior de `EstimatedRTT` e o novo valor para `SampleRTT`. O valor recomendado de α é $\alpha = 0{,}125$ (isto é, 1/8) [RFC 2988], caso em que essa fórmula se torna:

$$\texttt{EstimatedRTT} = 0{,}875 \cdot \texttt{EstimatedRTT} + 0{,}125 \cdot \texttt{SampleRTT}.$$

Note que `EstimatedRTT` é uma média ponderada dos valores de `SampleRTT`. Como veremos em um exercício ao final deste capítulo, essa média ponderada atribui um peso maior às amostras recentes do que às amostras antigas. Isso é natural, pois as amostras mais recentes refletem melhor o estado atual de congestionamento da rede. Em estatística, esse tipo de média é denominada **média móvel exponencial ponderada**. A palavra 'exponencial' aparece na MMEP porque o peso atribuído a um dado `SampleRTT` diminui exponencialmente à medida que as atualizações são realizadas. Os exercícios pedirão que você derive o termo exponencial em `EstimatedRTT`.

A Figura 3.32 mostra os valores de `SampleRTT` e `EstimatedRTT` para um valor de $\alpha = 1/8$, para uma conexão TCP entre gaia.cs.umass.edu (em Amherst, Massachusetts) e fantasia.eurecom.fr (no sul da França). Fica claro que as variações em `SampleRTT` são atenuadas no cálculo de `EstimatedRTT`.

Além de ter uma estimativa do RTT, também é valioso ter uma medida de sua variabilidade. O [RFC 2988] define a variação do RTT, `DevRTT`, como uma estimativa do desvio típico entre `SampleRTT` e `EstimatedRTT`:

$$\texttt{DevRTT} = (1 - \beta) \cdot \texttt{DevRTT} + \beta \cdot |\ \texttt{SampleRTT} - \texttt{EstimatedRTT}\ |$$

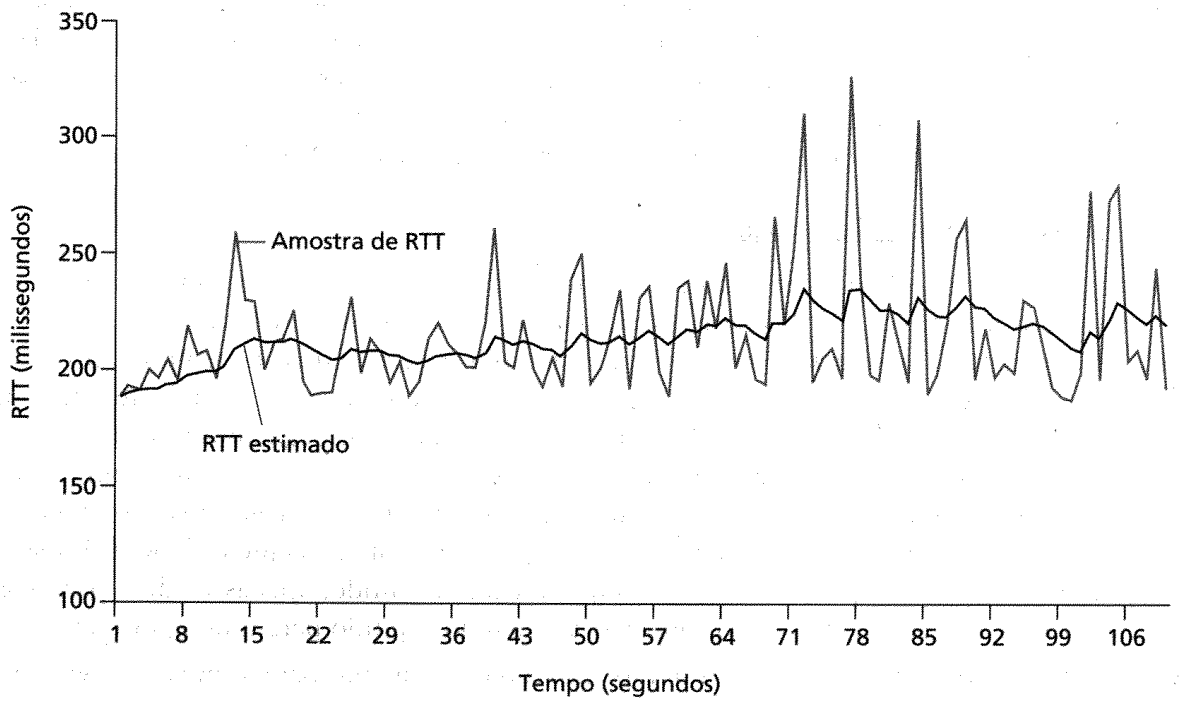


Figura 3.32 Amostras de RTTs e RTTs estimados

Note que `DevRTT` é uma MMEP da diferença entre `SampleRTT` e `EstimatedRTT`. Se os valores de `SampleRTT` apresentarem pouca variação, então `DevRTT` será pequeno; por outro lado, se houver muita variação, `DevRTT` será grande. O valor recomendado para β é 0,25.

Princípios na prática

O TCP fornece transferência confiável de dados usando reconhecimentos positivos e temporizadores, de modo muito parecido com o que estudamos na Seção 3.4. O protocolo reconhece dados que foram recebidos corretamente e retransmite segmentos quando entende que eles ou seus reconhecimentos correspondentes foram perdidos ou corrompidos. Certas versões do TCP também têm um mecanismo NAK implícito — com o mecanismo de retransmissão rápida do TCP, o recebimento de três ACKs duplicados para um dado segmento serve como um NAK implícito para o segmento seguinte, acionando a retransmissão daquele segmento antes que o tempo se esgote. O TCP usa números de seqüência para permitir que o destinatário identifique segmentos perdidos ou duplicados. Exatamente como no caso de nosso protocolo de transferência confiável de dados `rdt3.0`, o TCP em si não pode determinar com certeza se um segmento, ou seu ACK, está perdido, corrompido ou excessivamente atrasado. No remetente, a resposta do TCP será a mesma: retransmitir o segmento em questão. O TCP também utiliza paralelismo, permitindo que o remetente tenha, a qualquer tempo, múltiplos segmentos transmitidos mas ainda não reconhecidos. Vimos anteriormente que o paralelismo pode melhorar muito a vazão de uma sessão quando a razão entre o tamanho do segmento e o atraso de viagem de ida e volta é pequena. O número específico de segmentos não reconhecidos que um remetente pode ter é determinado pelos mecanismos de controle de fluxo e controle de congestionamento do TCP. O controle de fluxo do TCP é discutido no final desta seção; o controle de congestionamento do TCP é discutido na Seção 3.7. Por enquanto, devemos apenas ficar cientes de que o TCP remetente usa paralelismo.

Estabelecimento e gerenciamento da temporização de retransmissão

Dados valores de EstimatedRTT e DevRTT, que valor deve ser utilizado para a temporização de retransmissão do TCP? É óbvio que o intervalo deve ser maior ou igual a EstimatedRTT, caso contrário seriam enviadas retransmissões desnecessárias. Mas a temporização de retransmissão não deve ser muito maior do que EstimatedRTT, senão, quando um segmento fosse perdido, o TCP não o retransmitiria rapidamente, o que resultaria em grandes atrasos de transferência de dados. Portanto, é desejável que o valor estabelecido para a temporização seja igual a EstimatedRTT mais uma certa margem, que deverá ser grande quando houver muita variação nos valores de SampleRTT e pequena quando houver pouca variação. Assim, o valor de DevRTT deve entrar em jogo. Todas essas considerações são levadas em conta no método do TCP para determinar a temporização de retransmissão:

```
TimeoutInterval = EstimatedRTT + 4 · DevRTT
```

3.5.4 Transferência confiável de dados

Lembre-se de que o serviço da camada de rede da Internet (serviço IP) não é confiável. O IP não garante a entrega de datagramas na ordem correta nem a integridade de seus dados nos datagramas. Com o serviço IP, os datagramas podem transbordar dos buffers dos roteadores e jamais alcançar seu destino; podem também chegar fora de ordem. Além disso, os bits dos datagramas podem ser corrompidos (passar de 0 para 1 e vice-versa). Como os segmentos da camada de transporte são carregados pela rede por datagramas IPs, também eles podem sofrer esses mesmos problemas.

O TCP cria um **serviço de transferência confiável de dados** sobre o serviço de melhor esforço do IP. Esse serviço de transferência garante que a cadeia de dados que um processo lê a partir de seu buffer de recebimento TCP não está corrompida, não tem lacunas, não tem duplicações e está em seqüência, isto é, a cadeia de bytes é exatamente a mesma cadeia de bytes enviada pelo sistema final que está do outro lado da conexão. O modo como o TCP provê transferência confiável de dados envolve muitos dos princípios estudados na Seção 3.4.

Quando desenvolvemos anteriormente técnicas de transferência confiável de dados, era conceitualmente mais fácil admitir que existia um temporizador individual associado com cada segmento transmitido mas ainda não reconhecido. Embora, em teoria, isso seja ótimo, o gerenciamento de temporizadores pode exigir considerável sobrecarga. Assim, os procedimentos recomendados no [RFC 2988] para gerenciamento de temporizadores TCP utilizam apenas um *único* temporizador de retransmissão, mesmo que haja vários segmentos transmitidos ainda não reconhecidos. O protocolo TCP apresentado nesta seção segue essa recomendação.

Discutiremos como o TCP provê transferência confiável de dados em duas etapas incrementais. Em primeiro lugar, apresentamos uma descrição muito simplificada de um remetente TCP que utiliza somente controle de temporizadores para se recuperar da perda de segmentos; em seguida, apresentaremos uma descrição mais complexa que utiliza reconhecimentos duplicados além de temporizadores de retransmissão. Na discussão que se segue, admitimos que os dados estão sendo enviados em uma direção somente, do hospedeiro A ao hospedeiro B, e que o hospedeiro A está enviando um arquivo grande.

A Figura 3.33 apresenta uma descrição muito simplificada de um remetente TCP.

Vemos que há três eventos importantes relacionados com a transmissão e a retransmissão de dados no TCP remetente: dados recebidos da aplicação acima; esgotamento do temporizador e recebimento de ACK. Quando ocorre o primeiro evento importante, o TCP recebe dados da camada de aplicação, encapsula-os em um segmento e passa-o ao IP. Note que cada segmento inclui um número de seqüência que é o número da corrente de bytes do primeiro byte de dados no segmento, como descrito na Seção 3.5.2. Note também que, se o temporizador não estiver funcionando naquele instante para algum outro segmento, o TCP aciona

```
/* Presuma que o remetente não seja compelido pelo fluxo de TCP ou controle de congestio-
namento, que o tamanho dos dados vindos de cima seja maior do que o MSS e que a transfe-
rência de dados ocorra apenas em uma direção. */

NextSeqNum=InitialSeqNumber
SendBase=InitialSeqNumber

loop (forever) {
    switch (event)

        event: data received from the application above
            create TCP segment with sequence number NextSeqNum
            if (timer currently not running)
                start timer
            pass segment to IP
            NextSeqNum=NextSeqNum=length (data)
            break;

        event: timer timeout
            retransmit not-yet-acknowledged segment with
                smallest sequence number
            start timer
            break;

        event: ACK received, with ACK field value of y
            if (y > SendBase) {
                SendBase=y
                If (there are currently any not-yet-acknowledged segments)
                    start timer
                }
                break;

    } /* fim do loop forever */
```

Figura 3.33 Remetente TCP simplificado

o temporizador quando o segmento é passado para o IP. (Fica mais fácil se você imaginar que o temporizador está associado com o mais antigo segmento não reconhecido.) O intervalo de expiração para esse temporizador é o `TimeoutInterval`, calculado a partir de `EstimatedRTT` e `DevRTT`, como descrito na Seção 3.5.3.

O segundo evento importante é o esgotamento do temporizador. O TCP responde a esse evento retransmitindo o segmento que causou o esgotamento da temporização e então reinicia o temporizador.

O terceiro evento importante que deve ser manipulado pelo TCP remetente é a chegada de um segmento de reconhecimento (ACK) do destinatário (mais especificamente, um segmento contendo um valor de campo de ACK válido). Quando da ocorrência desse evento, o TCP compara o valor do ACK, `y`, com sua variável `SendBase`. A variável de estado `SendBase` do TCP é o número de seqüência do mais antigo byte não reconhecido. (Assim, `SendBase-1` é o número de seqüência do último byte que se sabe ter sido recebi-

do pelo destinatário corretamente e na ordem certa.) Como comentamos anteriormente, o TCP usa reconhecimentos cumulativos, de modo que y reconhece o recebimento de todos os bytes antes do byte número y. Se y > SendBase, então o ACK está reconhecendo um ou mais bytes não reconhecidos anteriormente. Desse modo, o remetente atualiza sua variável SendBase e também reinicia o temporizador se houver quaisquer segmentos ainda não reconhecidos.

Alguns cenários interessantes

Acabamos de descrever uma versão muito simplificada do modo como o TCP provê transferência confiável de dados, mas mesmo essa descrição tão simplificada tem muitas sutilezas. Para ter uma boa percepção de como esse protocolo funciona, vamos agora examinar alguns cenários simples. A Figura 3.34 ilustra o primeiro cenário em que um hospedeiro A envia um segmento ao hospedeiro B. Suponha que esse segmento tenha número de seqüência 92 e contenha 8 bytes de dados. Após enviá-lo, o hospedeiro A espera por um segmento de B com número de reconhecimento 100. Embora o segmento de A seja recebido em B, o reconhecimento de B para A se perde. Nesse caso, ocorre o evento de expiração do temporizador e o hospedeiro A retransmite o mesmo segmento. É claro que, quando recebe a retransmissão, o hospedeiro B observa, pelo número de seqüência, que o segmento contém dados que já foram recebidos. Assim, o TCP no hospedeiro B descarta os bytes do segmento retransmitido.

Em um segundo cenário, mostrado na Figura 3.35, o hospedeiro A envia dois segmentos seguidos. O primeiro segmento tem número de seqüência 92 e 8 bytes de dados. O segundo segmento tem número de seqüência 100 e 20 bytes de dados. Suponha que ambos cheguem intactos em B e que B envie dois reconhecimentos separados para cada um desses segmentos. O primeiro desses reconhecimentos tem número de reconhecimento 100; o segundo, número de reconhecimento 120. Suponha agora que nenhum dos reconhecimentos chegue ao hospedeiro A antes do esgotamento do temporizador. Quando ocorre o evento de expiração do temporizador, o hospedeiro A reenvia o primeiro segmento com número de seqüência 92 e reinicia o temporizador. Contanto que o ACK do segundo segmento chegue antes que o temporizador expire novamente, o segundo segmento não será retransmitido.

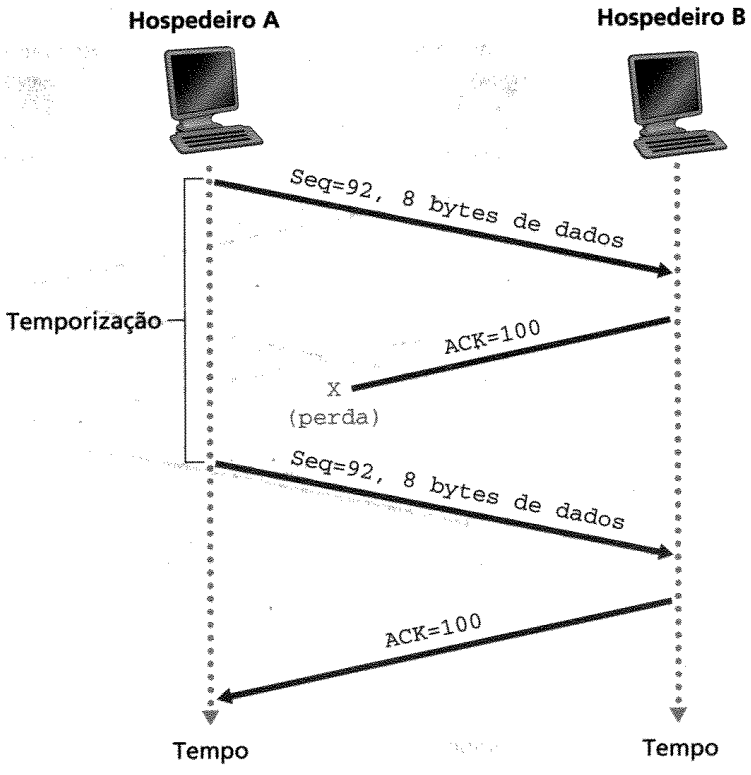


Figura 3.34 Retransmissão devido a um reconhecimento perdido

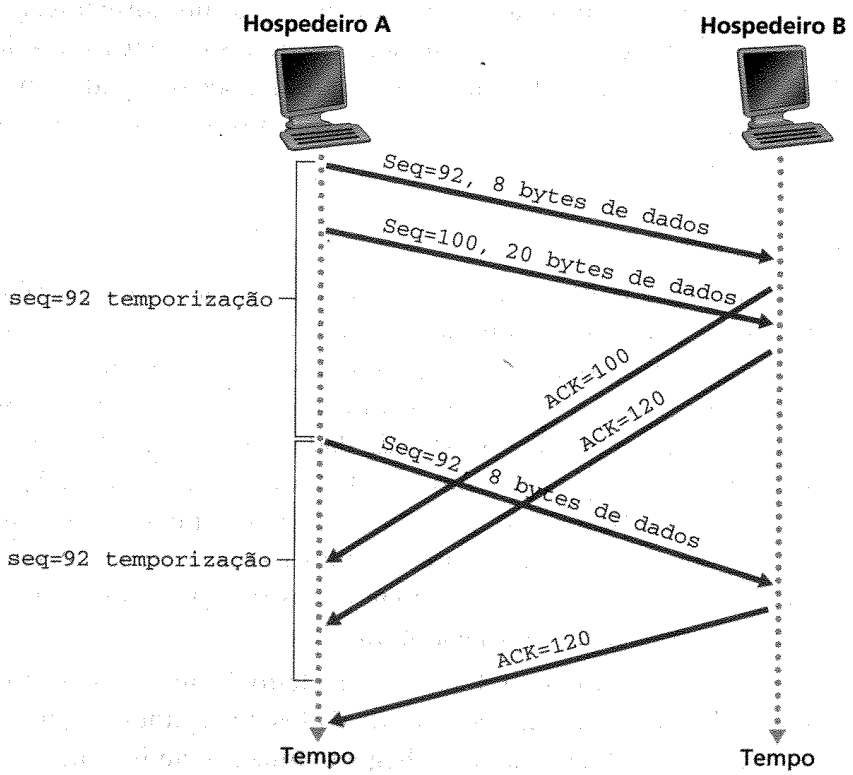


Figura 3.35 Segmento 100 não retransmitido

Em um terceiro e último cenário, suponha que o hospedeiro A envie dois segmentos, exatamente como no segundo exemplo. O reconhecimento do primeiro segmento é perdido na rede, mas, um pouco antes do evento de expiração, o hospedeiro A recebe um reconhecimento com número 120. O hospedeiro A, portanto, sabe que B recebeu *tudo* até o byte 119; portanto, ele não reenvia nenhum dos dois segmentos. Esse cenário está ilustrado na Figura 3.36.

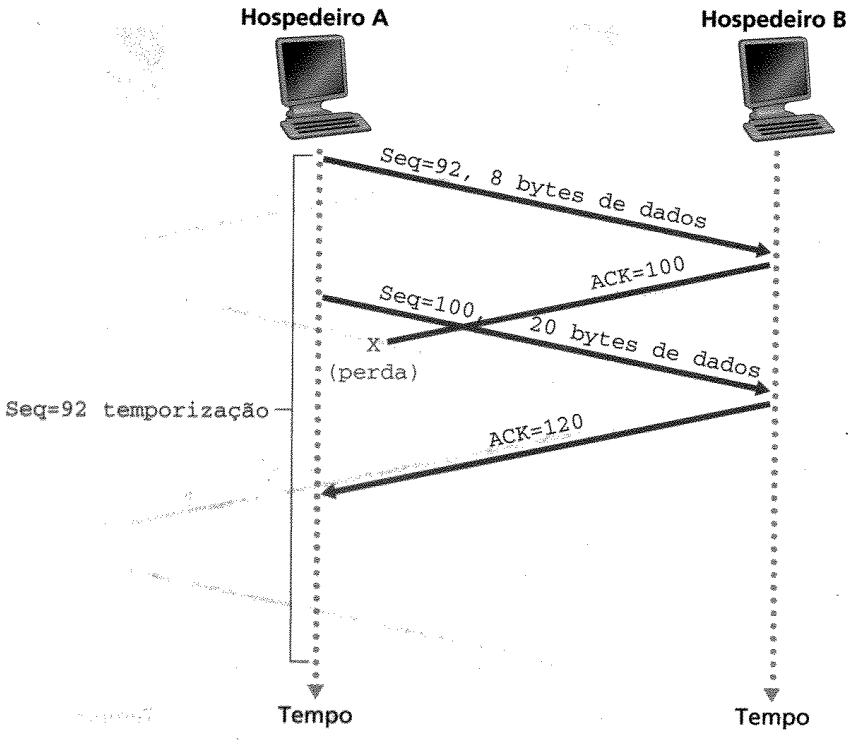


Figura 3.36 Um reconhecimento cumulativo evita retransmissão do primeiro segmento

Duplicação do tempo de expiração

Discutiremos agora algumas modificações empregadas por grande parte das implementações do TCP. A primeira refere-se à duração do tempo de expiração após a expiração de um temporizador. Nessa modificação, sempre que ocorre o evento de expiração do temporizador, o TCP retransmite o segmento ainda não reconhecido que tenha o menor número de seqüência, como descrevemos anteriormente. Mas, a cada retransmissão, o TCP ajusta o próximo tempo de expiração para o dobro do valor anterior em vez de derivá-lo dos últimos `EstimatedRTT` e `DevRTT` (como descrito na Seção 3.5.3). Por exemplo, suponha que o `TimeoutInterval` associado com o mais antigo segmento ainda não reconhecido seja 0,75 segundo quando o temporizador expirar pela primeira vez. O TCP então retransmite esse segmento e ajusta o novo tempo de expiração para 1,5 segundo. Se o temporizador expirar novamente 1,5 segundo mais tarde, o TCP retransmitirá novamente esse segmento, agora ajustando o tempo de expiração para 3,0 segundos. Assim, o tempo aumenta exponencialmente após cada retransmissão. Todavia, sempre que o temporizador é iniciado após qualquer um dos outros dois eventos (isto é, dados recebidos da aplicação acima e ACK recebido), o `TimeoutInterval` será derivado dos valores mais recentes de `EstimatedRTT` e `DevRTT`.

Essa modificação provê uma forma limitada de controle de congestionamento. (Maneiras mais abrangentes de controle de congestionamento TCP serão estudadas na Seção 3.7). A causa mais provável da expiração do temporizador é o congestionamento na rede, isto é, um número muito grande de pacotes chegando a uma (ou mais) fila de roteadores no caminho entre a fonte e o destino, o que provoca descarte de pacotes e/ou longos atrasos de fila. Se as fontes continuarem a retransmitir pacotes persistentemente durante um congestionamento, ele pode piorar. Para que isso não aconteça, o TCP age mais educadamente: cada remetente retransmite após intervalos cada vez mais longos. Veremos que uma idéia semelhante a essa é utilizada pela Ethernet, quando estudarmos CSMA/CD no Capítulo 5.

Retransmissão rápida

Um dos problemas de retransmissões acionadas por expiração de temporizador é que o período de expiração pode ser relativamente longo. Quando um segmento é perdido, esse longo período força o remetente a atrasar o reenvio do pacote perdido, conseqüentemente aumentando o atraso fim-a-fim. Felizmente, o remetente pode com freqüência detectar perda de pacote bem antes de ocorrer o evento de expiração, observando os denominados ACKs duplicados. Um **ACK duplicado** é um ACK que reconhece novamente um segmento para o qual o remetente já recebeu um reconhecimento anterior. Para entender a resposta do remetente a um ACK duplicado, devemos examinar por que o destinatário envia um ACK duplicado em primeiro lugar. A Tabela 3.2 resume a política de geração de ACKs do TCP destinatário [RFC 1122, RFC 2581].

Evento	Ação do TCP Destinatário
Chegada de segmento na ordem com número de seqüência esperado. Todos os dados até o número de seqüência esperado já reconhecidos.	ACK retardado. Espera de até 500 milissegundos pela chegada de um outro segmento na ordem. Se o segmento seguinte na ordem não chegar nesse intervalo, envia um ACK.
Chegada de segmento na ordem com número de seqüência esperado. Um outro segmento na ordem esperando por transmissão de ACK.	Envio imediato de um único ACK cumulativo, reconhecendo ambos os segmentos na ordem.
Chegada de um segmento fora da ordem com número de seqüência mais alto do que o esperado. Lacuna detectada.	Envio imediato de um ACK duplicado, indicando número de seqüência do byte seguinte esperado (que é a extremidade mais baixa da lacuna).
Chegada de um segmento que preenche, parcial ou completamente, a lacuna nos dados recebidos.	Envio imediato de um ACK, contanto que o segmento comece na extremidade mais baixa da lacuna.

Tabela 3.2 Recomendação para geração de ACKs TCP [RFC 1122, RFC 2581]

Quando um TCP destinatário recebe um segmento com um número de seqüência que é maior do que o número de seqüência subseqüente, esperado e na ordem, ele detecta uma lacuna na corrente de dados — isto é, a falta de um segmento. Essa lacuna poderia ser o resultado de segmentos perdidos ou reordenados dentro da rede. Uma vez que o TCP não usa reconhecimentos negativos, o destinatário não pode enviar um reconhecimento negativo explícito de volta ao remetente. Em vez disso, ele simplesmente reconhece novamente (isto é, gera um ACK duplicado) o último byte de dados que recebeu na ordem. (Note que a Tabela 3.2 admite o caso em que o destinatário não descarta segmentos fora de ordem.)

Como um remetente quase sempre envia um grande número de segmentos, um atrás do outro, se um segmento for perdido, provavelmente existirão muitos ACKs duplicados, também um após o outro. Se o TCP remetente receber três ACKs duplicados para os mesmos dados, ele tomará isso como indicação de que o segmento que se seguiu ao segmento reconhecido três vezes foi perdido. (Nos exercícios de fixação consideraremos por que o remetente espera três ACKs duplicados e não apenas um.) No caso de receber três ACKs duplicados, o TCP remetente realiza uma **retransmissão rápida** [RFC 2581], retransmitindo o segmento que falta *antes* da expiração do temporizador do segmento. Para o TCP com retransmissão rápida, o seguinte trecho de codificação substitui o evento ACK recebido na Figura 3.33:

```
event: ACK received, with ACK field value of y
        if (y > SendBase) {
                SendBase=y
                if (there are currently any not yet
                        acknowledged segments)
                    start timer
                }
        else { /* a duplicate ACK for already ACKed
                segment */
            increment number of duplicate ACKs
                received for y
            if (number of duplicate ACKS received
                for y==3) {
                /* TCP fast retransmit */
                resend segment with sequence number y
                }
            break;
```

Observamos anteriormente que muitas questões sutis vêm à tona quando um mecanismo de controle de temporização/retransmissão é implementado em um protocolo real como o TCP. Os procedimentos acima, cuja evolução é resultado de mais de 15 anos de experiência com temporizadores TCP, deve convencê-lo de que, indubitavelmente, é isso que acontece!

Go-Back-N ou repetição seletiva?

Vamos encerrar nosso estudo do mecanismo de recuperação de erros do TCP considerando a seguinte pergunta: o TCP é um protocolo GBN ou SR? Lembre-se de que, no TCP, os reconhecimentos são cumulativos e segmentos corretamente recebidos, mas fora da ordem, não são reconhecidos (ACK) individualmente pelo destinatário. Conseqüentemente, como mostra a Figura 3.33 (veja também a Figura 3.19), o TCP remetente precisa tão-somente lembrar o menor número de seqüência de um byte transmitido mas não reconhecido (`SendBase`) e o número de seqüência do byte seguinte a ser enviado (`NextSeqNum`). Nesse sentido, o TCP se parece muito com um protocolo ao estilo do GBN. Porém, há algumas diferenças surpreendentes

entre o TCP e o GBN. Muitas implementações do TCP armazenarão segmentos recebidos corretamente, mas fora da ordem [Stevens, 1994]. Considere também o que acontece quando o remetente envia uma seqüência de segmentos 1, 2, ..., N e todos os segmentos chegam ao destinatário na ordem e sem erro. Além disso, suponha que o reconhecimento para o pacote $n < N$ se perca, mas que os $N - 1$ reconhecimentos restantes cheguem ao remetente antes do esgotamento de suas respectivas temporizações. Nesse exemplo, o GBN retransmitiria não somente o pacote n, mas também todos os pacotes subseqüentes $n + 1$, $n + 2$, ..., N. O TCP, por outro lado, retransmitiria no máximo um segmento, a saber, o segmento n. E mais, o TCP nem ao menos retransmitiria o segmento n se o reconhecimento para o segmento $n + 1$ chegasse antes do final da temporização para o segmento n.

Uma modificação proposta para o TCP, denominada **reconhecimento seletivo** [RFC 2018], permite que um destinatário TCP reconheça seletivamente segmentos fora de ordem, em vez de apenas reconhecer cumulativamente o último segmento recebido corretamente e na ordem. Quando combinado com retransmissão seletiva — isto é, saltar a retransmissão de segmentos que já foram reconhecidos seletivamente pelo destinatário —, o TCP se parece muito com nosso protocolo SR genérico. Assim, o mecanismo de recuperação de erros do TCP provavelmente é mais bem caracterizado como um híbrido dos protocolos GBN e SR.

3.5.5 Controle de fluxo

Lembre-se de que os hospedeiros de cada lado de uma conexão TCP reservam um buffer de recepção para a conexão. Quando a conexão TCP recebe bytes que estão corretos e em seqüência, ele coloca os dados no buffer de recepção. O processo de aplicação associado lerá os dados a partir desse buffer, mas não necessariamente no momento em que são recebidos. Na verdade, a aplicação receptora pode estar ocupada com alguma outra tarefa e nem ao menos tentar ler os dados até muito após a chegada deles. Se a aplicação for relativamente lenta na leitura dos dados, o remetente pode muito facilmente saturar o buffer de recepção da conexão por enviar demasiados dados muito rapidamente.

O TCP provê um **serviço de controle de fluxo** às suas aplicações, para eliminar a possibilidade de o remetente saturar o buffer do destinatário. Assim, controle de fluxo é um serviço de compatibilização de velocidades — compatibiliza a taxa à qual o remetente está enviando com a aquela à qual a aplicação receptora está lendo. Como notamos anteriormente, um TCP remetente também pode ser estrangulado devido ao congestionamento dentro da rede IP. Esse modo de controle do remetente é denominado **controle de congestionamento**, um tópico que será examinado detalhadamente nas seções 3.6 e 3.7. Mesmo que as ações executadas pelo controle de fluxo e pelo controle de congestionamento sejam similares (a regulagem do remetente), fica evidente que elas são executadas por razões muito diferentes. Infelizmente, muitos autores usam os termos de modo intercambiável, e o leitor esperto tem de tomar muito cuidado para distinguir os dois casos. Vamos agora discutir como o TCP provê seu serviço de controle de fluxo. Para podermos enxergar o quadro geral, sem nos fixarmos nos detalhes, nesta seção admitiremos que essa implementação do TCP descarta segmentos fora da ordem.

O TCP provê serviço de controle de fluxo fazendo com que o *remetente* mantenha uma variável denominada **janela de recepção**. Informalmente, a janela de recepção é usada para dar ao remetente uma idéia do espaço de buffer livre disponível no destinatário. Como o TCP é *full-duplex*, o remetente de cada lado da conexão mantém uma janela de recepção distinta. Vamos examinar a janela de recepção no contexto de uma transferência de arquivo. Suponha que o hospedeiro A esteja enviando um arquivo grande ao hospedeiro B por uma conexão TCP. O hospedeiro B aloca um buffer de recepção a essa conexão; denominemos seu tamanho `RcvBuffer`. De tempos em tempos, o processo de aplicação no hospedeiro B faz a leitura do buffer. São definidas as seguintes variáveis:

- `LastByteRead` = o número do último byte na cadeia de dados lido do buffer pelo processo de aplicação em B.

`LastByteRcvd` = o número do último byte na cadeia de dados que chegou da rede e foi colocado no buffer de recepção de B.

Como o TCP não tem permissão para saturar o buffer alocado, devemos ter:

```
LastByteRcvd - LastByteRead ≤ RcvBuffer
```

A janela de recepção, denominada `RcvWindow`, é ajustada para a quantidade de espaço disponível no buffer:

```
RcvWindow = RcvBuffer - [LastByteRcvd - LastByteRead]
```

Como o espaço disponível muda com o tempo, `RcvWindow` é dinâmica. A variável `RcvWindow` está ilustrada na Figura 3.37.

Como a conexão usa a variável `RcvWindow` para prover o serviço de controle de fluxo? O hospedeiro A diz ao hospedeiro B quanto espaço disponível ele tem no buffer da conexão colocando o valor corrente de `RcvWindow` no campo de janela de recepção de cada segmento que envia a B. Inicialmente, o hospedeiro B estabelece `RcvWindow = RcvBuffer`. Note que, para conseguir isso, o hospedeiro B deve monitorar diversas variáveis específicas da conexão.

O hospedeiro A, por sua vez, monitora duas variáveis, `LastByteSent` e `LastByteAcked`, cujos significados são óbvios. Note que a diferença entre essas duas variáveis, `LastByteSent - LastByteAcked`, é a quantidade de dados não reconhecidos que A enviou para a conexão. Mantendo a quantidade de dados não reconhecidos menor do que o valor de `RcvWindow`, o hospedeiro A tem certeza de que não está fazendo transbordar o buffer de recepção no hospedeiro B. Assim, o hospedeiro A tem de certificar-se, durante toda a duração da conexão, de que:

```
LastByteSent - LastByteAcked ≤ RcvWindow
```

Há um pequeno problema técnico com esse esquema. Para percebê-lo, suponha que o buffer de recepção do hospedeiro B fique tão cheio que `RcvWindow` =0. Após anunciar ao hospedeiro A que `RcvWindow` = 0, imagine que B não tenha *nada* para enviar ao hospedeiro A. Agora considere o que acontece. Enquanto o processo de aplicação em B esvazia o buffer, o TCP não envia novos segmentos com novos valores `RcvWindow` para o hospedeiro A. Na verdade, o TCP enviará um segmento ao hospedeiro A somente se tiver dados ou um reconhecimento para enviar. Por conseguinte, o hospedeiro A nunca será informado de que foi aberto algum espaço no buffer de recepção do hospedeiro B: ele ficará bloqueado e não poderá transmitir mais dados! Para resolver esse problema, a especificação do TCP requer que o hospedeiro A continue a enviar segmentos com um byte de dados quando a janela de recepção de B for zero. Esses segmentos serão

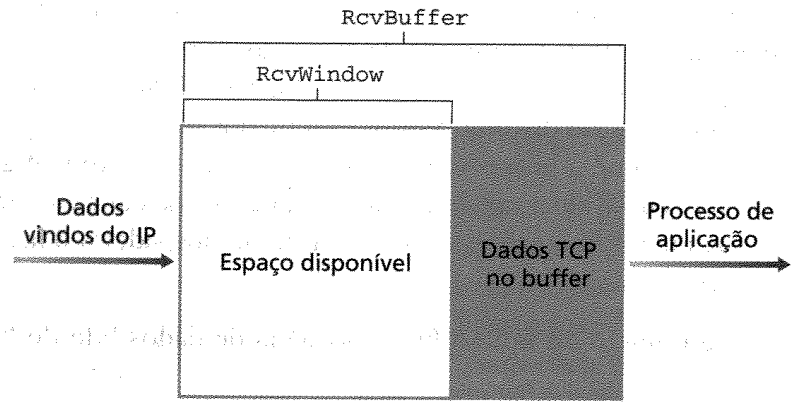


Figura 3.37 A janela de recepção (`RcvWindow`) e o buffer de recepção (`RcvBuffer`)

reconhecidos pelo destinatário. Inevitavelmente, o buffer começará a esvaziar e os reconhecimentos conterão um valor diferente de zero em RcvWindow.

O Companion Website do livro (http://www.aw.com/kurose_br) fornece um applet interativo em Java que ilustra a operação da janela de recepção do TCP.

Agora que descrevemos o serviço de controle de fluxo do TCP, mencionaremos brevemente que o UDP não provê controle de fluxo. Para entender a questão, considere o envio de uma série de segmentos UDP de um processo no hospedeiro A para um processo no hospedeiro B. Para uma implementação UDP típica, o UDP anexará os segmentos a um buffer de tamanho finito que 'precede' o socket correspondente (isto é, o socket para o processo). O processo lê um segmento inteiro do buffer por vez. Se o processo não ler os segmentos com rapidez suficiente, o buffer transbordará e os segmentos serão descartados.

3.5.6 Gerenciamento da conexão TCP

Nesta subseção, examinamos mais de perto como uma conexão TCP é estabelecida e encerrada. Embora esse tópico talvez não pareça particularmente interessante, é importante, porque o estabelecimento da conexão TCP tem um peso significativo nos atrasos percebidos (por exemplo, ao navegar pela Web). Além disso, muitos dos ataques mais comuns a redes — entre eles o incrivelmente popular ataque de inundação SYN — exploram vulnerabilidades no gerenciamento da conexão TCP. Em primeiro lugar, vamos ver como essa conexão é estabelecida. Suponha que um processo que roda em um hospedeiro (cliente) queira iniciar uma conexão com outro processo em outro hospedeiro (servidor). O processo de aplicação cliente primeiramente informa ao TCP cliente que quer estabelecer uma conexão com um processo no servidor. O TCP no cliente então estabelece uma conexão TCP com o TCP no servidor da seguinte maneira:

- **Etapa 1**. O lado cliente do TCP primeiramente envia um segmento TCP especial ao lado servidor do TCP. Esse segmento especial não contém nenhum dado de camada de aplicação, mas um dos bits de flag no seu cabeçalho (veja a Figura 3.29), o bit SYN, é ajustado para 1. Por essa razão, esse segmento é denominado um segmento SYN. Além disso, o cliente escolhe aleatoriamente um número de seqüência inicial (client_isn) e coloca esse número no campo de número de seqüência do segmento TCP SYN inicial. Esse segmento é encapsulado em um datagrama IP e enviado ao servidor. A aleatoriedade adequada da escolha de client_isn de modo a evitar certos ataques à segurança tem despertado considerável interesse [CERT 2001-09].
- **Etapa 2**. Assim que o datagrama IP contendo o segmento TCP SYN chega ao hospedeiro servidor (admitindo-se que ele realmente chegue!), o servidor extrai o segmento TCP SYN do datagrama, aloca buffers e variáveis TCP à conexão e envia um segmento de aceitação de conexão ao TCP cliente. (Veremos, no Capítulo 8, que a alocação desses buffers e variáveis, antes da conclusão da terceira etapa da apresentação de três vias, torna o TCP vulnerável a um ataque de recusa de serviço conhecido como inundação SYN.) Esse segmento de aceitação de conexão também não contém nenhum dado de camada de aplicação. Contudo, contém três informações importantes no cabeçalho do segmento: o bit SYN está com valor 1; o campo de reconhecimento do cabeçalho do segmento TCP está ajustado para client_isn+1; e, por fim, o servidor escolhe seu próprio número de seqüência inicial (server_isn) e coloca esse valor no campo de número de seqüência do cabeçalho do segmento TCP. Esse segmento de aceitação de conexão está dizendo, com efeito, "Recebi seu pacote SYN para começar uma conexão com seu número de seqüência inicial client_isn. Concordo em estabelecer essa conexão. Meu número de seqüência inicial é server_isn". O segmento de concessão da conexão às vezes é denominado **segmento SYNACK**.
- **Etapa 3**. Ao receber o segmento SYNACK, o cliente também reserva buffers e variáveis para a conexão. O hospedeiro cliente então envia ao servidor mais um segmento. Esse último segmento reconhece o segmento de confirmação da conexão do servidor (o cliente o faz colocando o valor server_isn+1 no campo de reconhecimento do cabeçalho do segmento TCP). O bit SYN é ajustado para 0, já que a conexão está estabelecida.

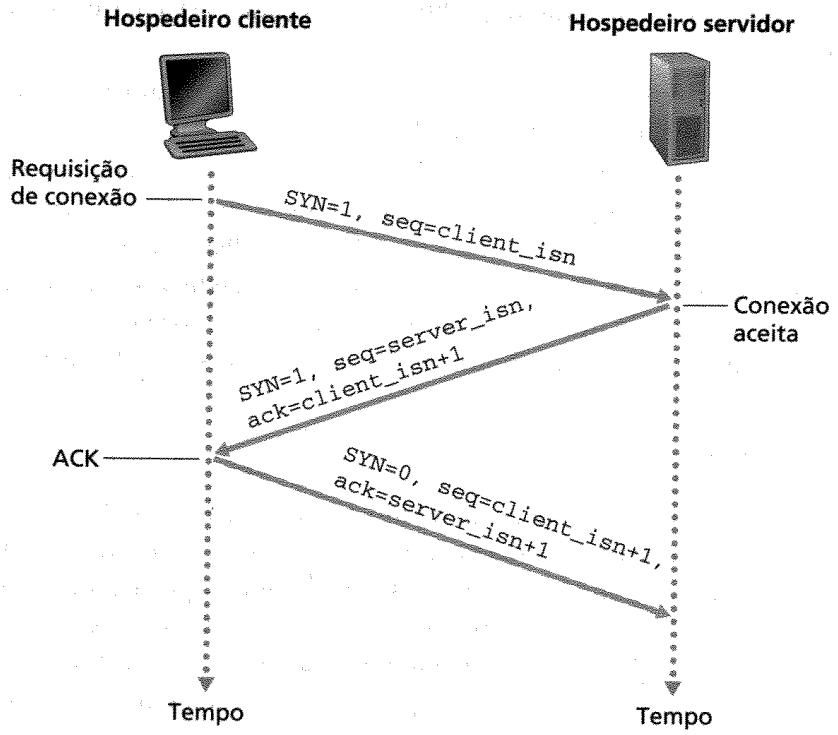


Figura 3.38 Apresentação de três vias do TCP: troca de segmentos

Completadas as três etapas, os hospedeiros cliente e servidor podem enviar segmentos contendo dados um ao outro. Em cada um desses futuros segmentos, o bit SYN estará ajustado para 0. Note que, para estabelecer a conexão, três pacotes são enviados entre dois hospedeiros, como ilustra a Figura 3.38. Por essa razão, esse procedimento de estabelecimento de conexão é freqüentemente denominado **apresentação de três vias**. Vários aspectos da apresentação de três vias do TCP são tratados nos exercícios ao final deste capítulo (Por que são necessários os números de seqüência iniciais? Por que é preciso uma apresentação de três vias, e não apenas de duas vias?) É interessante notar que um alpinista e seu amarrador (que fica mais abaixo e cuja tarefa é passar a corda de segurança ao alpinista) usa um protocolo de comunicação de apresentação de três vias idêntico ao do TCP para garantir que ambos os lados estejam prontos antes de o alpinista iniciar a escalada.

Tudo o que é bom dura pouco, e o mesmo é válido para uma conexão TCP. Qualquer um dos dois processos que participam de uma conexão TCP pode encerrar a conexão. Quando a conexão termina, os 'recursos' (isto é, os buffers e as variáveis) nos hospedeiros são liberados. Como exemplo, suponha que o cliente decida encerrar a conexão, como mostra a Figura 3.39. O processo de aplicação cliente emite um comando para fechar. Isso faz com que o TCP cliente envie um segmento TCP especial ao processo servidor, cujo bit de flag no cabeçalho do segmento, denominado bit FIN (veja a Figura 3.39), tem valor ajustado em 1. Quando o servidor recebe esse segmento, ele envia de volta ao cliente um segmento de reconhecimento. O servidor então envia seu próprio segmento de encerramento, que tem o bit FIN ajustado em 1. Por fim, o cliente reconhece o segmento de encerramento do servidor. Nesse ponto, todos os recursos dos dois hospedeiros estão liberados.

Durante a vida de uma conexão TCP, o protocolo TCP que roda em cada hospedeiro faz transições pelos vários **estados do TCP**. A Figura 3.40 ilustra uma seqüência típica de estados do TCP visitados pelo TCP cliente. O TCP cliente começa no estado CLOSED. A aplicação no lado cliente inicia uma nova conexão TCP (criando um objeto socket como nos exemplos em Java do Capítulo 2). Isso faz com que o TCP no cliente envie um segmento SYN ao TCP no servidor. Após ter enviado o segmento SYN, o TCP cliente entra no estado SYN_SENT e, enquanto isso, o TCP cliente espera por um segmento do TCP servidor que inclui um reconhecimento para o segmento anterior do cliente, e tem o bit SYN ajustado para o valor 1.

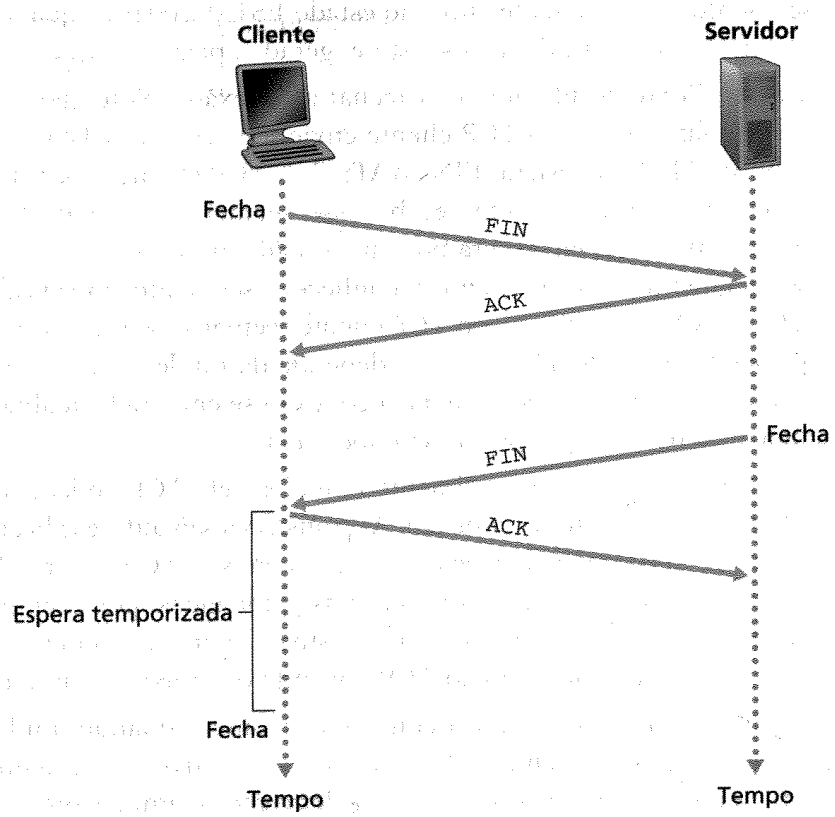


Figura 3.39 Encerramento de uma conexão TCP

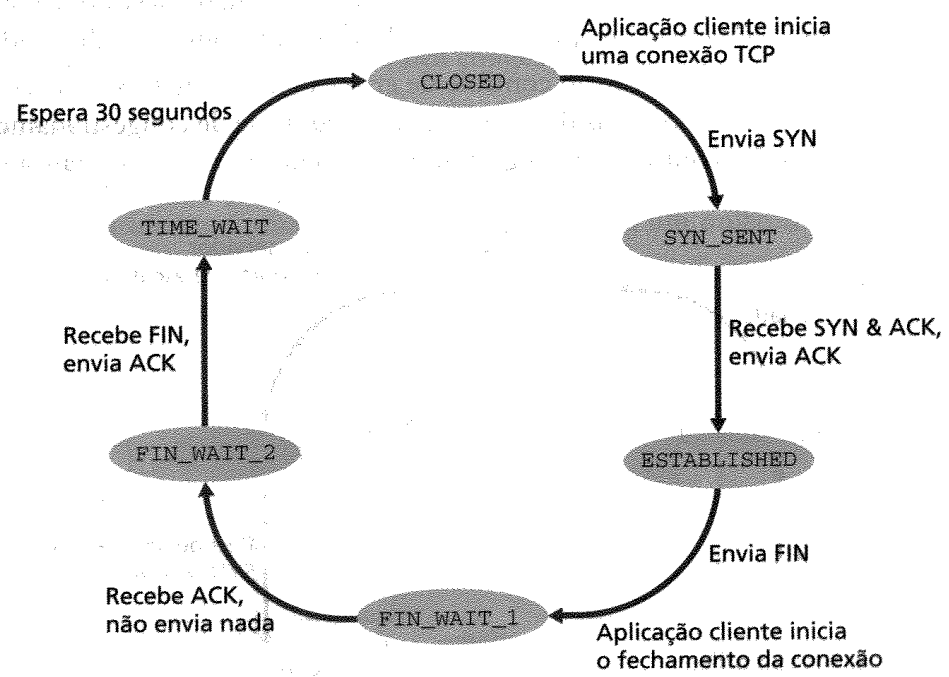


Figura 3.40 Uma seqüência típica de estados do TCP visitados por um TCP cliente

Assim que recebe esse segmento, o TCP cliente entra no estado ESTABLISHED, quando pode enviar e receber segmentos TCP que contêm carga útil de dados (isto é, gerados pela aplicação).

Suponha que a aplicação cliente decida que quer fechar a conexão. (Note que o servidor também tem a alternativa de fechá-la.) Isso faz com que o TCP cliente envie um segmento TCP com o bit FIN ajustado em 1 e entre no estado FIN_WAIT_1. No estado FIN_WAIT_1, o TCP cliente espera por um segmento TCP do servidor com um reconhecimento. Quando recebe esse segmento, o TCP cliente entra no estado FIN_WAIT_2. No estado FIN_WAIT_2, ele espera por outro segmento do servidor com o bit FIN ajustado para 1. Após receber esse segmento, o TCP cliente reconhece o segmento do servidor e entra no estado TIME_WAIT. O estado TIME_WAIT permite que o TCP cliente reenvie o reconhecimento final, caso o ACK seja perdido. O tempo passado no estado TIME_WAIT depende da implementação, mas os valores típicos são 30 segundos, 1 minuto e 2 minutos. Após a espera, a conexão se encerra formalmente e todos os recursos do lado cliente (inclusive os números de porta) são liberados.

A Figura 3.41 ilustra a série de estados normalmente visitados pelo TCP do lado servidor, admitindo-se que é o cliente quem inicia o encerramento da conexão. As transições são auto-explicativas. Nesses dois diagramas de transição de estados, mostramos apenas como uma conexão TCP é normalmente estabelecida e fechada. Não descrevemos o que acontece em certos cenários patológicos, por exemplo, quando ambos os lados de uma conexão querem fechar ao mesmo tempo. Se estiver interessado em aprender mais sobre esse assunto e sobre outros mais avançados referentes ao TCP, consulte o abrangente livro de [Stevens, 1994].

Antes de concluirmos esta seção, vamos considerar o que acontece quando um hospedeiro recebe um segmento TCP cujos números de porta ou endereço IP não são compatíveis com nenhuma das portas existentes no hospedeiro. Por exemplo, suponha que um hospedeiro receba um pacote TCP SYN com porta de destino 80, mas não está aceitando conexões nessa porta (isto é, não está rodando um servidor Web na porta 80). Então, ele enviará à fonte um segmento especial de reinicialização. Esse segmento TCP tem o bit de flag RST ajustado para 1 (veja Seção 3.5.2). Assim, quando um hospedeiro envia um segmento de reinicialização ele está dizendo à fonte: "Eu não tenho um socket para esse segmento. Favor não enviá-lo novamente". Quando um hospedeiro recebe um pacote UDP cujo número de porta de destino não é compatível com as portas de um UDP em curso, ele envia um datagrama ICMP especial, como será discutido no Capítulo 4.

Com isso, concluímos nossa introdução ao controle de erro e controle de fluxo em TCP. Voltaremos ao TCP na Seção 3.7 e então examinaremos mais detalhadamente o controle de congestionamento. Antes, contudo, vamos analisar a questão do controle de congestionamento em um contexto mais amplo.

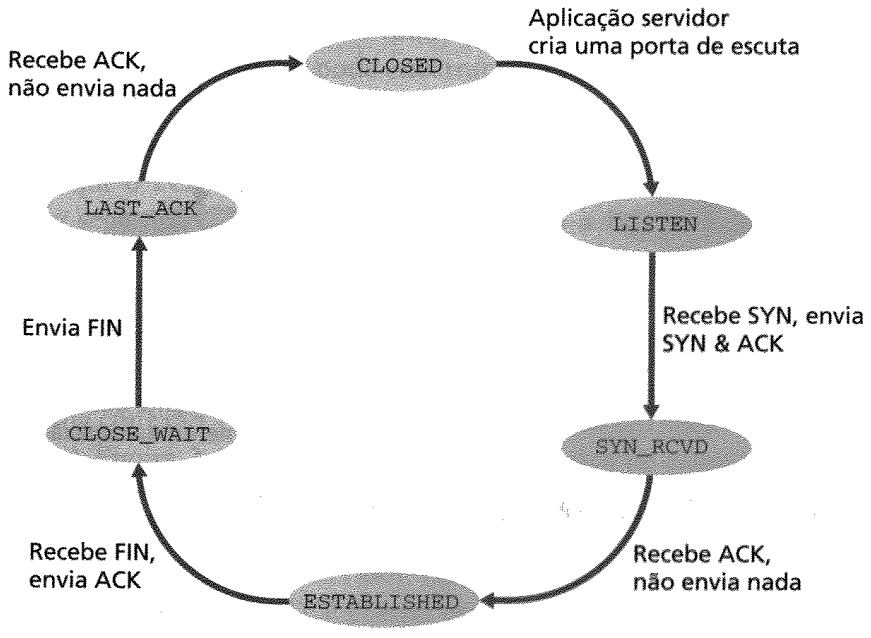


Figura 3.41 Uma seqüência típica de estados do TCP visitados por um TCP do lado do servidor

3.6 Princípios de controle de congestionamento

Nas seções anteriores, examinamos os princípios gerais e também os mecanismos específicos do TCP usados para prover um serviço de transferência confiável de dados em relação à perda de pacotes. Mencionamos anteriormente que, na prática, essa perda resulta, caracteristicamente, de uma saturação de buffers de roteadores à medida que a rede fica congestionada. Assim, a retransmissão de pacotes trata de um sintoma de congestionamento de rede (a perda de um segmento específico de camada de transporte), mas não trata da causa do congestionamento da rede: demasiadas fontes tentando enviar dados a uma taxa muito alta. Para tratar da causa do congestionamento de rede, são necessários mecanismos para regular os remetentes quando esse congestionamento ocorre.

Nesta seção, consideramos o problema do controle de congestionamento em um contexto geral, buscando entender por que o congestionamento é algo ruim, como o congestionamento de rede se manifesta no desempenho recebido por aplicações da camada superior e várias medidas que podem ser adotadas para evitar o congestionamento de rede ou reagir a ele. Esse estudo mais geral do controle de congestionamento é apropriado, já que, como acontece com a transferência confiável de dados, o congestionamento é um dos "dez mais" da lista de problemas fundamentalmente importantes no trabalho em rede. Concluímos esta seção com uma discussão sobre o controle de congestionamento no serviço de **taxa de bits disponível** (*available bit-rate* — ABR) em redes com **modo de transferência assíncrono** (ATM). A seção seguinte contém um estudo detalhado do algoritmo de controle de congestionamento do TCP.

3.6.1 As causas e os custos do congestionamento

Vamos começar nosso estudo geral do controle de congestionamento examinando três cenários de complexidade crescente nos quais ocorre o congestionamento. Em cada caso, examinaremos, primeiramente, por que ele ocorre e, depois, seu custo (no que se refere aos recursos não utilizados integralmente e ao baixo desempenho recebido pelos sistemas finais). Não focalizaremos (ainda) como reagir ao congestionamento, ou evitá-lo; preferimos estudar uma questão mais simples, que é entender o que acontece quando hospedeiros aumentam sua taxa de transmissão e a rede fica congestionada.

Cenário 1: dois remetentes, um roteador com buffers infinitos

Começamos considerando o que talvez seja o cenário de congestionamento mais simples possível: dois hospedeiros (A e B), cada um com uma conexão que compartilha um único trecho de rede entre a fonte e o destino, como mostra a Figura 3.42.

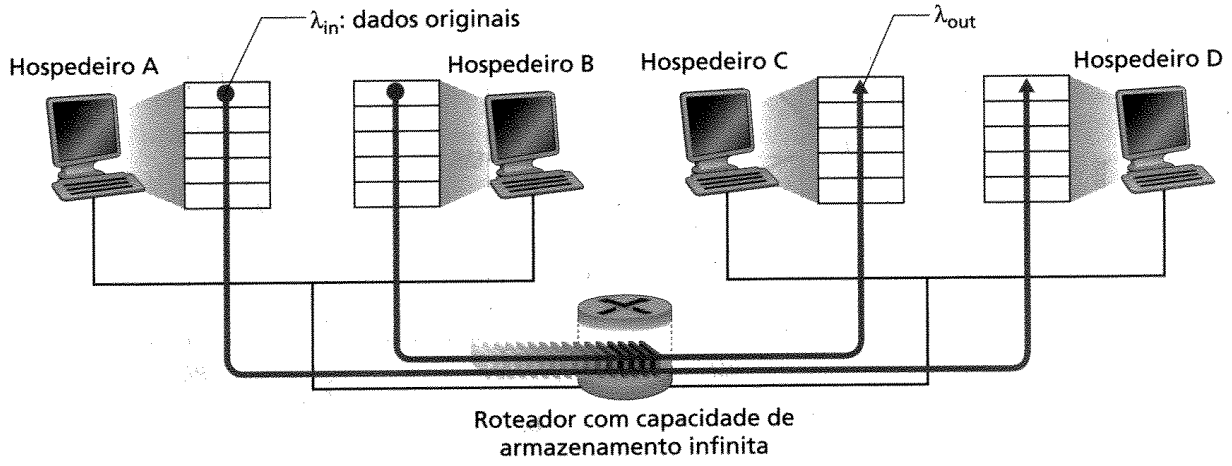


Figura 3.42 Cenário de congestionamento 1: duas conexões compartilhando um único roteador com número infinito de buffers

Vamos admitir que a aplicação no hospedeiro A esteja enviando dados para a conexão (por exemplo, passando dados para o protocolo de camada de transporte por um socket) a uma taxa média de λ_{in} bytes/segundo. Esses dados são originais no sentido de que cada unidade de dados é enviada para dentro do socket apenas uma vez. O protocolo de camada de transporte subjacente é simples. Os dados são encapsulados e enviados; não há recuperação de erros (por exemplo, retransmissão), controle de fluxo, nem controle de congestionamento. Desprezando a sobrecarga adicional causada pela adição de informações de cabeçalhos de camada de transporte e de camadas mais baixas, a taxa à qual o hospedeiro A oferece tráfego ao roteador nesse primeiro cenário é λ_{in} bytes/segundo. O hospedeiro B funciona de maneira semelhante, e admitimos, por simplicidade, que ele também esteja enviando dados a uma taxa de λ_{in} bytes/segundo. Os pacotes dos hospedeiros A e B passam por um roteador e por um enlace de saída compartilhado de capacidade *R*. O roteador tem buffers que lhe permitem armazenar os pacotes que chegam quando a taxa de chegada de pacotes excede a capacidade do enlace de saída. No primeiro cenário, admitimos que o roteador tenha capacidade de armazenamento infinita.

A Figura 3.43 apresenta o desempenho da conexão do hospedeiro A nesse primeiro cenário. O gráfico da esquerda dessa figura apresenta a **vazão por conexão** (número de bytes por segundo no destinatário) como uma função da taxa de envio da conexão. Para uma taxa de transmissão entre 0 e *R*/2, a vazão no destinatário é igual à velocidade de envio do remetente — tudo o que o remetente envia é recebido no destinatário com um atraso finito. Quando a velocidade de envio estiver acima de *R*/2, contudo, a vazão será somente *R*/2. Esse limite superior da vazão é conseqüência do compartilhamento da capacidade do enlace entre duas conexões. O enlace simplesmente não consegue entregar os pacotes a um destinatário com uma taxa em regime que exceda *R*/2. Não importa quão altas sejam as taxas de envio ajustadas nos hospedeiros A e B, eles jamais alcançarão uma vazão maior do que *R*/2.

Alcançar uma vazão de *R*/2 por conexão pode até parecer uma coisa boa, pois o enlace está sendo integralmente utilizado para entregar pacotes no destinatário. No entanto, o gráfico do lado direito da Figura 3.43 mostra as conseqüências de operar próximo à capacidade máxima do enlace. À medida que a taxa de envio se aproxima de *R*/2 (partindo da esquerda), o atraso médio fica cada vez maior. Quando a taxa de envio ultrapassa *R*/2, o número médio de pacotes na fila no roteador é ilimitado e o atraso médio entre a fonte e o destino se torna infinito (admitindo que as conexões operem a essas velocidades de transmissão durante um período de tempo infinito e que a capacidade de armazenamento também seja infinita). Assim, embora operar a uma vazão agregada próxima a *R* possa ser ideal do ponto de vista da vazão, está bem longe de ser ideal do ponto de vista do atraso. *Mesmo nesse cenário (extremamente) idealizado, já descobrimos um custo da rede congestionada — há grandes atrasos de fila quando a taxa de chegada de pacotes se aproxima da capacidade do enlace.*

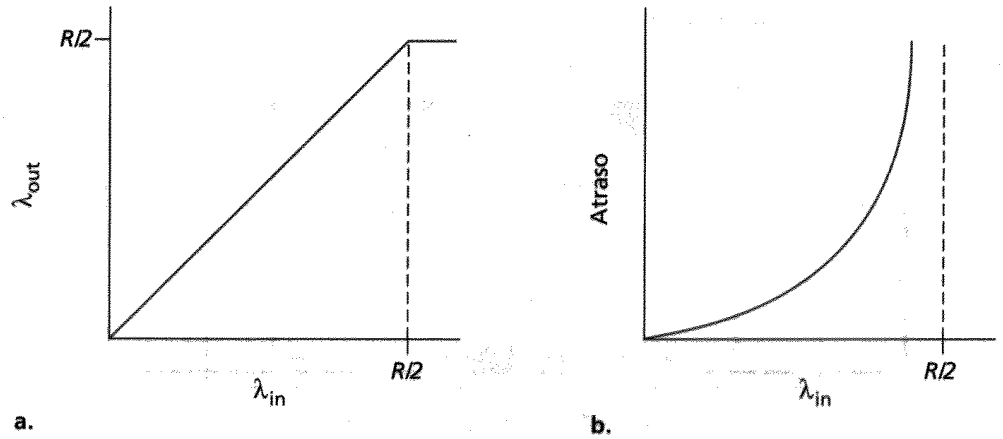


Figura 3.43 Cenário 1: vazão e atraso em função da taxa de envio do hospedeiro

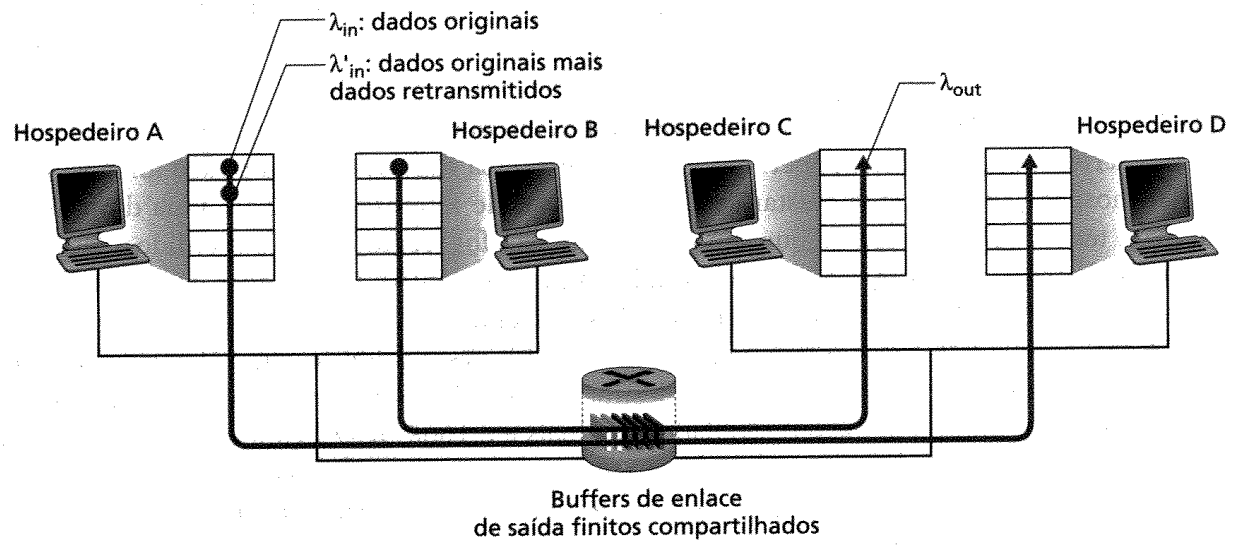


Figura 3.44 Cenário 2: dois hospedeiros (com retransmissões) e um roteador com buffers finitos

Cenário 2: dois remetentes, um roteador com buffers finitos

Vamos agora modificar ligeiramente o cenário 1 dos dois modos seguintes (veja a Figura 3.44). Em primeiro lugar, admitamos que a capacidade de armazenamento do roteador seja finita. Em uma situação real, essa suposição teria como conseqüência o descarte de pacotes que chegam a um buffer que já está cheio. Em segundo lugar, admitamos que cada conexão seja confiável. Se um pacote contendo um segmento de camada de transporte for descartado no roteador, o remetente por fim o retransmitirá. Como os pacotes podem ser retransmitidos, agora temos de ser mais cuidadosos com o uso da expressão "taxa de envio". Especificamente, vamos novamente designar a taxa com que a aplicação envia dados originais para dentro do socket como λ_{in} bytes/segundo. A taxa com que a camada de transporte envia segmentos (contendo dados originais *e* dados retransmitidos) para dentro da rede será denominada λ'_{in} bytes/segundo. Essa taxa (λ'_{in}) às vezes é denominada **carga oferecida** à rede.

O desempenho obtido no cenário 2 agora dependerá muito de como a retransmissão é realizada. Primeiramente, considere o caso não realista em que o hospedeiro A consiga, de algum modo (fazendo mágica!), determinar se um buffer do roteador está livre ou não. Assim, o hospedeiro A envia um pacote somente quando o buffer estiver livre. Nesse caso, não ocorreria nenhuma perda, λ_{in} seria igual a λ'_{in} e a vazão da conexão seria igual a λ_{in}. Esse caso é mostrado pela curva superior da Figura 3.45(a).

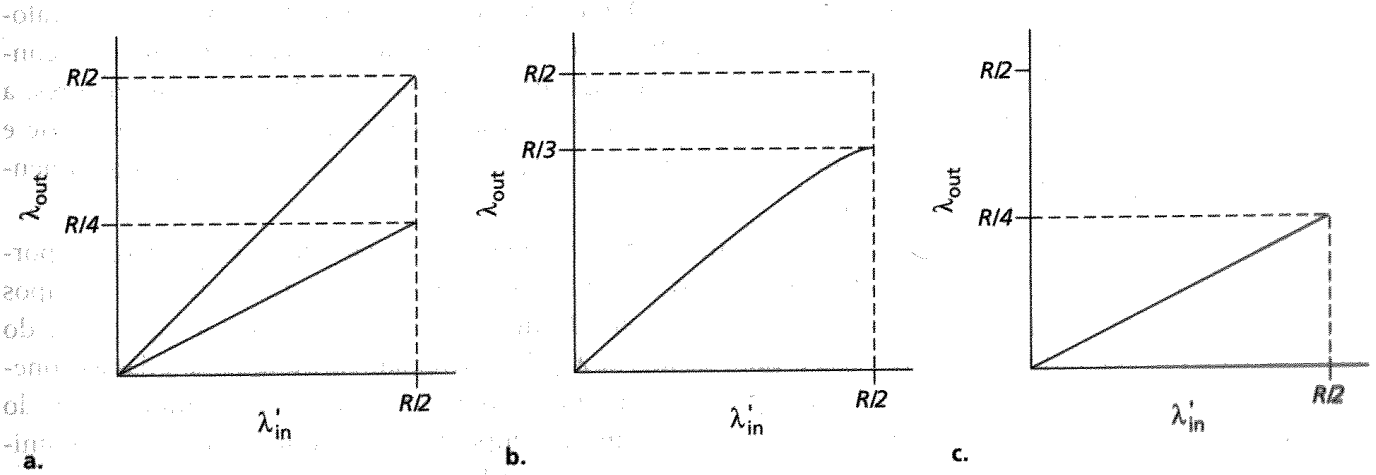


Figura 3.45 Desempenho no cenário 2 com buffers finitos

Do ponto de vista da vazão, o desempenho é ideal — tudo o que é enviado é recebido. Note que, nesse cenário, a taxa média de envio do hospedeiro não pode ultrapassar $R/2$, já que admitimos que nunca ocorre perda de pacote.

Considere, em seguida, o caso um pouco mais realista em que o remetente retransmite somente quando sabe, com certeza, que o pacote foi perdido. (Novamente, essa suposição é um pouco forçada. Contudo, é possível ao hospedeiro remetente ajustar seu temporizador de retransmissão para uma duração longa o suficiente para ter razoável certeza de que um pacote que não foi reconhecido foi perdido.) Nesse caso, o desempenho pode ser parecido com o que é mostrado na Figura 3.45(b). Para avaliar o que está acontecendo aqui, considere o caso em que a carga oferecida, λ_{in} (a taxa de transmissão dos dados originais mais as retransmissões) é igual a $R/2$. De acordo com a Figura 3.45(b), nesse valor de carga oferecida, a velocidade com a qual os dados são entregues à aplicação do destinatário é $R/3$. Assim, de $0{,}5R$ unidade de dados transmitida, $0{,}333R$ byte/segundo (em média) são dados originais e $0{,}166R$ byte/segundo (em média) são dados retransmitidos. *Observamos aqui outro custo de uma rede congestionada — o remetente deve realizar retransmissões para compensar os pacotes descartados (perdidos) devido ao esgotamento do buffer.*

Finalmente, vamos considerar o caso em que a temporização do remetente se esgote prematuramente e ele retransmita um pacote que ficou atrasado na fila, mas que ainda não está perdido. Aqui, tanto o pacote de dados original quanto a retransmissão podem alcançar o destinatário. É claro que o destinatário precisa apenas de uma cópia desse pacote e descartará a retransmissão. Nesse caso, o trabalho realizado pelo roteador ao repassar a cópia retransmitida do pacote original é desperdiçado, pois o destinatário já terá recebido a cópia original desse pacote. Em vez disso, seria melhor o roteador usar a capacidade de transmissão do enlace para enviar um pacote diferente. *Eis aqui mais um custo da rede congestionada — retransmissões desnecessárias feitas pelo remetente em face de grandes atrasos podem fazer com que um roteador use sua largura de banda de enlace para repassar cópias desnecessárias de um pacote.* A Figura 3.45(c) mostra a vazão *versus* a carga oferecida admitindo-se que cada pacote seja enviado (em média) duas vezes pelo roteador. Visto que cada pacote é enviado duas vezes, a vazão terá um valor assintótico de $R/4$ à medida que a carga oferecida se aproximar de $R/2$.

Cenário 3: quatro remetentes, roteadores com buffers finitos e trajetos com múltiplos roteadores

Em nosso cenário final de congestionamento, quatro hospedeiros transmitem pacotes sobre trajetos sobrepostos que apresentam dois saltos, como ilustrado na Figura 3.46. Novamente admitamos que cada hospedeiro use um mecanismo de temporização/retransmissão para implementar um serviço de transferência confiável de dados, que todos os hospedeiros tenham o mesmo valor de λ_{in} e que todos os enlaces dos roteadores tenham capacidade de R bytes/segundo.

Vamos considerar a conexão do hospedeiro A ao hospedeiro C que passa pelos roteadores R1 e R2. A conexão A–C compartilha o roteador R1 com a conexão D–B e o roteador 2 com a conexão B–D. Para valores extremamente pequenos de λ_{in}, esgotamentos de buffers são raros (como acontecia nos cenários de congestionamento 1 e 2) e a vazão é quase igual à carga oferecida. Para valores de λ_{in} ligeiramente maiores, a vazão correspondente é também maior, pois mais dados originais estão sendo transmitidos para a rede e entregues no destino, e os esgotamentos ainda são raros. Assim, para valores pequenos de λ_{in}, um aumento em λ_{in} resulta em um aumento em λ_{out}.

Como já analisamos o caso de tráfego extremamente baixo, vamos examinar aquele em que λ_{in} (e, portanto, λ'_{in}) é extremamente alto. Considere o roteador R2. O tráfego A–C que chega ao roteador R2 (após ter sido repassado de R1) pode ter uma taxa de chegada em R2 de, no máximo, R, que é a capacidade do enlace de R1 a R2, não importando qual seja o valor de λ_{in}. Se λ'_{in} for extremamente alto para todas as conexões (incluindo a conexão B–D), então a taxa de chegada do tráfego B–D em R2 poderá ser muito maior do que a taxado tráfego A–C. Como os tráfegos A–C e B–D têm de competir no roteador R2 pelo espaço limitado de buffer, a quantidade de tráfego A–C que consegue passar por R2 (isto é, que não se perde devido ao congestionamento de buffer) diminui cada vez mais à medida que a carga oferecida de B–D vai ficando maior. No limite, quando a carga oferecida se aproxima do infinito, um buffer vazio em R2 é imediatamente

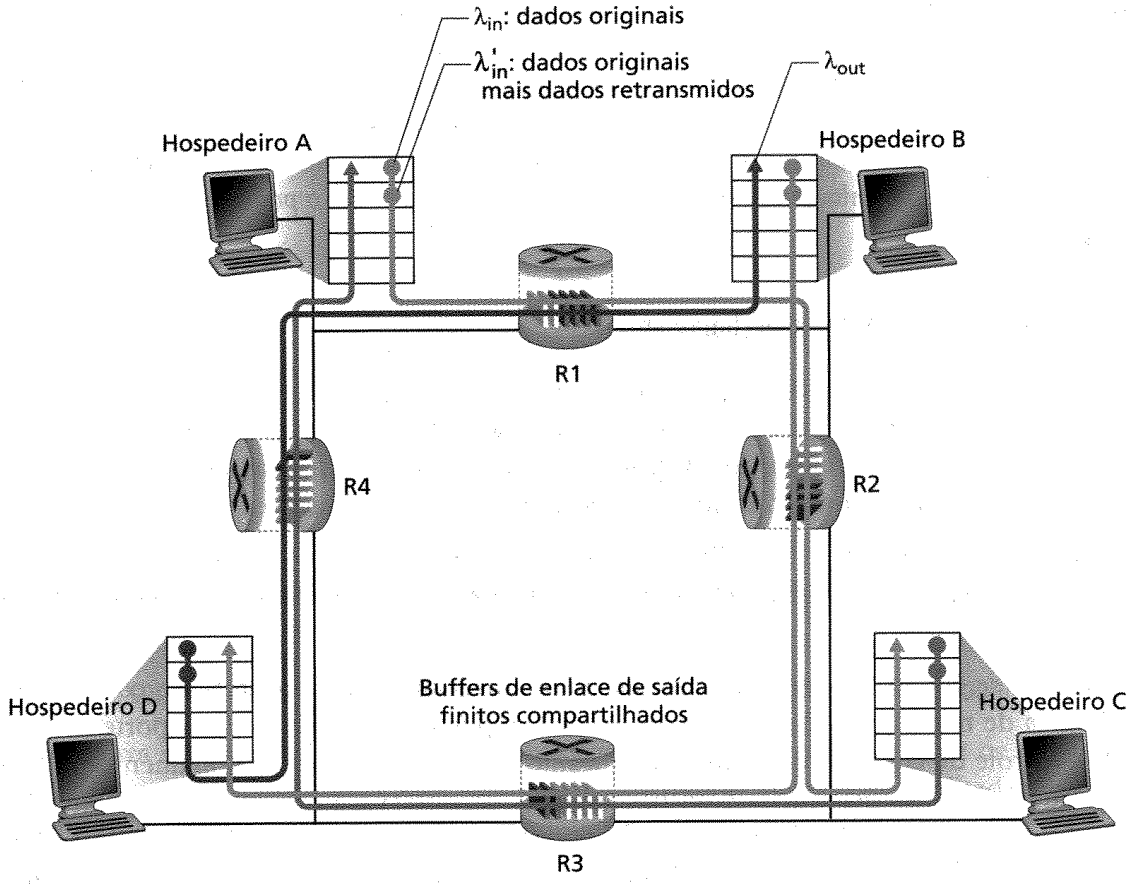


Figura 3.46 Quatro remetentes, roteadores com buffers finitos e trajetos com vários saltos

preenchido por um pacote B–D e a vazão da conexão A–C em R2 cai a zero. Isso, por sua vez, *implica que a vazão fim-a-fim de A–C vai a zero* no limite de tráfego pesado. Essas considerações dão origem ao comportamento da carga oferecida *versus* a vazão mostrada na Figura 3.47.

A razão para o eventual decréscimo da vazão com o crescimento da carga oferecida é evidente quando consideramos a quantidade de trabalho desperdiçado realizado pela rede. No cenário de alto tráfego que acabamos de descrever, sempre que um pacote é descartado em um segundo roteador, o trabalho realizado pelo primeiro para enviar o pacote ao segundo acaba sendo 'desperdiçado'. A rede teria se saído igualmente bem (melhor dizendo, igualmente mal) se o primeiro roteador tivesse simplesmente descartado aquele pacote e

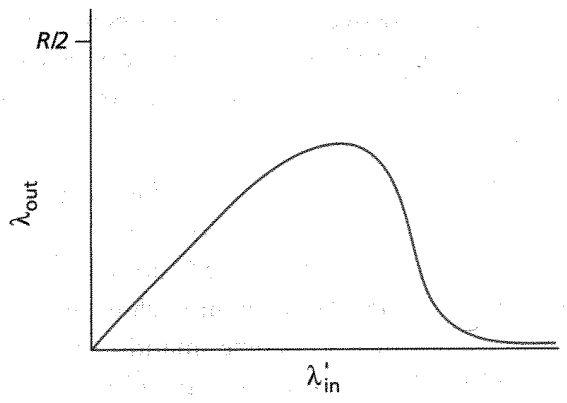


Figura 3.47 Desempenho obtido no cenário 3, com buffers finitos e trajetos com múltiplos roteadores

tivesse ficado inativo. Mais objetivamente, a capacidade de transmissão utilizada no primeiro roteador para enviar o pacote ao segundo teria sido maximizada para transmitir um pacote diferente. (Por exemplo, ao selecionar um pacote para transmissão, seria melhor que um roteador desse prioridade a pacotes que já atravessaram alguns roteadores anteriores.) *Portanto, vemos aqui mais um custo, o do descarte de pacotes devido ao congestionamento — quando um pacote é descartado ao longo de um caminho, a capacidade de transmissão que foi usada em cada um dos enlaces anteriores para repassar o pacote até o ponto em que foi descartado acaba sendo desperdiçada.*

3.6.2 Mecanismos de controle de congestionamento

Na Seção 3.7, examinaremos detalhadamente os mecanismos específicos do TCP para o controle de congestionamento. Nesta subseção, identificaremos os dois procedimentos mais comuns adotados, na prática, para o controle de congestionamento. Além disso, examinaremos arquiteturas específicas de rede e protocolos de controle de congestionamento que incorporam esses procedimentos.

No nível mais amplo, podemos distinguir mecanismos de controle de congestionamento conforme a camada de rede ofereça ou não assistência explícita à camada de transporte com a finalidade de controle de congestionamento:

- *Controle de congestionamento fim-a-fim.* Nesse método para o controle de congestionamento, a camada de rede não fornece nenhum suporte explícito à camada de transporte com a finalidade de controle de congestionamento. Até mesmo a presença de congestionamento na rede deve ser intuída pelos sistemas finais com base apenas na observação do comportamento da rede (por exemplo, perda de pacotes e atraso). Veremos na Seção 3.7 que o TCP deve necessariamente adotar esse método fim-a-fim para o controle de congestionamento, uma vez que a camada IP não fornece realimentação de informações aos sistemas finais quanto ao congestionamento da rede. A perda de segmentos TCP (apontada por uma temporização ou por três reconhecimentos duplicados) é tomada como indicação de congestionamento, e o TCP reduz o tamanho da janela de acordo com isso. Veremos também que as novas propostas para o TCP usam valores de atraso de viagem de ida e volta crescentes como indicadores de aumento do congestionamento da rede.
- *Controle de congestionamento assistido pela rede.* Com esse método, os componentes da camada de rede (isto é, roteadores) fornecem realimentação específica de informações ao remetente a respeito do estado de congestionamento na rede. Essa realimentação pode ser tão simples como um único bit indicando o congestionamento em um enlace. Adotada nas primeiras arquiteturas de rede IBM SNA [Schwartz, 1982] e DEC DECnet [Jain, 1989; Ramakrishnan, 1990], essa abordagem foi proposta recentemente para redes TCP/IP [Floyd TCP, 1994; RFC 2481]; ela é usada também no controle de congestionamento em ATM com serviço de transmissão ABR, como discutido a seguir. A realimentação mais sofisticada de rede também é possível. Por exemplo, uma forma de controle de congestionamento ATM ABR que estudaremos mais adiante permite que um roteador informe explicitamente ao remetente a velocidade de transmissão que ele (o roteador) pode suportar em um enlace de saída.

Para controle de congestionamento assistido pela rede, a informação sobre congestionamento é normalmente realimentada da rede para o remetente por um de dois modos, como mostra a Figura 3.48. Realimentação direta pode ser enviada de um roteador de rede ao remetente. Esse modo de notificação normalmente toma a forma de um **pacote de congestionamento** (*choke packet*) (que, em essência, diz: "Estou congestionado!"). O segundo modo de notificação ocorre quando um roteador marca/atualiza um campo em um pacote que está fluindo do remetente ao destinatário para indicar congestionamento. Ao receber um pacote marcado, o destinatário informa ao remetente a indicação de congestionamento. Note que esse último modo de notificação leva, no mínimo, o tempo total de uma viagem de ida e volta.

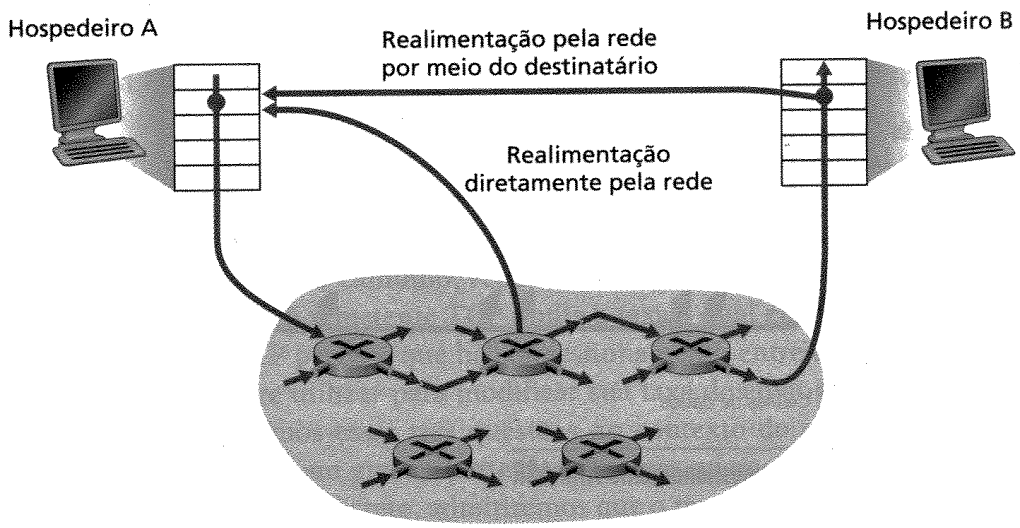


Figura 3.48 Dois caminhos de realimentação para informação sobre congestionamento indicado pela rede

3.6.3 Exemplo de controle de congestionamento assistido pela rede: controle de congestionamento ATM ABR

Concluímos esta seção com um breve estudo de caso do algoritmo de controle de congestionamento ATM ABR. Salientamos que nossa meta aqui *não* é, de modo algum, descrever detalhadamente aspectos da arquitetura ATM, mas mostrar um protocolo que adota uma abordagem de controle de congestionamento notavelmente diferente da adotada pelo protocolo TCP da Internet. Na verdade, apresentamos a seguir apenas os poucos aspectos da arquitetura ATM necessários para entender o controle de congestionamento ABR.

Fundamentalmente, o ATM adota uma abordagem orientada para circuito virtual (CV) da comutação de pacotes. Lembre, da nossa discussão no Capítulo 1, que isso significa que cada comutador no caminho entre a fonte e o destino manterá estado sobre o CV entre a fonte e o destino. Esse estado por CV permite que um comutador monitore o comportamento de remetentes individuais (por exemplo, monitorando sua taxa média de transmissão) e realize ações específicas de controle de congestionamento (tais como sinalizar explicitamente ao remetente para que ele reduza sua taxa quando o comutador fica congestionado). Esse estado por CV em comutadores de rede torna o ATM idealmente adequado para realizar controle de congestionamento assistido pela rede.

O ABR foi projetado como um serviço de transferência de dados elástico que faz lembrar, de certo modo, o TCP. Quando a rede está vazia, o serviço ABR deve ser capaz de aproveitar a vantagem da largura de banda disponível. Quando a rede está congestionada, deve limitar sua taxa de transmissão a algum valor mínimo predeterminado de taxa de transmissão. Um tutorial detalhado sobre controle de congestionamento e gerenciamento de tráfego ATM ABR é fornecido por [Jain, 1996].

A Figura 3.49 mostra a estrutura do controle de congestionamento para ATM ABR. Nessa discussão, adotaremos a terminologia ATM (por exemplo, usaremos o termo 'comutador' em vez de 'roteador' e o termo 'célula' em vez de 'pacote'). Com o serviço ATM ABR, as células de dados são transmitidas de uma fonte a um destino por meio de uma série de comutadores intermediários. Intercaladas às células de dados estão as **células de gerenciamento de recursos**, ou **células RM** (*resource-management cells*); essas células podem ser usadas para portar informações relacionadas ao congestionamento entre hospedeiros e comutadores. Quando uma célula RM chega a um destinatário, ela será 'virada' e enviada de volta ao remetente (possivelmente após o destinatário ter modificado o conteúdo dela). Também é possível que um comutador gere, ele mesmo, uma célula RM e a envie diretamente a uma fonte. Assim, células RM podem ser usadas para fornecer realimentação diretamente da rede e também realimentação da rede por intermédio do destinatário, como mostra a Figura 3.49.

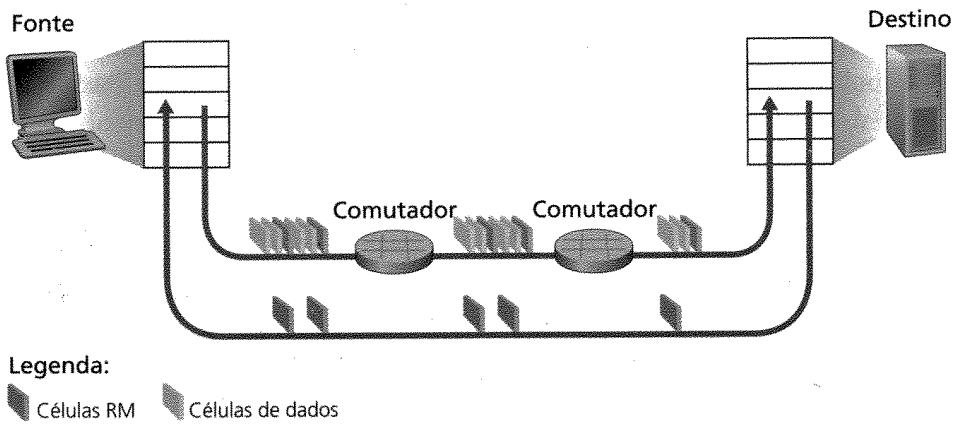


Figura 3.49 Estrutura de controle de congestionamento ATM ABR

O controle de congestionamento ATM ABR é um método baseado em taxa. O remetente estima explicitamente uma taxa máxima na qual pode enviar e se auto-regula de acordo com ela. O serviço ABR provê três mecanismos para sinalizar informações relacionadas a congestionamento dos comutadores ao destinatário:

- *Bit EFCI.* Cada *célula de dados* contém um **bit EFCI de indicação explícita de congestionamento** à frente (*explicit forward congestion indication*). Um comutador de rede congestionado pode modificar o bit EFCI dentro de uma célula para 1, a fim de sinalizar congestionamento ao hospedeiro destinatário, que deve verificar o bit EFCI em todas as células de dados recebidas. Quando uma célula RM chega ao destinatário, se a célula de dados recebida mais recentemente tiver o bit EFCI com valor 1, o destinatário modifica o bit de indicação de congestionamento (o bit CI) da célula RM para 1 e envia a célula RM de volta ao remetente. Usando o bit EFCI em células de dados e o bit CI em células RM, um remetente pode ser notificado sobre congestionamento em um comutador da rede.
- *Bits CI e NI.* Como observado anteriormente, células RM remetente/destinatário estão intercaladas com células de dados. A taxa de intercalação de células RM é um parâmetro ajustável, sendo uma célula RM a cada 32 células de dados o valor default. Essas células RM têm um bit **CI** e um bit **NI** (*no increase* — não aumentar) que podem ser ajustados por um comutador de rede congestionado. Especificamente, um comutador pode modificar para 1 o bit NI de uma célula RM que está passando quando a condição de congestionamento é leve e pode modificar o bit CI para 1 sob severas condições de congestionamento. Quando um hospedeiro destinatário recebe uma célula RM, ele a enviará de volta ao remetente com seus bits CI e NI inalterados (exceto que o CI pode ser ajustado para 1 pelo destinatário como resultado do mecanismo EFCI descrito antes).
- *Ajuste de ER.* Cada célula RM contém também um **campo ER** (*explicit rate* — taxa explícita) de dois bytes. Um comutador congestionado pode reduzir o valor contido no campo ER de uma célula RM que está passando. Desse modo, o campo ER será ajustado para a taxa mínima suportável por todos os comutadores no trajeto fonte–destino.

Uma fonte ATM ABR ajusta a taxa na qual pode enviar células como uma função dos valores de CI, NI e ER de uma célula RM devolvida. As regras para fazer esse ajuste de taxa são bastante complicadas e um tanto tediosas. O leitor poderá consultar [Jain, 1996] se quiser saber mais detalhes.

3.7 Controle de congestionamento TCP

Nesta seção, voltamos ao estudo do TCP. Como aprendemos na Seção 3.5, o TCP provê um serviço de transferência confiável entre dois processos que rodam em hospedeiros diferentes. Outro componente extre-

mamente importante do TCP é seu mecanismo de controle de congestionamento. Como indicamos na seção anterior, o TCP deve usar controle de congestionamento fim-a-fim em vez de controle de congestionamento assistido pela rede, já que a camada IP não fornece aos sistemas finais realimentação explícita relativa ao congestionamento da rede.

A abordagem adotada pelo TCP é obrigar cada remetente a limitar a taxa à qual enviam tráfego para sua conexão como uma função do congestionamento de rede percebido. Se um remetente TCP perceber que há pouco congestionamento no caminho entre ele e o destinatário, aumentará sua taxa de envio; se perceber que há congestionamento ao longo do caminho, reduzirá sua taxa de envio. Mas essa abordagem levanta três questões. A primeira é como um remetente TCP limita a taxa à qual envia tráfego para sua conexão? A segunda é como um remetente TCP percebe que há congestionamento entre ele e o destinatário? E a terceira, que algoritmo o remetente deve utilizar para modificar sua taxa de envio como uma função do congestionamento fim-a-fim? Examinaremos essas três questões no contexto do algoritmo Reno de controle de congestionamento TCP, que é usado em grande parte dos sistemas operacionais modernos [Padhye, 2001]. Para que a discussão fique no campo concreto, admitiremos que o remetente TCP esteja enviando um arquivo grande.

Em primeiro lugar, vamos examinar como um remetente TCP limita a taxa de envio à qual envia tráfego para sua conexão. Na Seção 3.5, vimos que cada lado de uma conexão TCP consiste em um buffer de recepção, um buffer de envio e diversas variáveis (`LastByteRead`, `RcvWindow` e assim por diante). O mecanismo de controle de congestionamento faz com que cada lado de uma conexão monitore uma variável adicional, a **janela de congestionamento**. A janela de congestionamento, denominada `CongWin`, impõe uma limitação à taxa à qual um remetente TCP pode enviar tráfego para dentro da rede. Especificamente, a quantidade de dados não reconhecidos em um hospedeiro não pode exceder o mínimo de `CongWin` e `RcvWindow`, ou seja:

```
LastByteSent − LastByteAcked ≤ min{CongWin, RcvWindow}
```

Para focalizar a discussão no controle de congestionamento (ao contrário do controle de fluxo), daqui em diante vamos admitir que o buffer de recepção TCP seja tão grande que a limitação da janela de recepção pode ser desprezada; assim, a quantidade de dados não reconhecidos no remetente estará limitada exclusivamente por `CongWin`.

A restrição acima limita a quantidade de dados não reconhecidos no remetente e, por conseguinte, limita indiretamente a taxa de envio do remetente. Para entender melhor, considere uma conexão na qual perdas e atrasos de transmissão de pacotes sejam desprezíveis. Então, em linhas gerais, no início de cada RTT a limitação permite que o remetente envie `CongWin` bytes de dados para a conexão; ao final do RTT, o remetente recebe reconhecimentos para os dados. *Assim, a taxa de envio do remetente é aproximadamente CongWin/RTT bytes por segundo. Portanto, ajustando o valor de* `CongWin`, *o remetente pode ajustar a taxa à qual envia dados para sua conexão.*

Em seguida, vamos considerar como um remetente TCP percebe que há congestionamento no caminho entre ele e o destino. Definimos "evento de perda" em um remetente TCP como a ocorrência de um esgotamento de temporização ou do recebimento de três ACKs duplicados do destinatário (lembre-se da nossa discussão, na Seção 3.5.4, do evento de temporização apresentado na Figura 3.33 e da subseqüente modificação para incluir transmissão rápida quando do recebimento de três ACKs duplicados). Quando há congestionamento excessivo, então um (ou mais) buffers de roteadores ao longo do caminho transborda, fazendo com que um datagrama (contendo um segmento TCP) seja descartado. Esse datagrama descartado, por sua vez, resulta em um evento de perda no remetente — ou um esgotamento de temporização ou o recebimento de três ACKs duplicados — que é tomado por ele como uma indicação de congestionamento no caminho remetente-destinatário.

Já consideramos como é detectado o congestionamento; agora vamos considerar o caso mais otimista de uma rede livre de congestionamento, isto é, quando não ocorre um evento de perda. Nesse caso, o TCP remetente receberá reconhecimentos para segmentos anteriormente não reconhecidos. Como veremos, o TCP considerará a chegada desses reconhecimentos como uma indicação de que tudo está bem — os segmentos que estão sendo transmitidos para a rede estão sendo entregues com sucesso no destinatário — e usará reconhecimentos para aumentar o tamanho de sua janela de congestionamento (e, por conseguinte, sua taxa de transmissão). Note que, se os reconhecimentos chegarem ao remetente a uma taxa relativamente baixa (por exemplo, se o atraso no caminho fim-a-fim for alto ou se nele houver um enlace de baixa largura de banda), então a janela de congestionamento será aumentada a uma taxa relativamente baixa. Por outro lado, se os reconhecimentos chegarem a uma taxa alta, então a janela de congestionamento será aumentada mais rapidamente. Como o TCP utiliza reconhecimentos para acionar (ou regular) o aumento de tamanho de sua janela de congestionamento, diz-se que o TCP é **auto-regulado**.

Agora estamos prontos para considerar os detalhes do algoritmo que um remetente TCP usa para regular sua taxa de envio como uma função do congestionamento percebido. Esse algoritmo é o famoso **algoritmo de controle de congestionamento TCP**. O algoritmo tem três componentes importantes: (1) aumento aditivo, diminuição multiplicativa, (2) partida lenta e (3) reação a eventos de esgotamento de temporização.

Aumento aditivo, diminuição multiplicativa

A idéia básica por trás do controle de congestionamento TCP é o remetente reduzir sua taxa de envio (reduzindo o tamanho de sua janela de congestionamento, `CongWin`) quando ocorre um evento de perda. Visto que outras conexões TCP que estão passando pelos mesmos roteadores congestionados provavelmente também estarão sofrendo eventos de perda, também elas provavelmente reduzirão suas taxas de envio reduzindo seus próprios valores de `CongWin`. O efeito geral, então, será o seguinte: as fontes cujos caminhos passam por roteadores congestionados reduzirão a taxa à qual enviam dados para a rede, o que, por sua vez, deverá aliviar o congestionamento nos roteadores congestionados. Mas, quanto um remetente TCP deve reduzir sua janela de congestionamento quando ocorrer um evento de perda? O TCP utiliza uma abordagem denominada "diminuição multiplicativa", reduzindo à metade o valor corrente de `CongWin` após um evento de perda. Assim, se o valor corrente de `CongWin` de um remetente TCP é 20 kbytes e for detectada uma perda, `CongWin` será reduzida à metade, isto é, a 10 kbytes. Se ocorrer um outro evento de perda, `CongWin` será reduzida mais ainda, a até 5 kbytes. O valor de `CongWin` pode continuar a cair, mas não é permitido que caia abaixo de 1 MSS. (Essa é uma descrição em linhas gerais do modo como a janela de congestionamento muda após um evento de perda. Na verdade, as coisas são um pouco mais complicadas. Como veremos brevemente, o valor de `CongWin` na verdade cai a 1 MSS após o evento de esgotamento de temporização e então aumenta rapidamente até a metade de seu valor anterior.)

Agora que acabamos de descrever como um remetente TCP reduz sua taxa de envio em face de um congestionamento detectado, é natural que passemos a considerar como o TCP deve *aumentar* sua taxa de envio quando percebe que não há congestionamento, isto é, quando chegam ACKs para dados que ainda não tinham sido reconhecidos previamente. O princípio racional para um aumento na taxa é que, se nenhum congestionamento for detectado, então provavelmente há largura de banda disponível (não utilizada) que poderia ser usada adicionalmente pela conexão TCP. Nessas circunstâncias, o TCP aumenta sua janela de congestionamento lentamente, sondando cuidadosamente para verificar se há largura de banda adicional disponível no caminho fim-a-fim. O remetente TCP faz isso incrementando `CongWin` um pouco cada vez que recebe um reconhecimento, tendo como meta aumentar `CongWin` de 1 MSS a cada tempo de viagem de ida e volta [RFC 2581]. Isso pode ser realizado de diversas maneiras; uma abordagem comum é o remetente TCP aumentar sua `CongWin` de MSS × (MSS/`CongWin`) bytes sempre que chegar um novo reconhecimento. Por exemplo, se MSS for 1.460 bytes e `CongWin` for 14.600 bytes, então dez segmentos estão sendo enviados dentro de um RTT. Cada ACK que chega (admitindo-se um ACK por segmento) aumenta o tamanho da janela de congestionamento de 1/10 MSS, e assim, após ter recebido reconhecimen-

tos para todos os dez segmentos, o valor da janela de congestionamento terá sido aumentado de 1 MSS, como desejado.

Em resumo, um remetente TCP aumenta aditivamente sua taxa quando percebe que o caminho fim-a-fim não está congestionado e a reduz multiplicativamente quando detecta (por meio de um evento de perda) que o caminho está congestionado. Por essa razão, o controle de congestionamento TCP é freqüentemente denominado **algoritmo de aumento aditivo, diminuição multiplicativa** (*additive-increase, multiplicative-decrease algorithm* — AIMD). A fase de aumento linear do protocolo de controle de congestionamento do TCP é conhecida como **prevenção de congestionamento**. O valor de `CongWin` passa repetitivamente por ciclos durante os quais aumenta linearmente e, repentinamente, cai para metade de seu valor corrente (quando ocorre um evento de perda), dando origem a um padrão semelhante a dentes de serra em conexões que estão ativas há longo tempo, como mostra a Figura 3.50.

Partida lenta

Quando uma conexão TCP começa, o valor de `CongWin` normalmente é inicializado em 1 MSS [RFC 3390], resultando em uma taxa inicial de envio de aproximadamente MSS/RTT. Como exemplo, se MSS = 500 bytes e RTT = 200 milissegundos, então a taxa de envio inicial resultante é aproximadamente 20 kbps apenas. Como a largura de banda disponível para a conexão pode ser muito maior do que MSS/RTT, seria uma pena aumentar a taxa apenas linearmente e ter de esperar por um tempo desproporcionalmente longo antes que a taxa aumentasse até algum nível respeitável. Portanto, em vez de aumentar sua taxa linearmente durante essa fase inicial, um remetente TCP a aumenta exponencialmente duplicando seu valor de `CongWin` a cada RTT. O remetente TCP continua a aumentar sua taxa de envio exponencialmente até ocorrer um evento de perda, quando então `CongWin` é cortada ao meio e passa a aumentar linearmente, como descrito anteriormente. Assim, durante essa fase inicial, que é denominada **partida lenta** (*slow start* — SS), o remetente TCP começa transmitindo a uma taxa lenta (daí o termo *partida lenta*), mas aumenta sua taxa de envio exponencialmente. O remetente gera o crescimento exponencial aumentando o valor de `CongWin` de 1 MSS toda vez que um segmento transmitido é reconhecido. Especificamente, o TCP envia o primeiro segmento para dentro da rede e espera um reconhecimento. Se esse segmento for reconhecido antes de um evento de perda, o remetente TCP aumenta a janela de congestionamento de 1 MSS e envia dois segmentos de tamanho máximo. Se esses segmentos forem reconhecidos antes de eventos de perda, o remetente aumenta a janela de congestionamento por 1 MSS para cada um dos segmentos conhecidos, resultando em uma janela de congestionamento de 4 MSS; e envia quatro segmentos de tamanho máximo. Esse procedimento continua enquanto chegarem reconhecimentos, até ocorrer, finalmente, um evento de perda. Assim, o valor de `CongWin` efetivamente duplica a cada RTT durante a fase de partida lenta.

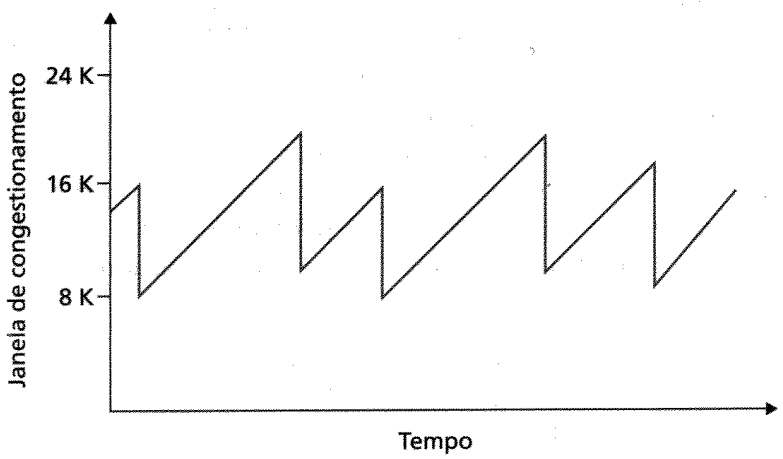


Figura 3.50 Controle de congestionamento por aumento aditivo, diminuição multiplicativa

Reação a eventos de esgotamento de temporização

Até aqui, o quadro que apresentamos da janela de congestionamento do TCP é o de um crescimento exponencial a partir de 1 MSS (durante a partida lenta) até a ocorrência de um evento, ponto em que começa o padrão em dentes de serra do AIMD. Embora esse quadro esteja próximo do real, seríamos omissos se não mencionássemos que, na verdade, o controle de congestionamento TCP reage de maneiras diferentes a eventos de perda detectados por um evento de esgotamento de temporização e a eventos de perda detectados por meio do recebimento de três ACKs duplicados. Após três ACKs duplicados, o TCP reage como acabamos de descrever — a janela de congestionamento é cortada ao meio e então aumenta linearmente. Mas, após um evento de esgotamento de temporização, um remetente TCP entra em uma fase de partida lenta — isto é, ajusta a janela de congestionamento para 1 MSS e então aumenta a janela exponencialmente. A janela continua a aumentar exponencialmente até `CongWin` alcançar metade do valor que tinha antes do evento de esgotamento de temporização. Nesse ponto, `CongWin` aumenta linearmente, exatamente como aumentaria após três ACKs duplicados.

O TCP gerencia essa dinâmica mais complexa mantendo uma variável denominada `Threshold` (patamar), que determina o tamanho da janela no qual terminará a partida lenta e começará a prevenção de congestionamento. A variável `Threshold` é inicialmente ajustada em um valor alto (65 kbps, na prática [Stevens, 1994]), portanto, não tem efeito inicial. Sempre que ocorrer um evento de perda, o valor de `Threshold` é ajustado para a metade do valor corrente de `CongWin`. Por exemplo, se a janela de congestionamento é 20 kbytes um pouco antes de um evento de perda, então o valor de `Threshold` é ajustado para 10 kbytes e manterá esse valor até o próximo evento de perda.

Agora que já descrevemos a variável `Threshold`, poderemos mostrar com exatidão como `CongWin` se comporta após um evento de esgotamento de temporização. Como indicamos anteriormente, um remetente TCP entra na fase de partida lenta após um evento de esgotamento de temporização. Enquanto está na partida lenta, o remetente TCP aumenta o valor de `CongWin` com rapidez exponencial até `CongWin` alcançar `Threshold`. Quando isso acontece, o TCP entra na fase de prevenção de congestionamento durante a qual `CongWin` aumenta exponencialmente como descrito anteriormente.

A discussão que apresentamos do algoritmo de controle de congestionamento do TCP está resumida na Tabela 3.3. Nesse ponto é natural considerar por que o controle de congestionamento TCP se comporta de maneiras diferentes após um evento de esgotamento de temporização e após o recebimento de três ACKs duplicados. Especificamente, por que um remetente TCP comporta-se de modo conservador após um evento de esgotamento de temporização, reduzindo sua janela de congestionamento para 1 MSS, enquanto, após receber três ACKs duplicados, apenas reduz sua janela de congestionamento à metade? O interessante é que uma versão mais antiga do TCP, conhecida como **TCP Tahoe**, corta incondicionalmente sua janela de congestionamento para 1 MSS e entra na fase de partida lenta após qualquer um dos tipos de evento de perda citados. A versão mais nova do TCP, **TCP Reno**, cancela a fase de partida lenta após três ACKs duplicados. A filosofia por trás do cancelamento da partida lenta nesse caso é que, mesmo que um pacote tenha sido perdido, a chegada de três ACKs duplicados indica que alguns segmentos (especificamente três segmentos adicionais além do segmento perdido) foram recebidos no remetente. Assim, diferente do caso de um esgotamento de temporização, a rede está mostrando que é capaz de entregar ao menos alguns segmentos, mesmo que esteja perdendo outros por causa do congestionamento. Esse cancelamento da fase de partida lenta após três ACKs duplicados é denominado **recuperação rápida.**

Estado	Evento	Ação de controle de congestionamento do remetente TCP	Comentário
Partida lenta (SS)	Recebimento de ACK para dados não reconhecidos anteriormente	CongWin = CongWin + MSS, Se (CongWin > Threshold) ajustar estado para "Prevenção de congestionamento"	Resulta em uma duplicação de CongWin a cada RTT
Prevenção de Congestionamento (CA)	Recebimento de ACK para dados não reconhecidos anteriormente	CongWin = CongWin + MSS · (MSS/CongWin)	Aumento aditivo, resultando em aumento de CongWin de 1 MSS a cada RTT
SS ou CA	Evento de perda detectado por três ACKs duplicados	Threshold = CongWin/2, CongWin = Threshold, ajustar estado para "Prevenção de congestionamento"	Recuperação rápida, implementando diminuição multiplicativa. CongWin não cairá abaixo de 1 MSS
SS ou CA	Esgotamento de temporização	Threshold = CongWin/2, CongWin = 1 MSS, ajustar estado para "Partida lenta"	Entra em partida lenta
SS ou CA	ACK duplicado	Incrementa contagem de ACK duplicado para segmento que está sendo reconhecido	CongWin e Threshold não mudam

Tabela 3.3 Controle de congestionamento de remetente TCP [RFC 2581], admitindo valor inicial de CongWin igual a MSS, valor de Threshold grande (por exemplo, 65 kbytes [Stevens, 1994]), e remetente TCP começando em partida lenta. O valor de estado mostrado é o estado do remetente TCP um pouco antes da ocorrência do evento. Veja [RFC 2581] se quiser mais detalhes.

A Figura 3.51 ilustra a evolução da janela de congestionamento para os TCPs Reno e Tahoe. Nessa figura, o threshold é inicialmente igual a 8 MSS. A janela de congestionamento aumenta com rapidez exponencial durante a partida lenta e alcança o threshold na quarta rodada de transmissão. Então, a janela de congestionamento aumenta linearmente até ocorrerem três ACKs duplicados logo após a oitava rodada de transmissão. Note que a janela de congestionamento é 12 · MSS quando ocorre esse evento de perda. Então, o threshold é ajustado para 0,5 · `CongWin` = 6 · MSS. Com TCP Reno, a janela de congestionamento é ajustada para `CongWin` = 6 · MSS e então aumenta linearmente. Com o TCP Tahoe, a janela de congestionamento é ajustada para 1 MSS e aumenta exponencialmente até alcançar o threshold. Esse algoritmo de controle de congestionamento foi criado por V. Jacobson [Jacobson, 1988]; várias modificações do algoritmo inicial de Jacobson são descritas em [Stevens, 1994] e no [RFC 2581].

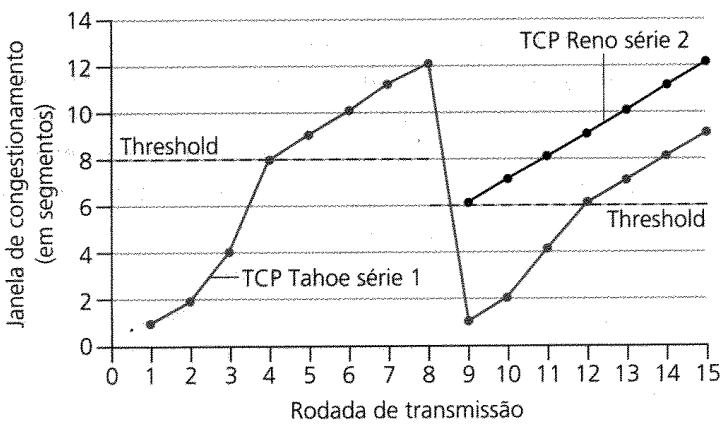


Figura 3.51 Evolução da janela de congestionamento do TCP (Tahoe e Reno)

Como já comentamos, hoje a maioria das implementações TCP usa o algoritmo Reno. Foram propostas muitas variações desse algoritmo [RFC 2582; RFC 2018]. Uma dessas variações propostas, o algoritmo TCP Vegas [Brakmo, 1995; Ahn, 1995], tenta evitar congestionamento mantendo, ao mesmo tempo, boa vazão. A idéia básica desse algoritmo é (1) detectar congestionamento nos roteadores entre a fonte e o destino antes que ocorra perda de pacotes e (2) reduzir a taxa linearmente quando for detectada perda iminente de pacotes, prevista pela observação do RTT. Quanto maior for o RTT dos pacotes, maior será o congestionamento nos roteadores.

Descrição macroscópica da dinâmica do TCP

Dado o comportamento semelhante a dentes de serra do TCP, é natural considerar qual seria a vazão média (isto é, a taxa média) de uma conexão TCP ativa há longo tempo. Nessa análise, vamos ignorar as fases de partida lenta que ocorrem após eventos de esgotamento de temporização. (Essas fases normalmente são muito curtas, visto que o remetente sai delas com rapidez exponencial.) Durante um determinado intervalo de viagem de ida e volta, a taxa à qual o TCP envia dados é uma função da janela de congestionamento e do *RTT* corrente. Quando o tamanho da janela for *w* bytes, e o tempo de viagem de ida e volta corrente for *RTT* segundos, a taxa de transmissão do TCP será aproximadamente *w/RTT*. Então, o TCP faz uma sondagem em busca de alguma largura de banda adicional aumentando *w* de 1 MSS a cada *RTT* até ocorrer um evento de perda. Seja *W* o valor de *w* quando ocorre um evento de perda. Admitindo que *RTT* e *W* são aproximadamente constantes durante o período da conexão, a taxa de transmissão fica na faixa de $W/(2 \cdot RTT)$ a *W/RTT*.

Essas suposições levam a um modelo macroscópico muito simplificado para o comportamento do TCP em regime. A rede descarta um pacote da conexão quando a taxa aumenta para *W/RTT*; então a taxa é reduzida à metade e, em seguida, aumenta de um MSS/*RTT* a cada *RTT* até alcançar *W/RTT* novamente. Esse processo se repete continuamente. Como a vazão do TCP (isto é, a taxa) aumenta linearmente entre os dois valores extremos, temos:

$$\text{Vazão média de uma conexão} = \frac{0{,}75 \cdot W}{RTT}$$

Usando esse modelo muito idealizado para a dinâmica de regime permanente do TCP, podemos também derivar uma interessante expressão que relaciona a taxa de perda de uma conexão com sua largura de banda disponível [Mahdavi, 1997]. Essa derivação está delineada nos exercícios de fixação. Um modelo mais sofisticado que demonstrou empiricamente estar de acordo com dados medidos é apresentado em [Padhye, 2000].

TCPs futuros

É importante perceber que o controle de congestionamento TCP evoluiu ao longo dos anos e, na verdade, continua a evoluir. Um resumo do controle de congestionamento TCP no final da década de 1990 pode ser encontrado no [RFC 2581]; se quiser uma discussão sobre desenvolvimentos recentes do controle de congestionamento TCP, veja [Floyd, 2001]. O que era bom para a Internet quando a maior parte das conexões TCP carregava tráfego SMTP, FTP e Telnet não é necessariamente bom para a Internet de hoje, dominada pelo HTTP, ou para uma Internet futura, com serviços que ainda nem sonhamos.

A necessidade da evolução contínua do TCP pode ser ilustrada considerando as conexões TCP de alta velocidade que são necessárias para aplicações de computação em grade [Foster, 2001]. Por exemplo, considere uma conexão TCP com segmentos de 1.500 bytes e *RTT* de 100 ms, e suponha que queremos enviar dados por essa conexão a 10 Gbps. Seguindo o [RFC 3649] e utilizando a fórmula de vazão do TCP apresentada anteriormente, notamos que, para alcançar uma vazão de 10 Gbps, o tamanho médio da janela de congestionamento precisaria ser 83.333 segmentos. São *muitos* segmentos, e pensar que um deles poderia ser perdido em trânsito nos deixa bastante preocupados. O que aconteceria no caso de uma perda? Ou, em

outras palavras, que fração dos segmentos transmitidos poderia ser perdida e ainda assim permitisse que o algoritmo de controle de congestionamento TCP delineado na Tabela 3.3 alcançasse a desejada taxa de 10 Gbps? Nos exercícios de fixação deste capítulo você percorrerá a dedução de uma fórmula que relaciona a vazão de uma conexão TCP em função da taxa de perda (L), do tempo de viagem de ida e volta (RTT) e do tamanho máximo de segmento (MSS):

$$\text{Vazão média de uma conexão} = \frac{1{,}22 \cdot MSS}{RTT\sqrt{L}}$$

Usando essa fórmula, podemos ver que, para alcançar uma vazão de 10 Gbps, o algoritmo de controle de congestionamento TCP de hoje pode tolerar uma probabilidade de perda de segmentos de apenas $2 \cdot 10^{-10}$ (equivalente a um evento de perda a cada 5 bilhões de segmentos) — uma taxa muito baixa. Essa observação levou muitos pesquisadores a investigar novas versões do TCP especificamente projetadas para esses ambientes de alta velocidade; [Jin, 2004; RFC 3649; Kelly, 2003] apresentam discussões sobre esses esforços.

3.7.1 Justiça

Considere K conexões TCP, cada uma com um caminho fim-a-fim diferente, mas todas passando pelo gargalo em um enlace com taxa de transmissão de R bps (aqui, *gargalo em um enlace* quer dizer que nenhum dos outros enlaces ao longo do caminho de cada conexão está congestionado e que todos dispõem de abundante capacidade de transmissão em comparação à capacidade de transmissão do enlace com gargalo). Suponha que cada conexão está transferindo um arquivo grande e que não há tráfego UDP passando pelo enlace com gargalo. Dizemos que um mecanismo de controle de congestionamento é *justo* se a taxa média de transmissão de cada conexão for aproximadamente R/K; isto é, cada uma obtém uma parcela igual da largura de banda do enlace.

O algoritmo AIMD do TCP é justo, considerando, em particular, que diferentes conexões TCP podem começar em momentos diferentes e, assim, ter tamanhos de janela diferentes em um dado instante? [Chiu, 1989] explica, de um modo elegante e intuitivo, por que o controle de congestionamento converge para fornecer um compartilhamento justo da largura de banda do enlace entre conexões TCP concorrentes.

Vamos considerar o caso simples de duas conexões TCP compartilhando um único enlace com taxa de transmissão R, como mostra a Figura 3.52. Vamos admitir que as duas conexões tenham os mesmos MSS e RTT (de modo que, se o tamanho de suas janelas de congestionamento for o mesmo, então terão a mesma vazão) e uma grande quantidade de dados para enviar e que nenhuma outra conexão TCP ou datagramas UDP atravesse esse enlace compartilhado. Vamos ignorar também a fase de partida lenta do TCP e admitir que as conexões TCP estão operando em modo CA (AIMD) todo o tempo.

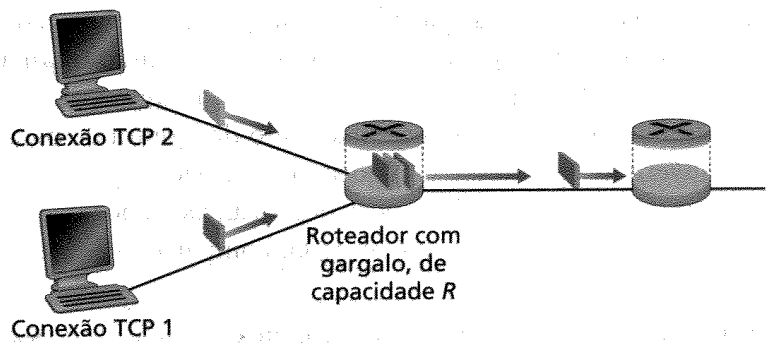


Figura 3.52 Duas conexões TCP compartilhando um único enlace congestionado

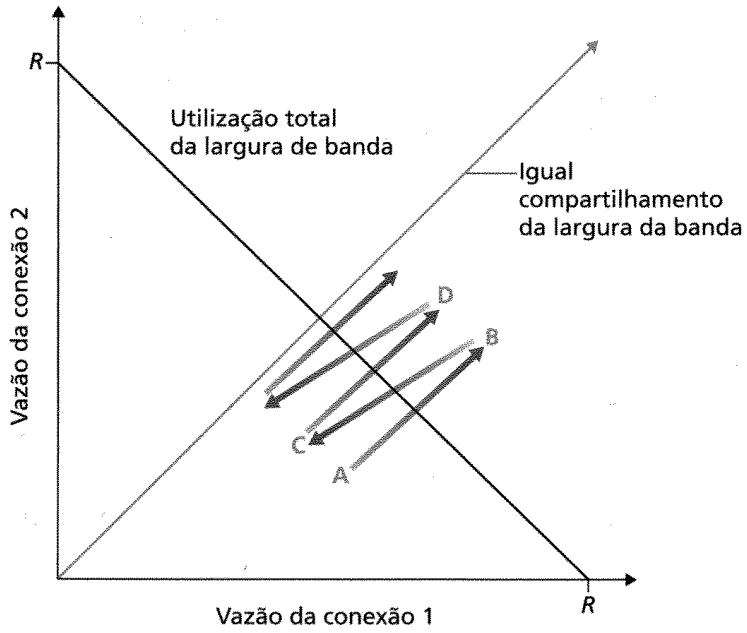


Figura 3.53 Vazão alcançada pelas conexões TCP 1 e TCP 2

A Figura 3.53 mostra a vazão alcançada pelas duas conexões TCP. Se for para o TCP compartilhar eqüitativamente a largura de banda do enlace entre as duas conexões, então a vazão alcançada deverá cair ao longo da linha a 45 graus (igual compartilhamento da largura de banda) que parte da origem. Idealmente, a soma das duas vazões seria igual a *R*. (Com certeza, não é uma situação desejável cada conexão receber um compartilhamento igual, mas igual a zero, da capacidade do enlace!) Portanto, o objetivo é que as vazões alcançadas fiquem em algum lugar perto da intersecção da linha de igual compartilhamento da largura de banda com a linha de utilização total da largura de banda da Figura 3.53.

Suponha que os tamanhos de janela TCP sejam tais que, em um determinado instante, as conexões 1 e 2 alcancem as vazões indicadas pelo ponto *A* na Figura 3.53. Como a quantidade de largura de banda do enlace consumida conjuntamente pelas duas conexões é menor do que *R*, não ocorrerá nenhuma perda e ambas as conexões aumentarão suas janelas de 1 por RTT como resultado do algoritmo de prevenção de congestionamento do TCP. Assim, a vazão conjunta das duas conexões continua ao longo da linha a 45 graus (aumento igual para as duas), começando no ponto *A*. Inevitavelmente, a largura de banda do enlace consumida em conjunto pelas duas conexões será maior do que *R* e, assim, também fatalmente, ocorrerá perda de pacote. Suponha que as conexões 1 e 2 experimentem perda de pacote quando alcançarem as vazões indicadas pelo ponto *B*. As conexões 1 e 2 então reduzirão suas janelas por um fator de 2. Assim, as vazões resultantes são as do ponto *C*, a meio caminho do vetor que começa em *B* e termina na origem. Como a utilização conjunta da largura de banda é menor do que *R* no ponto *C*, as duas conexões novamente aumentam suas vazões ao longo da linha a 45 graus que começa no ponto *C*. Mais cedo ou mais tarde ocorrerá perda, por exemplo, no ponto *D*, e as duas conexões novamente reduzirão o tamanho de suas janelas por um fator de 2 — e assim por diante. Você pode ter certeza de que a largura de banda alcançada pelas duas conexões flutuará ao longo da linha de igual compartilhamento da largura de banda. E também estar certo de que as duas conexões convergirão para esse comportamento, não importando onde elas comecem no espaço bidimensional! Embora haja uma série de suposições idealizadas por trás desse cenário, ainda assim ele dá uma idéia intuitiva de por que o TCP resulta em igual compartilhamento da largura de banda entre conexões.

Em nosso cenário idealizado, admitimos que somente conexões TCP atravessem o enlace com gargalo, que as conexões tenham o mesmo valor de RTT e que uma única conexão TCP esteja associada com um par hospedeiro/destinatário. Na prática, essas condições não são comumente encontradas e, assim, é possível que as aplicações cliente–servidor obtenham porções muito desiguais da largura de banda do enlace. Em

particular, foi demonstrado que, quando várias conexões compartilham um único enlace com gargalo, as sessões cujos RTTs são menores conseguem obter a largura de banda disponível naquele enlace mais rapidamente (isto é, abre suas janelas de congestionamento mais rapidamente) à medida que o enlace fica livre. Assim, conseguem vazões mais altas do que conexões com RTTs maiores [Lakshman, 1997].

Justiça e UDP

Acabamos de ver como o controle de congestionamento TCP regula a taxa de transmissão de uma aplicação por meio do mecanismo de janela de congestionamento. Muitas aplicações de multimídia, como telefone por Internet e videoconferência, não rodam sobre TCP exatamente por essa razão — elas não querem que sua taxa de transmissão seja limitada, mesmo que a rede esteja muito congestionada. Ao contrário, preferem rodar sobre UDP, que não tem controle de congestionamento. Quando rodam sobre esse protocolo, as aplicações podem passar seus áudios e vídeos para a rede a uma taxa constante e, ocasionalmente, perder pacotes, em vez de reduzir suas taxas a níveis 'justos' em horários de congestionamento e não perder nenhum deles. Da perspectiva do TCP, as aplicações de multimídia que rodam sobre UDP não são justas — elas não cooperam com as outras conexões nem ajustam suas taxas de transmissão de maneira adequada. Como o controle de congestionamento TCP reduzirá sua taxa de transmissão quando houver aumento de congestionamento (perda), enquanto fontes UDP não precisam fazer o mesmo, é possível que essas fontes desalojem o tráfego TCP. Uma área importante da pesquisa hoje é o desenvolvimento de mecanismos de controle de congestionamento que impeçam que o tráfego de UDP leve a vazão da Internet a uma parada repentina [Floyd, 1999; Floyd, 2000; Kohler, 2004].

Justiça e conexões TCP paralelas

Mas, mesmo que pudéssemos obrigar o tráfego UDP a se comportar com justiça, o problema ainda não estaria completamente resolvido. Isso porque não há nada que impeça uma aplicação de rodar sobre TCP usando múltiplas conexões paralelas. Por exemplo, browsers Web freqüentemente usam múltiplas conexões TCP paralelas para transferir os vários objetos de uma página Web. (O número exato de conexões múltiplas pode ser configurado na maioria dos browsers.) Quando usa múltiplas conexões paralelas, uma aplicação consegue uma fração maior da largura de banda de um enlace congestionado. Como exemplo, considere um enlace de taxa R que está suportando nove aplicações cliente–servidor em curso, e cada uma das aplicações está usando uma conexão TCP. Se surgir uma nova aplicação que também utilize uma conexão TCP, então cada aplicação conseguirá aproximadamente a mesma taxa de transmissão igual a $R/10$. Porém, se, em vez disso, essa nova aplicação usar 11 conexões TCP paralelas, então ela conseguirá uma alocação injusta de mais do que $R/2$. Como a penetração do tráfego Web na Internet é grande, as múltiplas conexões paralelas não são incomuns.

3.7.2 Modelagem do atraso TCP

Encerramos este capítulo com alguns modelos simples para calcular o tempo que leva para o TCP enviar um objeto (como uma imagem, um arquivo de texto ou um MP3). Para um dado objeto, definimos **latência** como o tempo que transcorre desde o momento em que um cliente inicia uma conexão TCP até o instante em que ele recebe o objeto requisitado na íntegra. Os modelos apresentados aqui dão uma importante percepção dos componentes fundamentais da latência, entre eles a apresentação inicial TCP, a partida lenta TCP e o tempo de transmissão do objeto.

Essa análise simples admite que a rede não está congestionada — isto é, que a conexão TCP que está transportando o objeto não tem de compartilhar largura de banda de enlace com outro tráfego de TCP ou UDP. E, também, para não ofuscar as questões centrais, a análise é realizada no contexto da rede simples de apenas um enlace mostrada na Figura 3.54. (Esse enlace pode modelar um único gargalo em um caminho fim-a-fim. Consulte também os exercícios ao final deste capítulo para uma extensão explícita para o caso de múltiplos enlaces.)

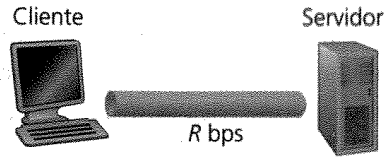


Figura 3.54 Uma rede simples de um único enlace conectando um cliente e um servidor

Vamos também fazer as seguintes suposições simplificadoras:

- A quantidade de dados que o remetente pode transmitir é limitada exclusivamente pela janela de congestionamento do remetente. (Assim, os buffers de recepção do TCP são grandes.)
- Os pacotes não são nem perdidos nem corrompidos; portanto, não há retransmissões.
- Todas as sobrecargas de cabeçalhos de protocolo — incluindo cabeçalhos TCP, IP e de camada de enlace — são desprezíveis e ignorados.
- O objeto (isto é, o arquivo) a ser transferido consiste em um número inteiro de segmentos de tamanho MSS (tamanho máximo de segmento).
- Os únicos pacotes que têm tempos de transmissão não desprezíveis são os que carregam segmentos TCP de tamanho máximo. Mensagens de requisição, reconhecimentos e segmentos de estabelecimento de conexão TCP são pequenos e têm tempos de transmissão insignificantes.
- O threshold inicial no mecanismo de controle de congestionamento TCP é um valor alto que nunca é atingido pela janela de congestionamento.

Introduzimos também a seguinte notação:

- O tamanho do objeto a ser transferido é O bits.
- O MSS é S bits (por exemplo, 536 bytes).
- A taxa de transmissão do enlace do servidor ao cliente é R bps.

Antes de iniciar a análise formal, vamos tentar alguma intuição. Qual seria a latência se não houvesse a limitação da janela de congestionamento, isto é, se fosse permitido ao servidor enviar segmentos um atrás do outro até o objeto inteiro ter sido enviado? Para responder a essa pergunta, note primeiramente que é exigido um RTT para iniciar a conexão TCP. Após um RTT, o cliente envia uma requisição para o objeto (que pega uma carona no terceiro segmento da apresentação de três vias do TCP). Após um total de dois RTTs, o cliente começa a receber dados do servidor. Ele recebe dados do servidor durante um período de tempo O/R, o tempo para o servidor transmitir o objeto inteiro. Assim, no caso de não haver limitação da janela de congestionamento, a latência total é $2\,RTT + O/R$. Isso representa um limite inferior; o procedimento de partida lenta, com sua janela de congestionamento dinâmica, evidentemente alongará essa latência.

Janela de congestionamento estática

Embora o TCP use uma janela de congestionamento dinâmica, é instrutivo analisar primeiramente uma janela de congestionamento estática. Vamos tomar W, um número inteiro, para designar uma janela de congestionamento estática de tamanho fixo. Com a janela de congestionamento estática, não é permitido que o servidor tenha mais do que W segmentos pendentes não reconhecidos. Quando o servidor recebe a requisição do cliente, imediatamente envia W segmentos um atrás do outro ao cliente. Então, envia um segmento para dentro da rede para cada reconhecimento que recebe do cliente. O servidor continua a enviar um segmento para cada reconhecimento até que todos os segmentos do objeto tenham sido enviados. Há dois casos a considerar:

1. $WS/R > RTT + S/R$. Nesse caso, o servidor recebe um reconhecimento para o primeiro segmento na primeira janela antes de o servidor completar a transmissão da primeira janela.
2. $WS/R < RTT + S/R$. Nesse caso, o servidor transmite a quantidade de segmentos equivalente à primeira janela antes de o servidor receber um reconhecimento para o primeiro segmento na janela.

Vamos considerar, de início, o primeiro caso, que está ilustrado na Figura 3.55. Nessa figura, o tamanho da janela é W = 4 segmentos.

É preciso um RTT para iniciar a conexão TCP. Após um RTT, o cliente envia uma requisição para o objeto (que pega uma carona no terceiro segmento da apresentação de três vias do TCP). Após um total de dois RTTs, o cliente começa a receber dados do servidor. Os segmentos chegam periodicamente do servidor a cada *R/S* segundos, e o cliente reconhece cada segmento que recebe do servidor. Como o servidor recebe o primeiro reconhecimento antes de concluir o envio dos segmentos equivalentes a uma janela, continua a transmitir segmentos após ter transmitido a quantidade equivalente à primeira janela de segmentos. E, como o reconhecimento chega periodicamente ao servidor a cada *S/R* segundos desde o instante em que chega o primeiro reconhecimento, o servidor transmite segmentos continuamente até transmitir o objeto inteiro. Assim, uma vez que comece a transmitir o objeto à taxa *R*, o servidor continua a transmiti-lo a essa taxa até que todo o objeto seja transmitido. Por conseguinte, a latência é 2 RTT + *O/R*.

Vamos agora considerar o segundo caso, que está ilustrado na Figura 3.56. Nessa figura, o tamanho da janela é W = 2 segmentos.

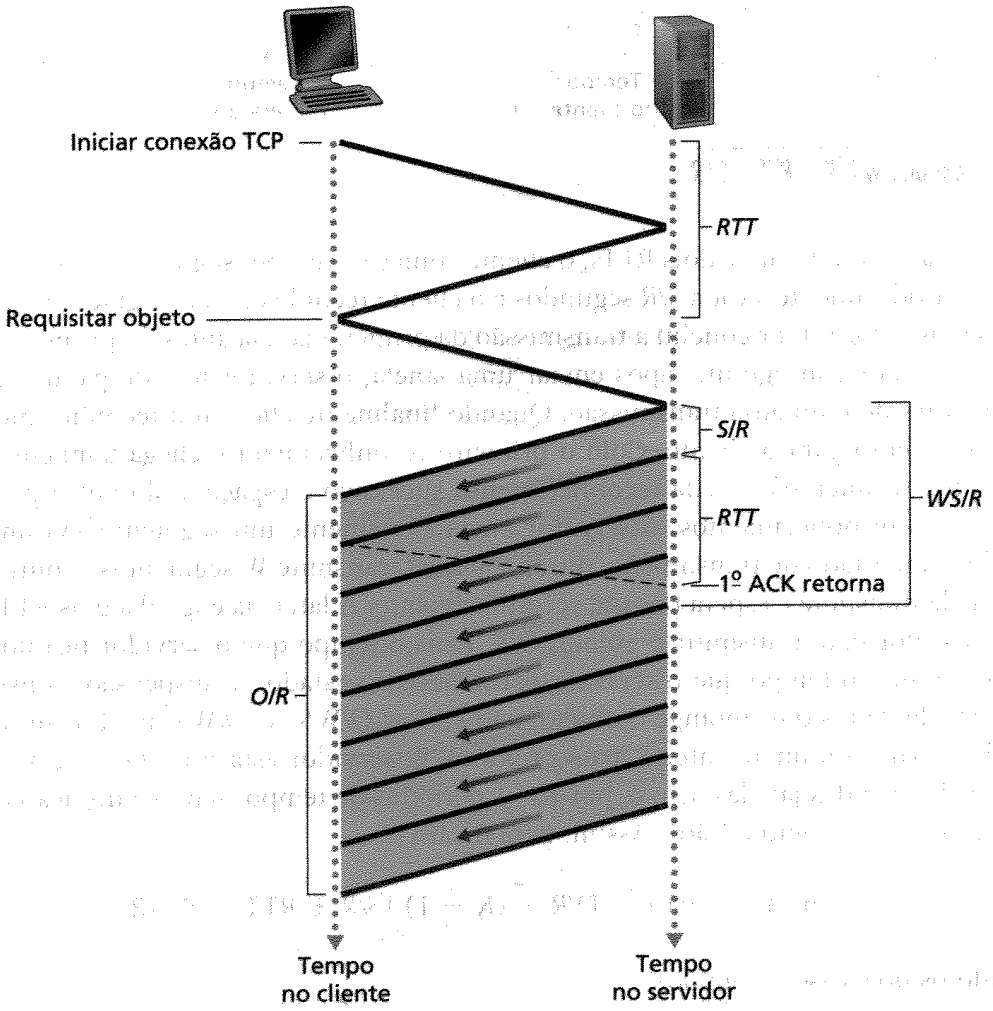


Figura 3.55 Caso em que $WS/R > RTT + S/R$

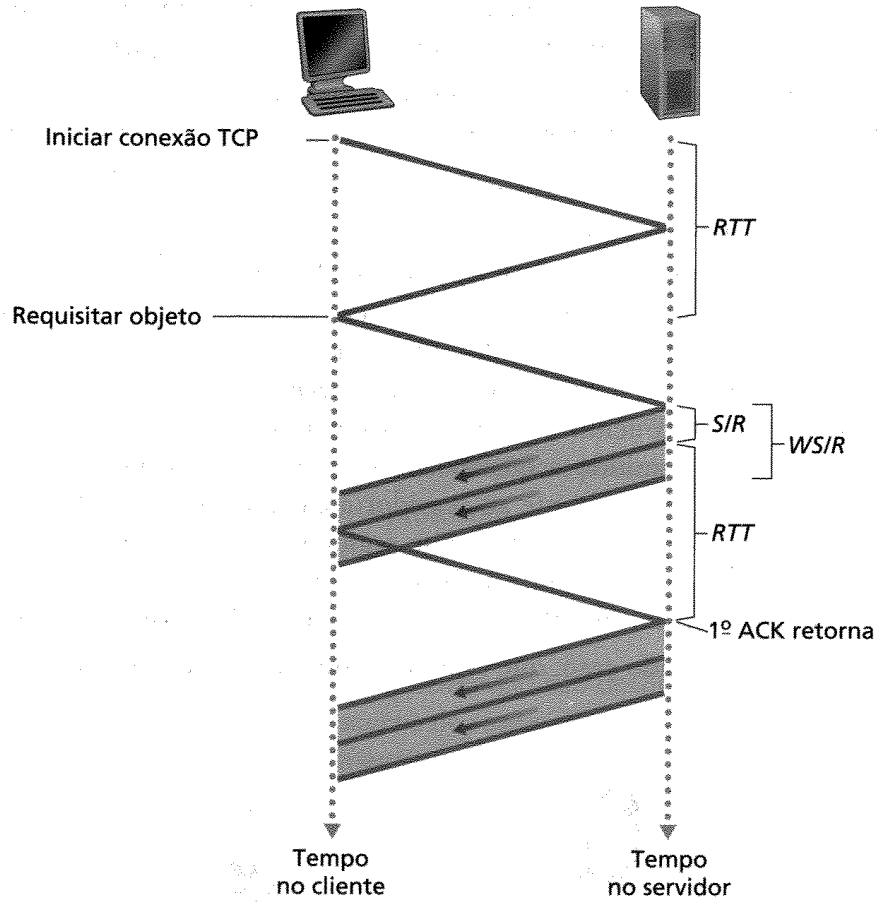


Figura 3.56 Caso em que $WS/R < RTT + S/R$

Mais uma vez, após um total de dois RTTs, o cliente começa a receber segmentos do servidor. Esses segmentos chegam periodicamente a cada S/R segundos e o cliente reconhece cada segmento que recebe do servidor. Mas, nesse caso, o servidor conclui a transmissão da primeira janela antes do primeiro reconhecimento chegar do cliente. Por conseguinte, após enviar uma janela, o servidor tem de parar e esperar por um reconhecimento antes de retomar a transmissão. Quando finalmente chega um reconhecimento, o servidor envia um novo segmento para o cliente. Com o primeiro reconhecimento, chega também o equivalente a uma janela de reconhecimentos, e cada reconhecimento sucessivo é espaçado do outro por S/R segundos. Para cada um desses reconhecimentos, o servidor envia exatamente um segmento. Assim, ele se alterna entre dois estados: um estado de transmissão, durante o qual transmite W segmentos, e outro de suspensão, durante o qual nada transmite e espera por um reconhecimento. A latência é igual a dois RTTs mais o tempo requerido pelo servidor para transmitir o objeto, O/R, mais o tempo que o servidor fica no estado de suspensão. Para determinar o tempo que o servidor permanece no estado de suspensão, considere que K é o número de janelas de dados que abrange o objeto; isto é, $K = O/WS$; se O/WS não for um número inteiro, então arredonde K para o número inteiro mais próximo. O servidor está no estado de suspensão entre a transmissão de cada uma das janelas, isto é, por $K - 1$ períodos de tempo, sendo a duração de cada período $\text{RTT} - (W - 1)S/R$ (veja a Figura 3.56). Assim, para o caso 2:

$$\text{Latência} = 2\,\text{RTT} + O/R + (K - 1)\,[S/R + \text{RTT} - WS/R]$$

Combinando os dois casos, obtemos:

$$\text{Latência} = 2\,\text{RTT} + O/R + (K - 1)\,[S/R + \text{RTT} - WS/R]^{+}$$

em que $[x]^+ = \max(x,0)$. Note que o atraso tem três componentes: 2 RTT para estabelecer a conexão TCP e para requisitar e começar a receber o objeto; O/R, o tempo para o servidor transmitir o objeto; e um termo final $(K - 1)\,[S/R + RTT - WS/R]^+$, para a quantidade de tempo em que o servidor fica em suspenso.

Isso conclui nossa análise de janelas estáticas. A análise seguinte para janelas dinâmicas é mais complicada mas semelhante à das janelas estáticas.

Janela de congestionamento dinâmica

Vamos agora considerar a janela de congestionamento dinâmica TCP no modelo da latência. Lembre-se de que o servidor inicia com uma janela de congestionamento de um segmento e envia um segmento ao cliente. Quando recebe um reconhecimento para o segmento, o servidor aumenta sua janela de congestionamento para dois segmentos e envia dois segmentos ao cliente (espaçados por S/R segundos). Ao receber os reconhecimentos para os dois segmentos, ele aumenta a janela de congestionamento para quatro segmentos e envia quatro segmentos ao cliente (novamente espaçados por S/R segundos). O processo continua, com a janela de congestionamento dobrando a cada RTT. Um diagrama de eventos do TCP está ilustrado na Figura 3.57.

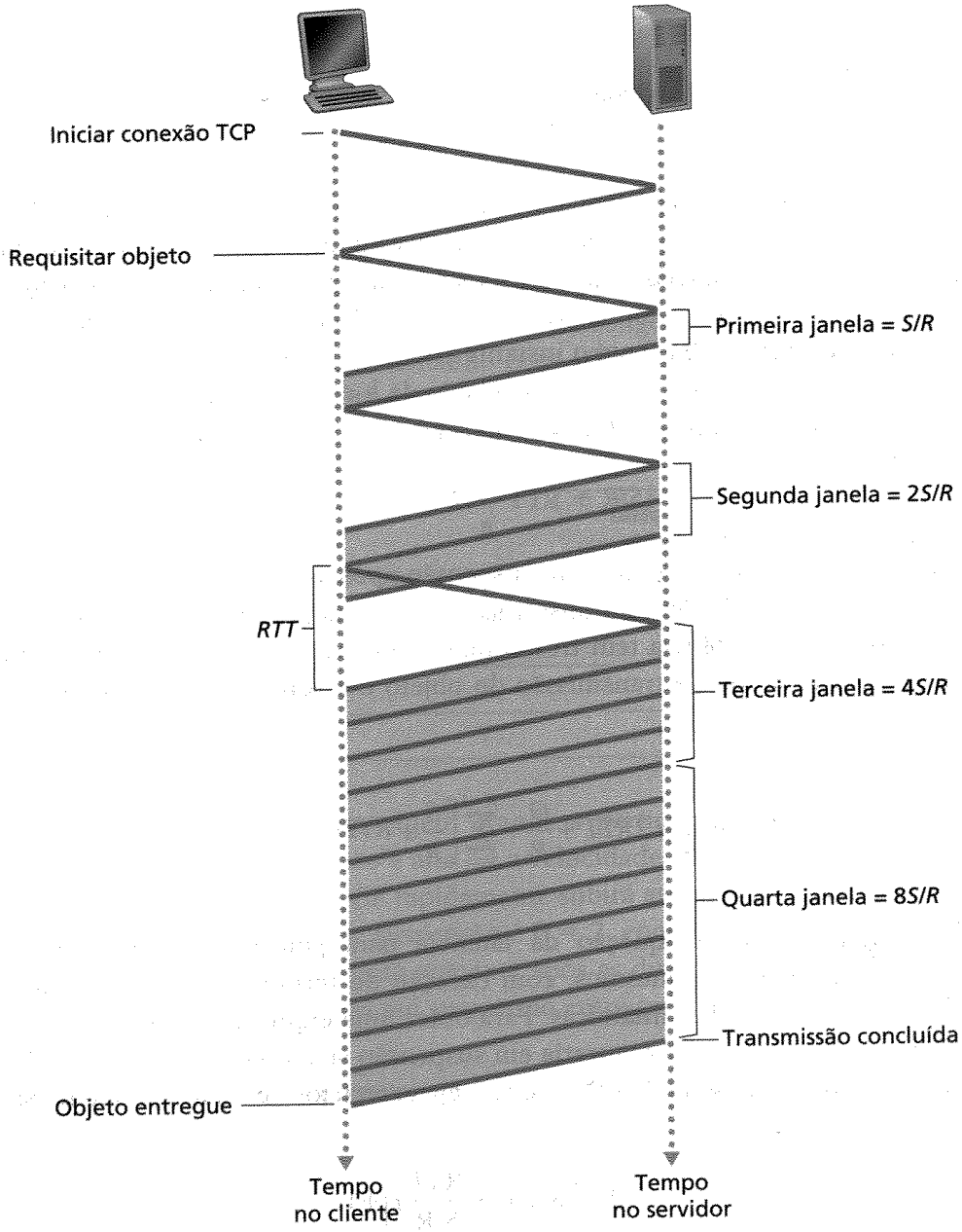


Figura 3.57 Eventos do TCP durante a partida lenta

Note que O/S é o número de segmentos no objeto; na Figura 3.57, $O/S = 15$. Considere o número de segmentos que estão em cada uma das janelas. A primeira janela contém um segmento, a segunda contém dois segmentos e a terceira contém quatro segmentos. De modo mais geral, a k-ésima janela contém 2^{k-1} segmentos. Seja K o número de janelas que abrangem o objeto; no diagrama anterior, $K = 4$. Em geral, podemos exprimir K em termos de O/S como segue:

$$K = \text{mín}\left\{k: 2^0 + 2^1 + \ldots + 2^{k-1} \geq \frac{O}{S}\right\}$$

$$= \text{mín}\left\{k: 2^k - 1 \geq \frac{O}{S}\right\}$$

$$= \text{mín}\left\{k: k \geq \log_2\left(\frac{O}{S} + 1\right)\right\}$$

$$= \left\lceil \log_2\left(\frac{O}{S} + 1\right)\right\rceil$$

Após transmitir o equivalente a uma janela de dados, o servidor pode ficar em suspenso (isto é, parar de transmitir) enquanto espera por um reconhecimento. Na Figura 3.55, o servidor fica em suspenso após transmitir a primeira e a segunda janelas, mas não após transmitir a terceira. Vamos calcular agora o tempo em suspensão após a transmissão da k-ésima janela. O tempo decorrido desde o momento em que o servidor começa a transmitir a k-ésima janela até o instante em que recebe um reconhecimento para o primeiro segmento na janela é $S/R + \text{RTT}$. O tempo de transmissão da k-ésima janela é $(S/R)\ 2^{k-1}$. O tempo de suspensão é a diferença entre essas duas quantidades, ou seja,

$$[S/R + \text{RTT} - 2^{k-1}\ (S/R)]^+$$

O servidor pode, potencialmente, entrar em suspensão após a transmissão de cada uma das primeiras $K - 1$ janelas. (O servidor concluiu após a transmissão da k-ésima janela.) Podemos agora calcular a latência para transferir o arquivo. A latência tem três componentes: 2 RTT para estabelecer a conexão TCP e requisitar o arquivo, O/R, o tempo de transmissão do objeto e a soma de todos os tempos de suspensão. Assim,

$$\text{Latência} = 2RTT + \frac{O}{R} + \sum_{k=1}^{K-1}\left[\frac{S}{R} + RTT - 2^{k-1}\frac{S}{R}\right]^+$$

Compare essa equação com a equação da latência para janelas de congestionamento estáticas. Praticamente todos os termos são os mesmos, com exceção do termo WS/R para janelas estáticas, que é substituído por $2^{k-1}(S/R)$ para janelas dinâmicas. Para obter uma expressão mais compacta para a latência, seja Q o número de vezes que o servidor entraria em suspensão se o objeto contivesse um número infinito de segmentos. Com uma dedução semelhante à de K (veja exercícios de fixação), obtemos:

$$Q = \left\lfloor \log_2\left(1 + \frac{\text{RTT}}{S/R}\right)\right\rfloor + 1$$

O número real de vezes que o servidor entra em suspensão é $P = \min\{Q, K - 1\}$. Na Figura 3.55, $P = Q = 2$. Combinando as duas equações anteriores (veja exercícios de fixação), temos a seguinte expressão fechada para a latência:

$$\text{Latência} = 2RTT + \frac{O}{R} + P\left[RTT + \frac{S}{R}\right] - (2^P - 1)\frac{S}{R}$$

Assim, para o cálculo da latência, devemos simplesmente calcular K e Q, definir $P = \min\{Q, K - 1\}$ e substituir P na fórmula anterior.

É interessante comparar a latência do TCP com a latência que ocorreria se não houvesse controle de congestionamento (isto é, se não houvesse limitação da janela de congestionamento). Sem controle de congestionamento, a latência é 2 RTT + O/R, que definimos como *latência mínima*. É um exercício simples mostrar que:

$$\frac{\text{Latência}}{\text{Latência Mínima}} \leq 1 + \frac{P}{[(O/R)/RTT] + 2}$$

Vemos por essa fórmula que a partida lenta do TCP não aumentará de maneira significativa a latência se RTT << O/R, isto é, se o tempo de viagem de ida e volta for muito menor do que o tempo de transmissão do objeto.

Vamos agora examinar alguns cenários. Para todos os cenários, adotamos $S = 536$ bytes, um valor default comum para o TCP. Usamos um RTT de 100 milissegundos, que é um valor típico para um atraso continental ou intercontinental para enlaces moderadamente congestionados. Em primeiro lugar, considere enviar um objeto relativamente grande de tamanho $O = 100$ kbytes. O número de janelas que abrange esse objeto é $K = 8$. Para uma série de taxas de transmissão, a seguinte tabela apresenta o efeito do mecanismo da partida lenta sobre a latência.

R	O/R	P	Latência mínima: $O/R + 2$ RTT	Latência com partida lenta
28 kbps	28,6 s	1	28,8 s	28,9 s
100 kbps	8 s	2	8,2 s	8,4 s
1 Mbps	800 ms	5	1 s	1,5 s
10 Mbps	80 ms	7	0,28 s	0,98 s

Verificamos nessa tabela que para um objeto grande a partida lenta adiciona um atraso apreciável somente quando a velocidade de transmissão é alta. Se a velocidade de transmissão for baixa, os reconhecimentos voltarão com relativa rapidez e o TCP rapidamente subirá até sua taxa máxima. Por exemplo, quando $R = 100$ kbps, o número de períodos de suspensão é $P = 2$, enquanto o número de janelas a transmitir é $K = 8$; assim, o servidor entra em suspensão somente após as duas primeiras das oito janelas. Por outro lado, quando $R = 10$ Mbps, o servidor entra em suspensão entre cada janela, o que causa um aumento significativo no atraso.

Considere, agora, enviar um objeto pequeno de tamanho $O = 5$ kbytes. O número de janelas que abrange esse objeto é $K = 4$. Para uma série de taxas de transmissão, a tabela a seguir examina o efeito do mecanismo de partida lenta.

R	O/R	P	Latência mínima: O/R + 2 RTT	Latência com partida lenta
28 kbps	1,43 s	1	1,63 s	1,73 s
100 kbps	0,4 s	2	0,6 s	0,76 s
1 Mbps	40 ms	3	0,24 s	0,52 s
10 Mbps	4 ms	3	0,20 s	0,50 s

Mais uma vez, a partida lenta adiciona um atraso apreciável quando a velocidade de transmissão é alta. Por exemplo, quando $R = 1$ Mbps, o servidor entra em suspensão após cada janela, o que faz com que a latência seja mais do que duas vezes a latência mínima.

Para um RTT maior, o efeito da partida lenta se torna significativo para objetos pequenos para velocidades de transmissão menores. A tabela a seguir examina o efeito da partida lenta para RTT = 1 segundo e objeto $O = 5$ kbytes ($K = 4$).

R	O/R	P	Latência mínima: O/R + 2 RTT	Latência com partida lenta
28 kbps	1,43 s	3	3,4 s	5,8 s
100 kbps	0,4 s	3	2,4 s	5,2 s
1 Mbps	40 ms	3	2,0 s	5,0 s
10 Mbps	4 ms	3	2,0 s	5,0 s

Em resumo, a partida lenta pode aumentar significativamente a latência quando o tamanho do objeto é relativamente pequeno e o RTT, relativamente grande. Lamentavelmente, esse é o cenário freqüente quando enviamos objetos pela Web.

Um exemplo: HTTP

Como aplicação da análise da latência, vamos agora calcular o tempo de resposta para uma página Web enviada por HTTP não persistente. Suponha que a página consista em uma página HTML básica e M imagens referenciadas. Para simplificar, considere que cada um dos $M + 1$ objetos contenha exatamente O bits.

Com HTTP não persistente, cada objeto é transferido independentemente, um após o outro. O tempo de resposta da página Web é, portanto, a soma das latências para cada objeto. Assim,

$$\text{Tempo de resposta} = (M+1)\left\{2RTT + \frac{O}{R} + P\left[RTT + \frac{S}{R}\right] - (2^P - 1)\frac{S}{R}\right\}$$

Note que o tempo de resposta para o HTTP não persistente toma a seguinte forma:

Tempo de resposta = $(M + 1)\ O/R + 2(M + 1)$ RTT + latência devida à partida lenta TCP para cada um dos $M + 1$ objetos.

Fica claro que, se houver muitos objetos na página Web e se RTT for grande, o HTTP não persistente terá um fraco desempenho de tempo de resposta. Nos exercícios ao final deste capítulo, examinaremos o tempo de resposta para outros esquemas de transporte HTTP. Aconselhamos ao leitor consultar [Heidemann, 1997; Cardwell, 2000] para uma análise relacionada.

3.8 Resumo

Começamos este capítulo estudando os serviços que um protocolo de camada de transporte pode prover às aplicações de rede. Por um lado, o protocolo de camada de transporte pode ser muito simples e oferecer serviços básicos às aplicações, provendo apenas uma função de multiplexação/demultiplexação para processos comunicantes. O protocolo UDP da Internet é um exemplo desse serviço básico de um protocolo de camada de transporte. Por outro lado, um protocolo de camada de transporte pode fornecer uma variedade de garantias às aplicações, como entrega confiável de dados, garantias contra atrasos e garantias de largura de banda. Não obstante, os serviços que um protocolo de transporte pode prover são freqüentemente limitados pelo modelo de serviço do protocolo subjacente de camada de rede. Se o protocolo de camada de rede não puder proporcionar garantias contra atraso ou garantias de largura de banda para segmentos da camada de transporte, então o protocolo de camada de transporte não poderá fornecer essas garantias para as mensagens enviadas entre processos.

Aprendemos na Seção 3.4 que um protocolo de camada de transporte pode prover transferência confiável de dados mesmo que a camada de rede subjacente seja não confiável. Vimos que há muitos pontos sutis na transferência confiável de dados, mas que a tarefa pode ser realizada pela combinação cuidadosa de reconhecimentos, temporizadores, retransmissões e números de seqüência.

Embora tenhamos examinado a transferência confiável de dados neste capítulo, devemos ter em mente que essa transferência pode ser fornecida por protocolos de camada de enlace, de rede, de transporte ou de aplicação. Qualquer uma das camadas superiores da pilha de protocolos pode implementar reconhecimentos, temporizadores, retransmissões e números de seqüência e prover transferência confiável de dados para a camada situada acima dela. Na verdade, com o passar dos anos, engenheiros e cientistas da computação projetaram e implementaram, independentemente, protocolos de camada de enlace, de rede, de transporte e de aplicação que fornecem transferência confiável de dados (embora muitos desses protocolos tenham desaparecido silenciosamente).

Na Seção 3.5, examinamos detalhadamente o TCP, o protocolo de camada de transporte confiável orientado para conexão da Internet. Aprendemos que o TCP é complexo e envolve gerenciamento da conexão, controle de fluxo, estimativa de tempo de viagem de ida e volta, bem como transferência confiável de dados. Na verdade, o TCP é mais complexo do que nossa descrição — intencionalmente, não discutimos uma variedade de ajustes, acertos e melhorias que estão implementados em várias versões do TCP. Toda essa complexidade, no entanto, fica escondida da aplicação de rede. Se um cliente em um hospedeiro quiser enviar dados de maneira confiável para outro hospedeiro, ele simplesmente abre uma porta TCP para o servidor e passa dados para dentro dessa porta. A aplicação cliente–servidor felizmente fica alheia a toda a complexidade do TCP.

Na Seção 3.6, examinamos o controle de congestionamento de uma perspectiva mais ampla e, na Seção 3.7, demonstramos como o TCP implementa controle de congestionamento. Aprendemos que o controle de congestionamento é imperativo para o bem-estar da rede. Sem ele, uma rede pode facilmente ficar travada, com pouco ou nenhum dado sendo transportado fim-a-fim. Na Seção 3.7, aprendemos também que o TCP implementa um mecanismo de controle de congestionamento fim-a-fim que aumenta aditivamente sua taxa de transmissão quando julga que o caminho da conexão TCP está livre de congestionamento e reduz multiplicativamente sua taxa de transmissão quando ocorre perda. Esse mecanismo também luta para dar a cada conexão TCP que passa por um enlace congestionado uma parcela igual da largura de banda da conexão. Examinamos ainda com um certo detalhe o impacto do estabelecimento da conexão TCP e da partida lenta sobre a latência. Observamos que, em muitos cenários importantes, o estabelecimento da conexão e a partida lenta contribuem de modo significativo para o atraso fim-a-fim. Enfatizamos mais uma vez que, embora tenha evoluído com o passar dos anos, o controle de congestionamento TCP permanece como uma área de pesquisa intensa e, provavelmente, continuará a evoluir nos anos vindouros.

No Capítulo 1, dissemos que uma rede de computadores pode ser dividida em 'periferia da rede' e 'núcleo da rede'. A periferia da rede abrange tudo o que acontece nos sistemas finais. Com o exame da cama-

da de aplicação e da camada de transporte, nossa discussão sobre a periferia da rede está completa. É hora de explorar o núcleo da rede! Essa jornada começa no próximo capítulo, em que estudaremos a camada de rede, e continua no Capítulo 5, em que estudaremos a camada de enlace.

Exercícios de fixação

Capítulo 3 Questões de revisão

Seções 3.1 a 3.3

1. Considere uma conexão TCP entre o hospedeiro A e o hospedeiro B. Suponha que os segmentos TCP que trafegam do hospedeiro A para o hospedeiro B tenham número de porta da fonte x e número de porta do destino y. Quais são os números de porta da fonte e do destino para os segmentos que trafegam do hospedeiro B para o hospedeiro A?
2. Descreva por que um desenvolvedor de aplicação pode escolher rodar uma aplicação sobre UDP em vez de sobre TCP.
3. É possível que uma aplicação desfrute de transferência confiável de dados mesmo quando roda sobre UDP? Caso a resposta seja afirmativa, como isso acontece?

Seção 3.5

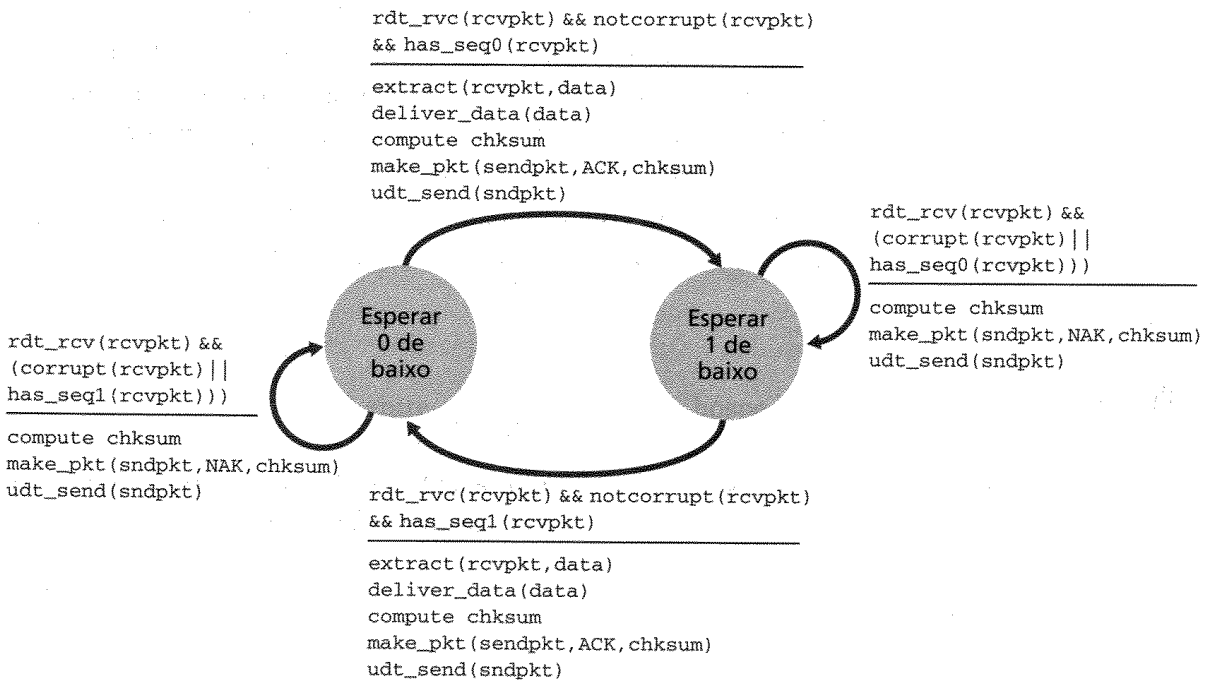
4. Verdadeiro ou falso:
   a. O hospedeiro A está enviando ao hospedeiro B um arquivo grande por uma conexão TCP. Suponha que o hospedeiro B não tenha dados para enviar para o hospedeiro A. O hospedeiro B não enviará reconhecimentos para o hospedeiro A porque ele não pode dar carona aos reconhecimentos dos dados.
   b. O tamanho de `RcvWindow` do TCP nunca muda enquanto dura a conexão.
   c. Suponha que o hospedeiro A esteja enviando ao hospedeiro B um arquivo grande por uma conexão TCP. O número de bytes não reconhecidos que o hospedeiro A envia não pode exceder o tamanho do buffer de recepção.
   d. Imagine que o hospedeiro A esteja enviando ao hospedeiro B um arquivo grande por uma conexão TCP. Se o número de seqüência para um segmento dessa conexão for m, então o número de seqüência para o segmento subseqüente será necessariamente $m + 1$.
   e. O segmento TCP tem um campo em seu cabeçalho para `RcvWindow`.
   f. Suponha que o último `SampleRTT` de uma conexão TCP seja igual a 1 segundo. Então, o valor corrente de `TimeoutInterval` para a conexão será necessariamente ajustado para um valor ≥ 1 segundo.
   g. Imagine que o hospedeiro A envie ao hospedeiro B, por uma conexão TCP, um segmento com o número de seqüência 38 e 4 bytes de dados. Nesse mesmo segmento, o número de reconhecimento será necessariamente 42.
5. Suponha que o hospedeiro A envie dois segmentos TCP um atrás do outro ao hospedeiro B sobre uma conexão TCP. O primeiro segmento tem número de seqüência 90 e o segundo, número de seqüência 110.
   a. Quantos dados tem o primeiro segmento?
   b. Suponha que o primeiro segmento seja perdido, mas o segundo chegue a B. No reconhecimento que B envia a A, qual será o número?
6. Considere o exemplo do Telnet discutido na Seção 3.5. Alguns segundos após o usuário digitar a letra 'C', ele digitará a letra 'R'. Depois disso, quantos segmentos serão enviados e o que será colocado nos campos de número de seqüência e de reconhecimento dos segmentos?

Seção 3.7

7. Suponha que duas conexões TCP estejam presentes em algum enlace congestionado de velocidade *R* bps. Ambas as conexões têm um arquivo imenso para enviar (na mesma direção, pelo enlace congestionado). As transmissões dos arquivos começam exatamente ao mesmo tempo. Qual é a velocidade de transmissão que o TCP gostaria de dar a cada uma das conexões?
8. Verdadeiro ou falso: considere o controle de congestionamento no TCP. Quando um temporizador expira no remetente, o threshold é ajustado para a metade de seu valor anterior.

Problemas

1. Suponha que o cliente A inicie uma sessão Telnet com o servidor S. Quase ao mesmo tempo, o cliente B também inicia uma sessão Telnet com o servidor S. Forneça possíveis números de porta da fonte e do destino para:
   a. Os segmentos enviados de A para S.
   b. Os segmentos enviados de B para S.
   c. Os segmentos enviados de S para A.
   d. Os segmentos enviados de S para B.
   e. Se A e B são hospedeiros diferentes, é possível que o número de porta da fonte nos segmentos de A para S seja o mesmo que nos de B para S?
   f. E se forem o mesmo hospedeiro?
2. Considere a Figura 3.5. Quais são os valores da porta de fonte e da porta de destino nos segmentos que fluem do servidor de volta aos processos clientes? Quais são os endereços IP nos datagramas de camada de rede que carregam os segmentos de camada de transporte?
3. O UDP e o TCP usam complementos de 1 para suas somas de verificação. Suponha que você tenha as seguintes três palavras de 8 bits: 01010101, 01110000 e 01001100. Qual é o complemento de 1 para as somas dessas palavras? (Note que, embora o UDP e o TCP usem palavras de 16 bits no cálculo da soma de verificação, nesse problema solicitamos que você considere parcelas de 8 bits). Mostre todo o trabalho. Por que o UDP toma o complemento de 1 da soma, isto é, por que não toma apenas a soma? Com o esquema de complemento de 1, como o destinatário detecta erros? É possível que um erro de 1 bit passe despercebido? E um erro de 2 bits?
4. Considere nosso motivo para corrigir o protocolo `rtd2.1`. Demonstre que o destinatário apresentado na figura da página seguinte, quando em operação com o remetente mostrado na Figura 3.11, pode levar remetente e destinatário a entrar em estado de travamento, em que cada um espera por um evento que nunca vai ocorrer.
5. No protocolo `rdt3.0`, os pacotes ACK que fluem do destinatário ao remetente não têm números de seqüência (embora tenham um campo de ACK que contém o número de seqüência do pacote que estão reconhecendo). Por que nossos pacotes ACK não requerem números de seqüência?
6. Elabore a FSM para o lado destinatário do protocolo `rdt3.0`.
7. Elabore um diagrama de mensagens para a operação do protocolo `rdt3.0` quando pacotes de dados e de reconhecimento estão truncados. Seu diagrama deve ser semelhante ao usado na Figura 3.16.
8. Considere um canal que pode perder pacotes, mas cujo atraso máximo é conhecido. Modifique o protocolo `rdt2.1` para incluir esgotamento de temporização do remetente e retransmitir. Informalmente, argumente por que seu protocolo pode se comunicar de modo correto por esse canal.



9. O lado remetente do `rdt3.0` simplesmente ignora (isto é, não realiza nenhuma ação) todos os pacotes recebidos que estão errados ou com o valor errado no campo `acknum` de um pacote de reconhecimento. Suponha que em tais circunstâncias o `rdt3.0` devesse apenas retransmitir o pacote de dados corrente. Nesse caso, o protocolo ainda funcionaria? (*Dica*: considere o que aconteceria se houvesse apenas erros de bits; não há perdas de pacotes, mas ocorrem esgotamentos de temporização prematuros. Imagine quantas vezes o *enésimo* pacote seria enviado, no limite em que n se aproximasse do infinito.)

10. Considere o protocolo bit alternante (também conhecido como pare e espere). Elabore um diagrama mostrando que, se a conexão de rede entre o remetente e o destinatário puder alterar a ordem de mensagens (isto é, se for possível reordenar duas mensagens que se propagam no meio entre o remetente e o destinatário), então o protocolo bit alternante não funcionará corretamente (lembre-se de identificar claramente o sentido no qual o protocolo não funcionará corretamente). Seu diagrama deve mostrar o remetente à esquerda e o destinatário à direita; o eixo do tempo deverá estar orientado de cima para baixo na página e mostrar a troca de mensagem de dados (D) e de reconhecimento (A). Não esqueça de indicar o número de seqüência associado com qualquer segmento de dados ou de reconhecimento.

11. Considere um protocolo de transferência confiável de dados que use somente reconhecimentos negativos. Suponha que o remetente envie dados com pouca freqüência. Um protocolo que utiliza somente NAKs seria preferível a um protocolo que utiliza ACKs? Por quê? Agora suponha que o remetente tenha uma grande quantidade de dados para enviar e que a conexão fim-a-fim sofra poucas perdas. Nesse segundo caso, um protocolo que utilize somente NAKs seria preferível a um protocolo que utilize ACKs? Por quê?

12. Considere o exemplo em que se atravessa os Estados Unidos mostrado na Figura 3.17. Que tamanho deveria ter a janela para que a utilização do canal fosse maior do que 90 por cento?

13. Projete um protocolo de transferência confiável de dados com paralelismo que use apenas reconhecimentos negativos. Com que rapidez seu protocolo responderia à perda de pacotes quando a taxa de chegada de dados ao remetente fosse baixa? E quando fosse alta?

14. No protocolo genérico SR que estudamos na Seção 3.4.4, o remetente transmite uma mensagem assim que ela está disponível (se ela estiver na janela), sem esperar por um reconhecimento. Suponha, agora, que queiramos um protocolo SR que envie duas mensagens de cada vez. Isto é, o remetente enviará um par de mensagens, e o par de mensagens subseqüente somente deverá ser enviado quando o remeten-

te souber que ambas as mensagens do primeiro par foram recebidas corretamente. Suponha que o canal possa perder mensagens, mas que não as corromperá nem as reordenará. Elabore um protocolo de controle de erro para a transferência confiável unidirecional de mensagens. Dê uma descrição FSM do remetente e do destinatário. Descreva o formato dos pacotes enviados entre o remetente e o destinatário e vice-versa. Se você usar quaisquer procedimentos de chamada que não sejam os da Seção 3.4 (por exemplo, `udt_send()`, `start_timer()`, `rdt_rcv()` etc.), esclareça as ações desses procedimentos. Dê um exemplo (um diagrama de mensagens para o remetente e para o destinatário) mostrando como seu protocolo se recupera de uma perda de pacote.

15. Considere um cenário em que o hospedeiro A queira enviar mensagens para os hospedeiros B e C simultaneamente. O hospedeiro A está conectado a B e a C por um canal *broadcast* — um pacote enviado por A é levado pelo canal a B e a C. Suponha que o canal *broadcast* que conecta A, B e C possa, independentemente, perder e corromper mensagens (e assim, por exemplo, uma mensagem enviada de A poderia ser recebida corretamente por B, mas não por C). Projete um protocolo de controle de erro do tipo pare e espere para a transferência confiável de um pacote de A para B e para C, tal que A não receba novos dados da camada superior até que saiba que B e C receberam corretamente o pacote em questão. Dê descrições FSM de A e C. (*Dica*: a FSM para B deve ser essencialmente a mesma que para C.) Também dê uma descrição do(s) formato(s) de pacote usado(s).

16. Considere o protocolo GBN com um tamanho de janela remetente de 3 e uma faixa de números de seqüência de 1.024. Suponha que, no tempo t, o pacote seguinte na ordem, pelo qual o destinatário está esperando, tenha um número de seqüência k. Admita que o meio não reordene as mensagens. Responda às seguintes perguntas:

a. Quais são os possíveis conjuntos de números de seqüência dentro da janela do remetente no tempo t? Justifique sua resposta.

b. Quais são todos os possíveis valores do campo ACK em todas as mensagens que estão correntemente se propagando de volta ao remetente no tempo t? Justifique sua resposta.

17. Suponha que haja duas entidades de rede A e B e que B tenha um suprimento de mensagens de dados que será enviado a A de acordo com as seguintes convenções: quando A recebe uma solicitação da camada superior para extrair a mensagem de dados seguinte (D) de B, A deve enviar uma mensagem de requisição (R) a B no canal A a B; somente quando B receber uma mensagem R, ele poderá enviar uma mensagem de dados (D) de volta a A pelo canal B a A; A deve entregar uma cópia de cada mensagem D à camada superior; R mensagens podem ser perdidas (mas não corrompidas) no canal A a B; D mensagens, uma vez enviadas, são sempre entregues corretamente; o atraso entre ambos os canais é desconhecido e variável.

Elabore um protocolo (dê uma descrição FSM) que incorpore os mecanismos apropriados para compensar a propensão à perda do canal A a B e implemente passagem de mensagem para a camada superior na entidade A, como discutido antes. Utilize apenas os mecanismos absolutamente necessários.

18. Considere os protocolos GBN e SR. Suponha que o espaço de números de seqüência seja de tamanho k. Qual será o maior tamanho de janela permissível que evitará que ocorram problemas como os da Figura 3.27 para cada um desses protocolos?

19. Responda verdadeiro ou falso às seguintes perguntas e justifique resumidamente sua resposta:

a. Com o protocolo SR, é possível o remetente receber um ACK para um pacote que caia fora de sua janela corrente.

b. Com o GBN, é possível o remetente receber um ACK para um pacote que caia fora de sua janela corrente.

c. O protocolo bit alternante é o mesmo que o protocolo SR com janela do remetente e do destinatário de tamanho 1.

d. O protocolo bit alternante é o mesmo que o protocolo GBN com janela do remetente e do destinatário de tamanho 1.

20. Considere a transferência de um arquivo enorme de L bytes do hospedeiro A para o hospedeiro B. Suponha um MSS de 1.460 bytes.

a. Qual é o máximo valor de L tal que não sejam exauridos os números de seqüência TCP? Lembre-se de que o campo de número de seqüência TCP tem quatro bytes.

b. Para o L que obtiver em (a), descubra quanto tempo demora para transmitir o arquivo. Admita que um total de 66 bytes de cabeçalho de transporte, de rede e de enlace de dados seja adicionado a cada segmento antes que o pacote resultante seja enviado por um enlace de 10 Mbps. Ignore controle de fluxo e controle de congestionamento de modo que A possa enviar os segmentos um atrás do outro e continuamente.

21. Considere o procedimento TCP para estimar RTT. Suponha que $\alpha = 0{,}1$. Seja $\texttt{SampleRTT}_1$ a amostra mais recente de RTT, $\texttt{SampleRTT}_2$ a seguinte amostra mais recente de RTT etc.

a. Para uma dada conexão TCP, suponha que quatro reconhecimentos foram devolvidos com as amostras RTT correspondentes $\texttt{SampleRTT}_4$, $\texttt{SampleRTT}_3$, $\texttt{SampleRTT}_2$ e $\texttt{SampleRTT}_1$. Expresse EstimatedRTT em termos das quatro amostras RTT.

b. Generalize sua fórmula para n amostras de RTTs.

c. Para a fórmula em (b), considere n tendendo ao infinito. Comente por que esse procedimento de média é denominado média móvel exponencial.

22. Na Seção 3.5.3 discutimos estimativa de RTT para o TCP. Na sua opinião, por que o TCP evita medir o SampleRTT para segmentos retransmitidos?

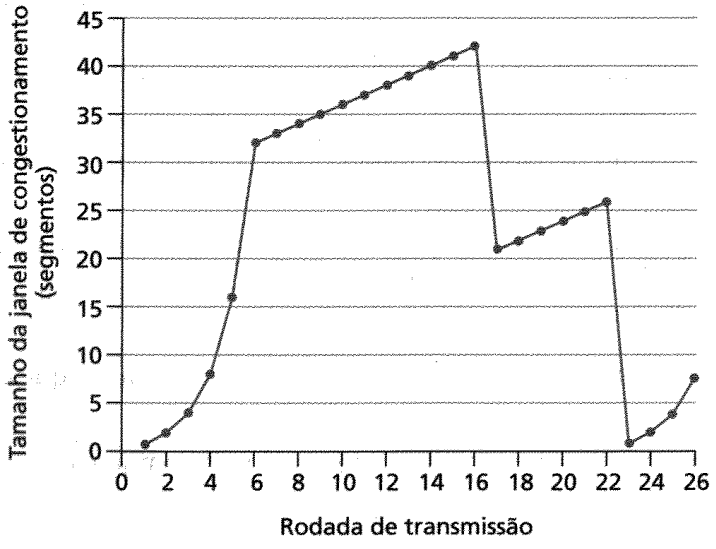
23. Qual é a relação entre a variável SendBase na Seção 3.5.4 e a variável LastByteRcvd na Seção 3.5.5?

24. Qual é a relação entre a variável LastByteRcvd na Seção 3.5.5 e a variável y na seção 3.5.4?

25. Na Seção 3.5.4 vimos que o TCP espera até receber três ACKs duplicados antes de realizar uma retransmissão rápida. Na sua opinião, por que os projetistas do TCP preferiram não realizar uma retransmissão rápida após ser recebido o primeiro ACK duplicado para um segmento?

26. Considere a Figura 3.45(b). Se λ'_{in} aumentar para mais do que $R/2$, λ_{out} poderá aumentar para mais do que $R/3$? Explique. Agora considere a Figura 3.45(c). Se λ'_{in} aumentar para mais do que $R/2$, λ_{out} poderá aumentar para mais de $R/4$ admitindo-se que um pacote será transmitido duas vezes, em média, do roteador para o destinatário? Explique.

27. Considere o seguinte gráfico que representa o tamanho de janela do TCP como uma função do tempo.



Admitindo-se que TCP Reno é o protocolo que experimenta o comportamento mostrado no gráfico, responda às seguintes perguntas. Em todos os casos você deverá apresentar uma justificativa resumida para sua resposta.

a. Quais os intervalos de tempo em que a partida lenta do TCP está em execução?

b. Quais os intervalos de tempo em que a prevenção de congestionamento do TCP está em execução?

c. Após a 16ª rodada de transmissão, a perda de segmento será detectada por três ACKs duplicados ou por um esgotamento de temporização?

d. Após a 22ª rodada de transmissão, a perda de segmento será detectada por três ACKs duplicados ou por um esgotamento de temporização?

e. Qual é o valor inicial de `Threshold` na primeira rodada de transmissão?

f. Qual é o valor inicial de `Threshold` na 18ª rodada de transmissão?

g. Qual é o valor de `Threshold` na 24ª rodada de transmissão?

h. Durante qual rodada de transmissão é enviado o 70º segmento?

i. Admitindo-se que uma perda de pacote será detectada após a 26ª rodada pelo recebimento de três ACKs duplicados, quais serão os valores do tamanho da janela de congestionamento e de `Threshold`?

28. Consulte a Figura 3.53, que ilustra a convergência do algoritmo AIMD do TCP. Suponha que, em vez de uma diminuição multiplicativa, o TCP reduza o tamanho da janela de uma quantidade constante. O AIMD resultante convergiria a um algoritmo de igual compartilhamento? Justifique sua resposta usando um diagrama semelhante ao da Figura 3.53.

29. Na Seção 3.5.4 discutimos a duplicação do intervalo de temporização após um evento de esgotamento de temporização. Esse mecanismo é uma forma de controle de congestionamento. Por que o TCP precisa de um mecanismo de controle de congestionamento que utiliza janelas (como estudado na Seção 3.7) além desse mecanismo de duplicação do intervalo de esgotamento de temporização?

30. O hospedeiro A está enviando um arquivo enorme ao hospedeiro B por uma conexão TCP. Nessa conexão nunca há perda de pacotes e os temporizadores nunca se esgotam. Seja R bps a taxa de transmissão do enlace que liga o hospedeiro A à Internet. Suponha que o processo no hospedeiro A consiga enviar dados para seu socket TCP a uma taxa de S bps, em que $S = 10 \cdot R$. Suponha ainda que o buffer de recepção do TCP seja grande o suficiente para conter o arquivo inteiro e que o buffer de envio possa conter somente um por cento do arquivo. O que impediria o processo no hospedeiro A de passar dados continuamente para seu socket TCP à taxa de S bps: o controle de fluxo do TCP; o controle de congestionamento do TCP; ou alguma outra coisa? Elabore sua resposta.

31. Relembre o modelo idealizado para a dinâmica de regime estável do TCP. No período de tempo transcorrido para a taxa da conexão variar de $W/(2 \cdot RTT)$ a W/RTT, apenas um pacote é perdido (bem ao final do período).

a. Mostre que a fração de pacotes perdidos é igual a

$$L = \text{fração de pacotes perdidos} = \frac{1}{\frac{3}{8}W^2 + \frac{3}{4}W}$$

b. Use o resultado anterior para mostrar que, se uma conexão tiver taxa de perda L, sua largura de banda média é dada aproximadamente por:

$$\approx \frac{1.22 \cdot MSS}{RTT\sqrt{L}}$$

32. Quando discutimos TCPs futuros na Seção 3.7, observamos que, para conseguir uma vazão de 10 Gbps, o TCP apenas poderia tolerar uma probabilidade de perda de segmentos de $2 \cdot 10^{-10}$ (ou, equivalentemente, uma perda para cada 5.000.000.000 segmentos). Mostre a derivação de valores para $2 \cdot 10^{-10}$ para 1 em cada 5.000.000 para os valores de RTT e MSS dados na Seção 3.7. Se o TCP precisasse suportar uma conexão de 100 Gbps, qual seria a perda tolerável?

33. Quando discutimos controle de congestionamento em TCP na Seção 3.7, admitimos implicitamente que o remetente TCP sempre tinha dados para enviar. Agora considere o caso em que o remetente TCP envie uma grande quantidade de dados e então fique ocioso em t_1 (já que não há mais dados a enviar). O TCP permanecerá ocioso por um período de tempo relativamente longo e então irá querer enviar mais dados em t_2. Quais são as vantagens e desvantagens do TCP utilizar os valores `CongWin` e `Threshold` de t_1 quando começar a enviar dados em t_2? Que alternativa você recomendaria? Por quê?

34. Considere o envio de um objeto de tamanho $O = 100$ kbytes do servidor ao cliente. Seja $S = 536$ bytes e $RTT = 100$ milissegundos. Suponha que o protocolo de transporte use janelas estáticas e tamanho de janela W. (Ver Seção 3.7.2).

a. Para uma velocidade de transmissão de 28 kbps, determine a mínima latência possível. Determine o tamanho mínimo de janela para atingir essa latência.

b. Repita (a) para 100 kbps

c. Repita (a) para 1 Mbps.

d. Repita (a) para 10 Mbps.

35. Suponha que o TCP aumente sua janela de congestionamento de um fator de 2, em vez de 1, para cada reconhecimento recebido durante a partida lenta. Assim, a primeira janela consistirá em um segmento; a segunda, em três segmentos; a terceira, em nove segmentos e assim por diante. Usando as técnicas da Seção 3.7.2:

a. Expresse K em termos de O e S.

b. Expresse Q em termos de RTT, S e R.

c. Expresse a latência em termos de $P = \min(K - 1, Q)$, O, R e RTT.

36. Considere o caso em que $RTT = 1$ segundo e $O = 100$ kbytes. Monte um gráfico (semelhante aos gráficos da Seção 3.7.2) que compare a latência mínima ($O/R + 2\ RTT$) com a latência com partida lenta para $R = 28$ kbps, 100 kbps, 1 Mbps e 10 Mbps.

37. Verdadeiro ou falso?

a. Se uma página Web consistir em exatamente um objeto, as conexões persistentes e não persistentes terão exatamente o mesmo desempenho de tempo de resposta.

b. Considere o envio de um objeto de tamanho O do servidor ao browser por TCP. Se $O > S$, em que S é o tamanho máximo de segmento, o servidor entrará em suspensão no mínimo uma vez.

c. Suponha que uma página Web consista em dez objetos, cada um de tamanho O bits. Para HTTP persistente, a porção RTT do tempo de resposta será 20 RTTs.

d. Imagine que uma página Web consista em dez objetos, cada um de tamanho O bits. Para HTTP não persistente com cinco conexões paralelas, a porção RTT do tempo de resposta será 12 RTTs.

38. Neste problema completamos alguns detalhes da derivação da latência na Seção 3.7.2.

a. Derive a fórmula

$$Q = \left\lfloor \log_2 \left(1 + \frac{RTT}{S/R}\right) \right\rfloor + 1$$

b. Use a identidade

$$\sum_{k=1}^{P} 2^{k-1} = 2^P - 1$$

para derivar a fórmula

$$\text{Latência} = 2RTT + \frac{O}{R} + P\left[RTT + \frac{S}{R}\right] - (2^P - 1)\frac{S}{R}$$

39. A análise de janelas dinâmicas na Seção 3.7.2 admite que há apenas um enlace entre servidor e cliente. Refaça a análise para T enlaces entre servidor e cliente. Admita que a rede não tenha congestionamento, de modo que os pacotes não sofram nenhum atraso de fila. Os pacotes sofrerão, contudo, um atraso de armazenagem e envio. A definição de RTT é a mesma que a dada na seção sobre controle de congestionamento TCP. (*Dica*: o tempo que transcorre entre o servidor enviar o primeiro segmento e receber o reconhecimento é $TS/R + RTT$.)

40. Relembre a discussão ao final da Seção 3.7.2 sobre o tempo de resposta para uma página Web. Para o caso de conexões não persistentes, determine a expressão geral para a *fração* do tempo de resposta que é devida à partida lenta do TCP.

Questões dissertativas

1. Considere a recepção de áudio armazenado. Faz sentido rodar a aplicação sobre UDP ou TCP? Que protocolo de transporte a RealNetworks usa? Por quê?

2. Considere essa mesma pergunta para os produtos de recepção de áudio armazenado da Microsoft.

3. Na Seção 3.7, observamos que um cliente-servidor pode criar muitas conexões paralelas "de modo injusto". O que pode ser feito para tornar a Internet verdadeiramente justa?

4. Consulte a literatura de pesquisa para saber o que quer dizer TCP *amigável*. Leia também a entrevista de Sally Floyd ao final deste capítulo. Redija uma página descrevendo a característica *amigável* do TCP.

5. Ao final da Seção 3.7.1, discutimos o fato de uma aplicação querer abrir várias conexões TCP e obter uma vazão mais alta (ou, o que é equivalente, um tempo menor de transferência de dados). Na sua opinião, por que o número de aplicações que tentaram melhorar seus desempenhos utilizando conexões múltiplas não é maior? O que aconteceria se todas as aplicações tentassem melhorar seus desempenhos utilizando conexões múltiplas? Cite algumas das dificuldades envolvidas na utilização de um elemento da rede para determinar se uma aplicação está ou não usando conexões TCP múltiplas.

Tarefas de programação

Detalhes completos das tarefas de programação podem ser encontrados no site `www.aw.com/kurose_br`.

Implementando um protocolo de transporte confiável

Nesta tarefa de programação de laboratório, você escreverá o código para a camada de transporte do remetente e do destinatário no caso da implementação de um protocolo simples de transferência confiável de

dados. Há duas versões deste laboratório: a do protocolo de bit alternante e a do GBN. Essa tarefa será muito divertida, já que a sua implementação não será muito diferente da que seria exigida em uma situação real.

Como você provavelmente não tem máquinas autônomas (com um sistema operacional que possa modificar), seu código terá de rodar em um ambiente hardware/software simulado. Contudo, a interface de programação fornecida às suas rotinas — isto é, o código que chamaria suas entidades de cima (da camada 5) e de baixo (da camada 3) — é muito próxima ao que é feito em um ambiente UNIX real. (Na verdade, as interfaces do software descritas nesta tarefa de programação são muito mais realistas do que os remetentes e destinatários de laço infinito descritos em muitos livros.) A parada e o acionamento dos temporizadores também são simulados, e as interrupções do temporizador farão com que sua rotina de tratamento de temporização seja ativada.

Ethereal Labs: Explorando o TCP

Neste laboratório, você usará seu browser Web para acessar um arquivo de um servidor Web. Como nos Ethereal Labs anteriores, você usará Ethereal para capturar os pacotes que estão chegando ao seu computador. Mas, diferentemente daqueles laboratórios, *também* poderá descarregar do mesmo servidor Web do qual descarregou o arquivo, um relatório de mensagens (trace) que pode ser lido pelo Ethereal. Nesse relatório de mensagens do servidor, você encontrará os pacotes que foram gerados pelo seu próprio acesso ao servidor Web. Você analisará os diagramas dos lados do cliente e do servidor de modo a explorar aspectos do TCP e, em particular, fará uma avaliação do desempenho da conexão TCP entre seu computador e o servidor Web. Utilizando o diagrama de mensagens, você analisará o comportamento da janela TCP e inferirá comportamentos de perda de pacote, de retransmissão, de controle de fluxo e de controle de congestionamento e do tempo de ida e volta estimado.

Como acontece com todos os Ethereal Labs, a descrição completa deste pode ser encontrada, em inglês, no site `www.aw.com/kurose_br`.

Entrevista

Sally Floyd

Sally Floyd é cientista pesquisadora no AT&T Center for Internet Research no ICSI (ACIRI), um instituto dedicado a questões da Internet e de redes. Ela é conhecida no setor por seu trabalho em projetos de protocolos de Internet, em particular nas áreas de *multicast* confiável, controle de congestionamento (TCP), escalonamento de pacotes (RED) e análise de protocolos. Sally é bacharel em Sociologia pela Universidade da Califórnia em Berkeley, e mestre e doutora em ciência da computação por essa mesma universidade.

Como você decidiu estudar ciência da computação?

Após terminar o bacharelado em sociologia, comecei a me preocupar em obter meu próprio sustento. Acabei fazendo um curso de eletrônica de dois anos na faculdade local e, em seguida, passei dez anos trabalhando com eletrônica e ciência da computação. Nesses dez anos, estão incluídos os oito que trabalhei como engenheira de sistemas de computação, cuidando dos computadores dos trens da Bay Area Rapid Transit (BART). Mais tarde, decidi estudar ciência da computação mais formalmente e iniciei a pós-graduação no Departamento de Ciência da Computação da Universidade da Califórnia em Berkeley.

O que a fez se decidir pela especialização em redes?

Na pós-graduação, comecei a me interessar pela parte teórica da ciência da computação. Primeiramente, dediquei-me à análise probabilística de algoritmos e, mais tarde, à teoria do aprendizado da computação. Nessa época, trabalhava também no Lawrence Berkeley Laboratory (LBL) uma vez por mês, e meu escritório ficava em frente ao escritório de Van Jacobson, que pesquisava algoritmos de controle de congestionamento TCP. Certo dia, Van me perguntou se eu gostaria de trabalhar na análise de alguns algoritmos para um problema relacionado a redes, que envolvia a indesejada sincronização de mensagens periódicas de roteamento. Isso me pareceu interessante, e foi o que fiz durante o verão.

Após eu ter concluído minha tese, Van me ofereceu um emprego de tempo integral para continuar trabalhando com redes. Eu não tinha planejado ficar trabalhando com redes durante dez anos, mas, para mim, a pesquisa é mais gratificante do que a ciência teórica da computação. Sinto mais satisfação em trabalhos aplicados, pois as conseqüências do que faço são mais tangíveis.

Qual foi seu primeiro emprego no setor de computação? O que implicava?

Meu primeiro emprego na área de computação foi na BART, onde trabalhei de 1975 a 1982 com os computadores que controlavam os trens. Comecei como técnica em manutenção e conserto dos vários sistemas distribuídos de computadores envolvidos na operação do sistema da BART.

Entre esses sistemas estavam: um sistema central de computadores e um sistema distribuído de minicomputadores, que controlavam a movimentação dos trens; um sistema de computadores da DEC, que exibia avisos e destinos de trens nos painéis, e um sistema de computadores Modcomp, que coletava informações das bilheterias. Meus últimos anos na BART foram dedicados a um projeto conjunto BART/LBL para estruturar um sistema substituto para o sistema de computadores de controle de trens da BART, que já estava ficando ultrapassado.

Qual parte de seu trabalho lhe apresenta mais desafios?

A pesquisa é a parte mais desafiadora. No momento, isso significa projetar e explorar um novo mecanismo para controle de congestionamento fim-a-fim com base em controle de congestionamento por equações. A intenção não é que ele substitua o TCP, mas o tráfego *unicast*, tal como algum tipo de tráfego de taxa adaptativa em tempo real, para o qual seria melhor evitar a dramática mudança na taxa que ocorre quando da redução de sua taxa à metade em resposta a uma única queda de pacote. O controle de congestionamento por equações é também de interesse como um fundamento potencial para o controle de congestionamento *multicast*.

Mais informações sobre esse assunto podem ser encontradas em http://www.psc.edu/networking/tcp_friendly.html.

Em sua opinião, qual é o futuro das redes e da Internet?

Uma possibilidade é que o congestionamento típico encontrado pelo tráfego da Internet se torne menos severo à medida que entrarem em ação os mecanismos de atribuição de preços e a largura de banda disponível aumentar mais rapidamente do que a demanda. Considero que a tendência está se dirigindo para um congestionamento menos intenso, embora não pareça de todo impossível contemplar um futuro de crescente congestionamento, a médio prazo, pontuado por colapso ocasional.

O futuro da Internet em si ou da arquitetura da Internet ainda não está claro para mim. Há muitos fatores que estão contribuindo para rápidas mudanças. Portanto, é difícil predizer como a Internet ou a arquitetura da Internet evoluirão ou mesmo prever o grau de sucesso que essa evolução será capaz de alcançar na prevenção das muitas armadilhas potenciais ao longo do caminho.

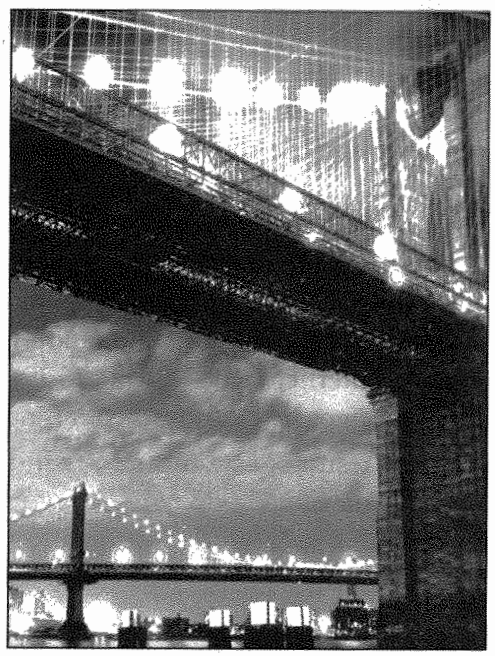
Quais pessoas a inspiraram profissionalmente?

Richard Karp, o orientador de minha tese, me ensinou a fazer, de fato, pesquisa. Van Jacobson, meu "chefe de grupo" no LBL, foi o responsável pelo meu interesse no estudo de redes e por grande parte do meu aprendizado sobre a infra-estrutura da Internet. Dave Clark me inspirou por sua visão clara da arquitetura da Internet e por seu papel no desenvolvimento dessa arquitetura por meio de pesquisa, de artigos e da participação na IETF e em outros fóruns públicos. Deborah Estrin me inspirou com sua capacidade de concentração e efetividade, e sua habilidade em tomar decisões lúcidas sobre o que investigar e por quê.

Uma das razões por que apreciei trabalhar na pesquisa de rede nos últimos dez anos é que encontrei nesse campo muitas pessoas de que gosto, a quem respeito e admiro. São pessoas inteligentes que trabalham arduamente e que estão profundamente comprometidas com o desenvolvimento da Internet, fazem um trabalho notável, divergem (ou concordam) de modo amigável e são bons companheiros para uma cerveja após um dia de reuniões.

Capítulo 4

A Camada de Rede



Vimos no capítulo anterior que a camada de transporte provê várias formas de comunicação processo a processo com base no serviço de comunicação hospedeiro a hospedeiro da camada de rede. Vimos também que a camada de transporte faz isso sem saber como a camada de rede implementa esse serviço. Portanto, é bem possível que agora você esteja imaginando o que está por baixo do serviço de comunicação hospedeiro a hospedeiro; o que o faz funcionar.

Neste capítulo estudaremos exatamente como a camada de rede implementa o serviço de comunicação hospedeiro a hospedeiro. Veremos que há um pedaço da camada de rede em cada um dos hospedeiros e roteadores na rede, o que não acontece com a camada de transporte. Por causa disso, os protocolos de camada de rede estão entre os mais desafiadores (e, portanto, os mais interessantes!) da pilha de protocolos.

A camada de rede é, também, uma das camadas mais complexas da pilha de protocolos e, assim, temos um longo caminho a percorrer. Iniciaremos nosso estudo com uma visão geral da camada de rede e dos serviços que ela pode prover. Em seguida, examinaremos novamente as duas abordagens da estruturação da entrega de pacotes de camada de rede — o modelo de datagramas e o modelo de circuitos virtuais — que vimos pela primeira vez no Capítulo 1 e veremos o papel fundamental que o endereçamento desempenha na entrega de um pacote a seu hospedeiro de destino.

Faremos, neste capítulo, uma distinção importante entre as funções de **repasse** e **roteamento** da camada de rede. Repasse envolve a transferência de um pacote de um enlace de entrada para um enlace de saída dentro de um *único* roteador. Roteamento envolve *todos* os roteadores de uma rede, cujas interações coletivas por meio do protocolo de roteamento determinam os caminhos (ou rotas) que os pacotes percorrem em suas viagens do nó de origem ao nó de destino. Manter essa distinção sempre em mente ao ler este capítulo o ajudará a situar muitos dos tópicos abordados em um contexto apropriado.

Para aprofundar nosso conhecimento do repasse de pacotes, examinaremos o "interior" de um roteador — a organização e a arquitetura de seu hardware. Então, examinaremos o repasse de pacotes na Internet, juntamente com o famoso Protocolo da Internet (IP). Investigaremos o endereçamento na camada de rede e o formato de datagrama IPv4. Em seguida, estudaremos a tradução de endereço de rede (*network address translation* — NAT), a fragmentação de datagrama, o Protocolo de Mensagem de Controle da Internet (*Internet Control Message Protocol* — ICMP) e IPv6.

Então voltaremos nossa atenção à função de roteamento da camada de rede. Veremos que o trabalho de um algoritmo de roteamento é determinar bons caminhos (ou rotas) entre remetentes e destinatários. Em primeiro lugar, estudaremos a teoria de algoritmos de roteamento concentrando nossa atenção nas duas classes de algoritmos mais predominantes: algoritmos de estado de enlace e de vetor de distâncias. Visto que a complexidade dos algoritmos de roteamento aumenta consideravelmente com o aumento do número de roteadores na rede, também será interessante abordar o roteamento hierárquico. Em seguida veremos como a teoria é posta em prática quando falarmos sobre os protocolos de roteamento de intra-sistemas autônomos da Internet (RIP, OSPF e IS-IS) e seu protocolo de roteamento de inter-sistemas autônomos, o BGP. Encerraremos este capítulo com uma discussão sobre roteamento *broadcast* e *multicast*.

Em resumo, este capítulo tem três partes importantes. A primeira, seções 4.1 e 4.2, aborda funções e serviços de camada de rede. A segunda, seções 4.3 e 4.4, examina o repasse. Finalmente, a terceira parte, seções 4.5 a 4.7, estuda roteamento.

4.1 Introdução

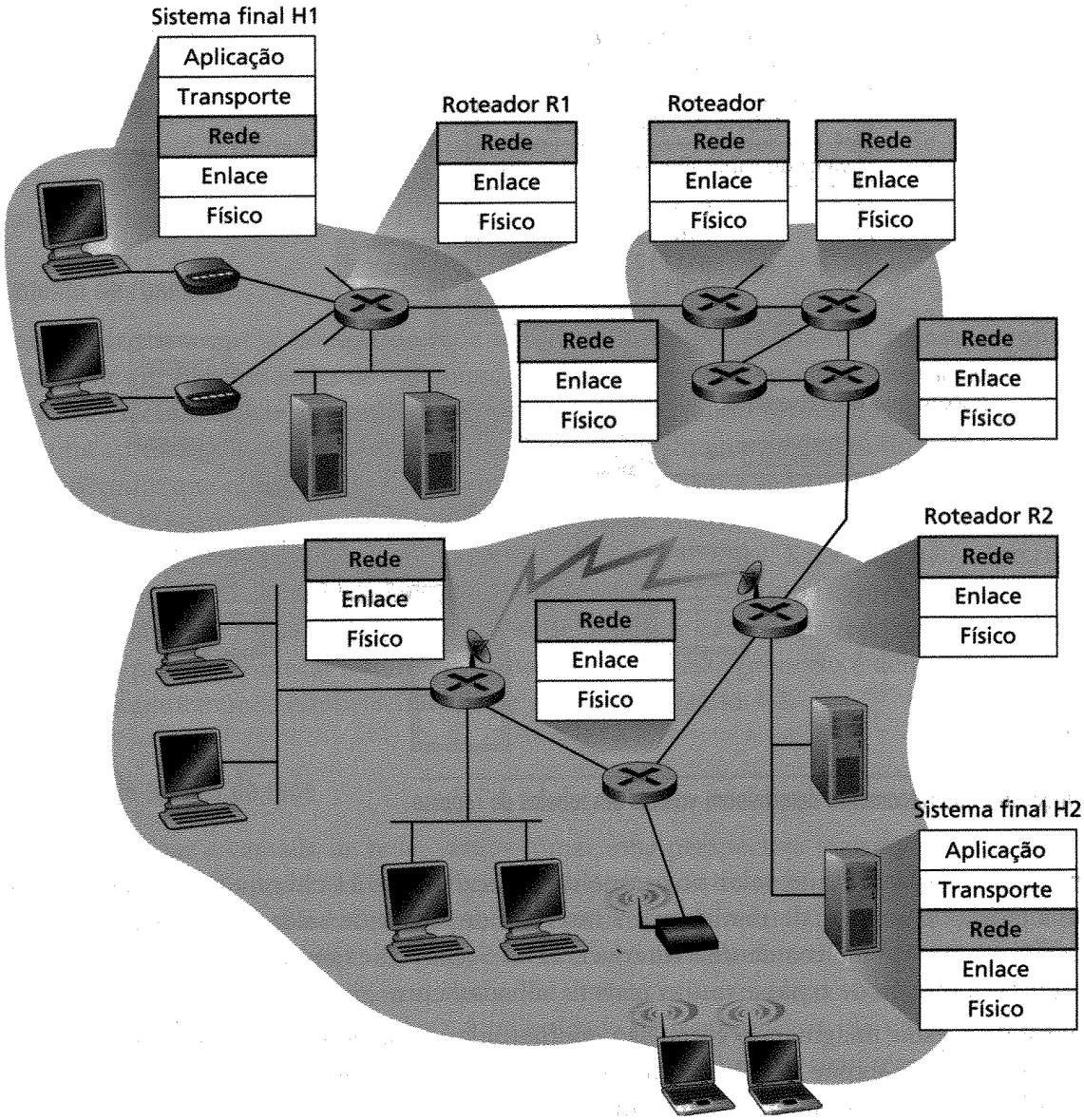
A Figura 4.1 mostra uma rede simples com dois hospedeiros, H1 e H2, e diversos roteadores no caminho entre H1 e H2. Suponha que H1 esteja enviando informações a H2 e considere o papel da camada de rede nesses hospedeiros e nos roteadores intervenientes. A camada de rede em H1 pegará segmentos da camada de transporte em H1, encapsulará cada segmento em um datagrama (isto é, em um pacote de camada de rede) e então dará início à jornada dos datagramas até seu destino, isto é, enviará os datagramas para seu roteador vizinho, R1. No hospedeiro receptor (H2), a camada de rede receberá os datagramas de seu roteador vizinho R2, extrairá os segmentos de camada de transporte e os entregará à camada de transporte em H2. O papel primordial dos roteadores é repassar datagramas de enlaces de entrada para enlaces de saída. Note que os roteadores da Figura 4.1 são mostrados com a pilha de protocolos truncada, isto é, sem as camadas superiores acima da camada de rede, porque (exceto para finalidades de controle) roteadores não rodam protocolos de camada de transporte e de aplicação como os que examinamos nos capítulos 2 e 3.

4.1.1 Repasse e roteamento

Assim, o papel da camada de rede é aparentemente simples — transportar pacotes de um hospedeiro remetente a um hospedeiro destinatário. Para fazê-lo, duas importantes funções da camada de rede podem ser identificadas:

- *Repasse*. Quando um pacote chega ao enlace de entrada de um roteador, este deve conduzi-lo até o enlace de saída apropriado. Por exemplo, um pacote proveniente do hospedeiro H1 que chega ao roteador R1 deve ser repassado ao roteador seguinte por um caminho até H2. Na Seção 4.3 examinaremos o interior de um roteador e investigaremos como um pacote é realmente repassado de um enlace de entrada de um roteador até um enlace de saída.
- *Roteamento*. A camada de rede deve determinar a rota ou caminho tomado pelos pacotes ao fluírem de um remetente a um destinatário. Os algoritmos que calculam esses caminhos são denominados **algoritmos de roteamento**. Um algoritmo de roteamento determinaria, por exemplo, o caminho ao longo do qual os pacotes fluiriam de H1 para H2.

Os termos *repasse* e *roteamento* são usados indistintamente por autores que discutem a camada de rede. Neste livro, usaremos esses termos com maior exatidão. *Repasse* refere-se à ação local realizada por um roteador para transferir um pacote da interface de um enlace de entrada para a interface de enlace de saída apropriada. *Roteamento* refere-se ao processo de âmbito geral da rede que determina os caminhos fim-a-fim que os pacotes percorrem desde a fonte até o destino. Para usar uma viagem como analogia, voltemos àquele nosso viajante da Seção 1.3.2, que vai da Pensilvânia à Flórida. Durante a viagem, nosso motorista passa por muitos cruzamentos de rodovias em sua rota até a Flórida. Podemos imaginar o repasse como o processo



gura 4.1 A camada de rede

e passar por um único cruzamento: um carro chega ao cruzamento, pede informações sobre como che-ar ao próximo em sua rota e pega a rodovia que se dirige ao cruzamento desejado. Podemos imaginar o oteamento como o processo de planejamento da viagem da Pensilvânia até a Flórida: antes de partir, o notorista consultou um mapa e escolheu um dos muitos caminhos possíveis. Cada um desses caminhos onsiste em uma série de trechos de rodovias conectados por cruzamentos. Na primeira parte deste capí-ulo, focalizaremos tópicos de camada de rede relativos ao repasse; em seguida, voltaremos nossa atenção o roteamento.

Cada roteador tem uma **tabela de repasse**. Um roteador repassa um pacote examinando o valor de m campo no cabeçalho do pacote que está chegando e então utiliza esse valor para indexar sua tabela de epasse. O resultado da tabela de repasse indica para qual das interfaces de enlace do roteador o pacote eve ser repassado. Dependendo do protocolo de camada de rede, esse valor no cabeçalho do pacote pode er o endereço de destino do pacote ou uma indicação da conexão à qual ele pertence. A Figura 4.2 dá um xemplo.

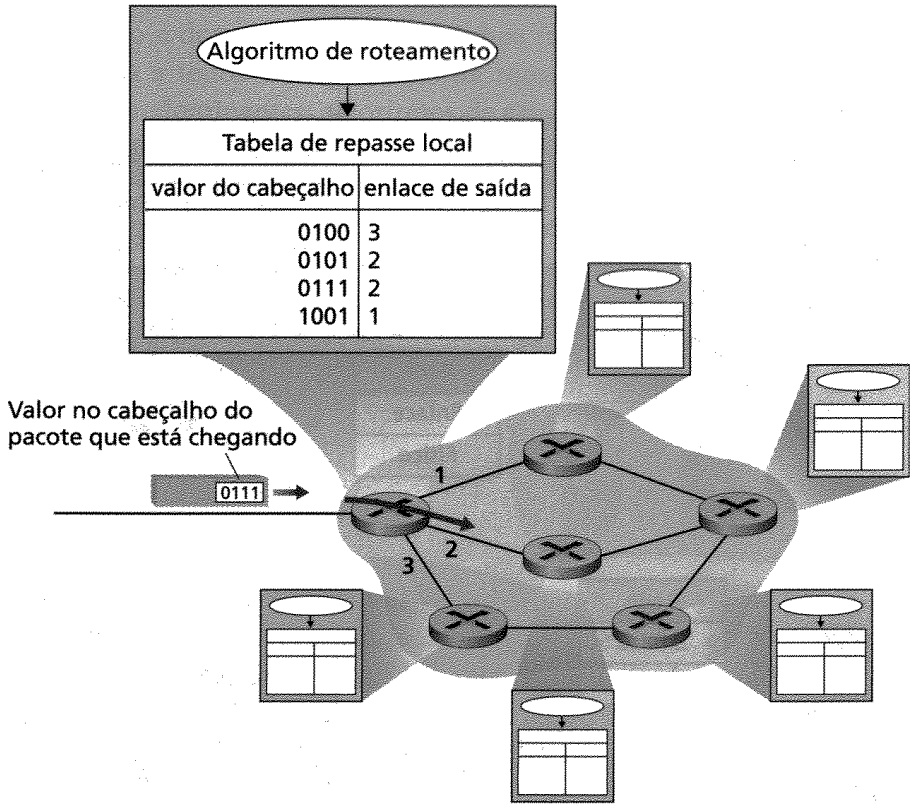


Figura 4.2 Algoritmos de roteamento determinam valores em tabelas de repasse

Na Figura 4.2, um pacote cujo valor no campo de cabeçalho é 0111 chega a um roteador. O roteador o indexa em sua tabela de repasse, determina que a interface de enlace de saída para esse pacote é a interface 2 e, então, repassa o pacote internamente à interface 2. Na Seção 4.3 examinaremos o interior de um roteador e estudaremos a função de repasse muito mais detalhadamente.

Agora você deve estar imaginando como são configuradas as tabelas de repasse nos roteadores. Essa é uma questão crucial, que expõe a importante interação entre roteamento e repasse. Como ilustrado na Figura 4.2, o algoritmo de roteamento determina os valores que são inseridos nas tabelas de repasse dos roteadores. O algoritmo de roteamento pode ser centralizado (por exemplo, com um algoritmo que roda em um local central e descarrega informações de roteamento a cada um dos roteadores) ou descentralizado (isto é, com um pedaço do algoritmo de roteamento distribuído funcionando em cada roteador). Em qualquer dos casos, um roteador recebe mensagens de protocolo de roteamento que são utilizadas para configurar sua tabela de repasse. As finalidades distintas e diferentes das funções de repasse e roteamento podem ser mais bem esclarecidas considerando o caso hipotético (e não realista, mas tecnicamente viável) de uma rede na qual todas as tabelas de repasse são configuradas diretamente por operadores de rede humanos, fisicamente presentes nos roteadores. Nesse caso, não seria preciso *nenhum* protocolo de roteamento! É claro que os operadores humanos precisariam interagir uns com os outros para garantir que as tabelas fossem configuradas de tal modo que os pacotes chegassem a seus destinos pretendidos. Também é provável que uma configuração humana seria mais propensa a erro e muito mais lenta do que um protocolo de roteamento para reagir a mudanças na topologia da rede. Portanto, sorte nossa que todas as redes têm uma função de repasse *e também* uma função de roteamento!

Enquanto estamos no tópico da terminologia, é interessante mencionar dois outros termos que também são utilizados indistintamente, mas que usaremos com maior cuidado. Reservaremos o termo *comutador de pacotes* para designar um dispositivo geral de comutação de pacotes que transfere um pacote de interface de enlace de entrada para interface de enlace de saída conforme o valor que está em um campo no cabeçalho do pacote. Alguns comutadores de pacotes, denominados **comutadores de camada de enlace** (que vere-

mos no Capítulo 5), baseiam a decisão de repasse no valor que está no campo da camada de enlace. Outros, denominados roteadores, baseiam sua decisão de repasse no valor que está no campo de camada de rede. (Para dar real valor a essa importante distinção, seria interessante você ler novamente a Seção 1.7.2, em que discutimos datagramas de camada de rede e quadros de camada de enlace e as relações entre eles.) Visto que o foco deste capítulo é a camada de rede, usaremos o termo *roteador* no lugar de *comutador de pacotes*. Usaremos o termo *roteador* até mesmo quando falarmos sobre comutadores de pacotes em redes de circuitos virtuais (que discutiremos em breve).

Estabelecimento de conexão

Acabamos de dizer que a camada de rede tem duas funções importantes, repasse e roteamento. Mas logo veremos que em algumas redes de computadores há uma terceira função importante de camada de rede, a saber, o **estabelecimento de conexão**. Lembre-se de que, quando estudamos o TCP, verificamos que é necessária uma apresentação de três vias antes de os dados realmente poderem fluir do remetente ao destinatário. Isso permite que o remetente e o destinatário estabeleçam a informação de estado necessária (por exemplo, número de seqüência e tamanho inicial da janela de controle de fluxo). De maneira análoga, algumas arquiteturas de camada de rede — por exemplo, ATM, *frame-relay*, X.25, mas não a Internet — exigem que roteadores ao longo do caminho escolhido desde a fonte até o destino troquem mensagens entre si com a finalidade de estabelecer estado antes que pacotes de dados de camada de rede dentro de uma dada conexão fonte-destino possam começar a fluir. Na camada de rede, esse processo é denominado estabelecimento de conexão. Examinaremos estabelecimento de conexão na Seção 4.2.

4.1.2 Modelos de serviço de rede

Antes de examinar a camada de rede, vamos tomar uma perspectiva mais ampla e considerar os diferentes tipos de serviço que poderiam ser oferecidos pela camada de rede. Quando a camada de transporte em um hospedeiro remetente transmite um pacote para dentro da rede (isto é, passa o pacote para a camada de rede do hospedeiro remetente), ela pode contar com a camada de rede para entregar o pacote no destino? Quando são enviados vários pacotes, eles serão entregues à camada de transporte no hospedeiro destinatário na ordem em que foram enviados? A quantidade de tempo decorrido entre duas transmissões de pacotes seqüenciais será a mesma quantidade de tempo decorrido entre suas recepções? A rede fornecerá algum tipo de informação sobre congestionamento na rede? Qual é o modelo (propriedades) abstrato do canal que conecta a camada de transporte nos hospedeiros remetente e destinatário? As respostas a essas e a outras perguntas são determinadas pelo modelo de serviço oferecido pela camada de rede. O **modelo de serviço de rede** define as características do transporte de dados fim-a-fim entre uma borda da rede e a outra, isto é, entre sistemas finais remetente e destinatário.

Vamos considerar agora alguns serviços possíveis que a camada de rede poderia prover. Quando a camada de transporte em um hospedeiro remetente passa um pacote para a camada de rede, alguns serviços específicos que poderiam ser oferecidos pela camada de rede são:

- *Entrega garantida*. Esse serviço assegura que o pacote mais cedo ou mais tarde chegará a seu destino.
- *Entrega garantida com atraso limitado*. Esse serviço não somente assegura a entrega de um pacote, mas também a entrega com um atraso hospedeiro a hospedeiro limitado e especificado (por exemplo, dentro de 100 milissegundos).

Além disso, há outros serviços que podem ser providos a um *fluxo de pacotes* entre uma fonte e um destino determinados, como os seguintes:

- *Entrega de pacotes na ordem*. Este serviço garante que pacotes chegarão ao destino na ordem em que foram enviados.

Arquitetura da rede	Modelo de serviço	Garantia de largura de banda	Garantia contra perda	Ordenamento	Temporização	Indicação de congestionamento
Internet	Melhor esforço	Nenhuma	Nenhuma	Qualquer ordem possível	Não mantida	Nenhuma
ATM	CBR	Taxa constante garantida	Sim	Na ordem	Mantida	Não ocorrerá congestionamento
ATM	ABR	Mínima garantida	Nenhuma	Na ordem	Não mantida	Indicação de congestionamento

Tabela 4.1 Modelos de serviço das redes Internet, ATM CBR e ATM ABR

Largura de banda mínima garantida. Esse serviço de camada de rede emula o comportamento de um enlace de transmissão com uma taxa de bits especificada (por exemplo, 1 Mbps) entre hospedeiros remetentes e destinatários (mesmo que o caminho fim-a-fim real passe por vários enlaces físicos). Contanto que o hospedeiro remetente transmita bits (como parte de pacotes) a uma taxa abaixo da taxa de bits especificada, nenhum pacote será perdido e cada um chegará dentro de um atraso hospedeiro a hospedeiro previamente especificado (por exemplo, dentro de 40 milissegundos).

Jitter máximo garantido. Este serviço garante que a quantidade de tempo entre a transmissão de dois pacotes sucessivos no remetente seja igual à quantidade de tempo entre o recebimento dos dois pacotes no destino (ou que esse espaçamento não mude mais do que algum valor especificado).

Essa é uma lista apenas parcial de serviços que uma camada de rede pode prover — há incontáveis variações possíveis.

A camada de rede da Internet fornece um único modelo de serviço, conhecido como **serviço de melhor esforço**. Consultando a Tabela 4.1, pode parecer que *serviço de melhor esforço* seja um eufemismo para *absolutamente nenhum serviço.* Com o serviço de melhor esforço, não há garantia de que a temporização entre pacotes seja preservada, não há garantia de que os pacotes sejam recebidos na ordem em que foram enviados e não há garantia da eventual entrega dos pacotes transmitidos. Dada essa definição, uma rede que não entregasse *nenhum* pacote ao destinatário satisfaria a definição de serviço de entrega de melhor esforço. (Na verdade, a Internet pública de hoje pode às vezes parecer um exemplo de rede que faz exatamente isso!) Contudo, como discutiremos adiante, há razões sólidas para esse modelo minimalista de serviço de camada de rede. Examinaremos outros modelos de serviço de Internet ainda em evolução no Capítulo 7.

Outras arquiteturas de rede definiram e implementaram modelos de serviço que vão além do serviço de melhor esforço da Internet. Por exemplo, a arquitetura de rede ATM [ATM Forum 2004; Black, 1995] habilita vários modelos de serviço, o que significa que, dentro da mesma rede, podem ser oferecidas conexões diferentes com classes de serviço diferentes. Discutir o modo como uma rede ATM provê esses serviços vai muito além do escopo deste livro; nossa meta aqui é apenas salientar que existem alternativas ao modelo de melhor esforço da Internet. Dois dos modelos mais importantes de serviço ATM são os serviços de taxa constante e de taxa disponível.

Serviço de rede de taxa constante de bits (constant bit rate — *CBR*). Esse foi o primeiro modelo de serviço ATM a ser padronizado, refletindo o interesse imediato das empresas de telefonia por esse serviço e a adequabilidade do serviço CBR para transmitir tráfego de áudio e vídeo de taxa constante de bits. O objetivo do serviço CBR é conceitualmente simples: prover um fluxo de pacotes (conhecidos como células na terminologia ATM) com uma tubulação virtual cujas propriedades são iguais às de um hipotético enlace de transmissão dedicado de largura de banda fixa entre os hospedeiros remetente e destinatário. Com serviço CBR, um fluxo de células ATM é carregado através da rede de modo tal que garanta que o atraso fim-a-fim de uma célula, a variabilidade desse

atraso (isto é, o *jitter*) e a fração de células perdidas ou entregues atrasadas sejam menores do que valores especificados. Esses valores são acertados entre o hospedeiro remetente e a rede ATM quando a conexão CBR é estabelecida pela primeira vez.

- *Serviço de rede de taxa de bits disponível (available bit rate — ABR).* Como o serviço oferecido pela Internet é "de melhor esforço", o ATM ABR pode ser caracterizado como um "serviço de melhor esforço ligeiramente melhorado". Como acontece com o modelo de serviço da Internet, também com o serviço ABR pode haver perda de células. Mas, diferentemente da Internet, as células não podem ser reordenadas (embora possam ser perdidas) e é garantida uma taxa mínima de transmissão de células (*minimum cell transmission rate* — MCR) para uma conexão que está usando o serviço ABR. Se, contudo, a rede dispuser de suficientes recursos livres em um dado momento, um remetente também poderá enviar células com sucesso a uma taxa mais alta do que a MCR.

Além disso, como vimos na Seção 3.6, o serviço ATM ABR pode prover realimentação ao remetente (em termos de um bit de notificação de congestionamento ou de uma taxa de envio explícita), que controla o modo como o remetente ajusta sua taxa entre a MCR e uma taxa de pico admissível.

4.2 Redes de circuitos virtuais e de datagramas

Lembre-se de que, no Capítulo 3, dissemos que a camada de transporte pode oferecer às aplicações serviço não orientado para conexão ou serviço orientado para conexão. Por exemplo, a camada de transporte da Internet oferece a cada aplicação uma alternativa entre dois serviços: UDP, um serviço não orientado para conexão; ou TCP, um serviço orientado para conexão. De modo semelhante, uma camada de rede também pode oferecer qualquer desses dois serviços. Serviços de camada de rede orientados para conexão e não orientados para conexão são, sob muitos aspectos, semelhantes a esses mesmos serviços providos pela camada de transporte. Por exemplo, um serviço de camada de rede orientado para conexão começa com uma apresentação entre os hospedeiros de origem e de destino; e um serviço de camada de rede não orientado para conexão não tem nenhuma apresentação preliminar.

Embora os serviços de camada de rede orientados para conexão e não orientados para conexão tenham algumas semelhanças com os mesmos serviços oferecidos pela camada de transporte, há diferenças cruciais:

- Na camada de rede, são serviços de hospedeiro a hospedeiro providos pela camada de rede à camada de transporte. Na camada de transporte, são serviços de processo a processo fornecidos pela camada de transporte à camada de aplicação.
- Em todas as arquiteturas importantes de redes de computadores existentes até agora (Internet, ATM, *frame-relay* e assim por diante), a camada de rede oferece um serviço entre hospedeiros não orientado para conexão, ou um serviço entre hospedeiros orientado para conexão, mas não ambos. Redes de computadores que oferecem apenas um serviço orientado para conexão na camada de rede são denominadas **redes de circuitos virtuais** (**redes CV**); redes de computadores que oferecem apenas um serviço não orientado para conexão na camada de rede são denominadas **redes de datagramas**.
- As implementações de serviço orientado para conexão na camada de transporte e de serviço de conexão na camada de rede são fundamentalmente diferentes. No capítulo anterior vimos que o serviço de camada de transporte orientado para conexão é implementado na borda da rede nos sistemas finais; em breve veremos que o serviço da camada de rede orientado para conexão é implementado nos roteadores no núcleo da rede, bem como nos sistemas finais.

Redes de circuitos virtuais e redes de datagramas são duas classes fundamentais de redes de computadores. Elas utilizam informações muito diferentes para tomar suas decisões de repasse. Vamos agora examinar suas implementações mais de perto.

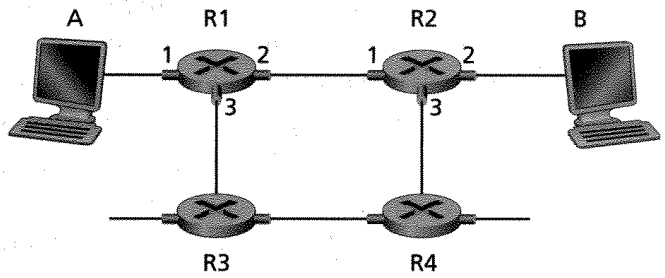


Figura 4.3 Uma rede de circuitos virtuais simples

4.2.1 Redes de circuitos virtuais

Já vimos que a Internet é uma rede de datagramas. Entretanto, muitas arquiteturas de rede alternativas — entre elas as das redes ATM, *frame relay* e X.25 — são redes de circuitos virtuais e, portanto, usam conexões na camada de rede. Essas conexões de camada de rede são denominadas **circuitos virtuais** (**CVs**). Vamos considerar agora como um serviço de CVs pode ser implantado em uma rede de computadores.

Um circuito virtual (CV) consiste em (1) um caminho (isto é, uma série de enlaces e roteadores) entre hospedeiros de origem e de destino, (2) números de CVs, um número para cada enlace ao longo do caminho e (3) registros na tabela de repasse em cada roteador ao longo do caminho. Um pacote que pertence a um circuito virtual portará um número de CV em seu cabeçalho. Como um circuito virtual pode ter um número de CV diferente em cada enlace, cada roteador interveniente deve substituir o número de CV de cada pacote em trânsito por um novo número. Esse número novo do CV é obtido da tabela de repasse.

Para ilustrar o conceito, considere a rede mostrada na Figura 4.3. Os números ao lado dos enlaces de R1 na Figura 4.3 são os números das interfaces de enlaces. Suponha agora que o hospedeiro A solicite à rede que estabeleça um CV entre ele e o hospedeiro B. Suponha ainda que a rede escolha o caminho A–R1–R2–B e atribua números de CV 12, 22, 32 aos três enlaces nesse caminho para esse circuito virtual. Nesse caso, quando um pacote nesse CV sai do hospedeiro A, o valor no campo do número de CV é 12; quando sai de R1, o valor é 22, e, quando sai de R2, o valor é 32.

Como o roteador determina o novo número de CV para um pacote que passa por ele? Para uma rede de CV, a tabela de repasse de cada roteador inclui a tradução de número de CV; por exemplo, a tabela de repasse de R1 pode ser algo parecido com o seguinte:

Interface de entrada	Nº do CV de entrada	Interface de saída	Nº do CV de saída
1	12	2	22
2	63	1	18
3	7	2	17
1	97	3	87
—	—	—	—

Sempre que um novo CV é estabelecido através de um roteador, um registro é adicionado à tabela de repasse. De maneira semelhante, sempre que um CV termina, são removidos os registros apropriados em cada tabela ao longo de seu caminho.

É bem possível que você esteja pensando por que um pacote não conserva o mesmo número de CV em cada um dos enlaces ao longo de sua rota. Isso ocorre por dois motivos: primeiro, substituir o número de enlace em enlace reduz o comprimento do campo do CV no cabeçalho do pacote. Segundo, e mais importante, o estabelecimento de um CV é consideravelmente simplificado se for permitido um número diferente de CV em cada enlace ao longo do caminho do circuito virtual. Especificamente, com vários números de

CV, cada enlace do caminho pode escolher um número de CV independentemente dos números de CV escolhidos em outros enlaces ao longo do caminho. Se fosse exigido um mesmo número de CV para todos os enlaces ao longo do caminho, os roteadores teriam de trocar e processar um número substancial de mensagens para escolher o número de CV a ser usado para uma conexão (por exemplo, um número que não está sendo usado por nenhum outro CV nesses roteadores).

Em uma rede de circuitos virtuais, os roteadores da rede devem manter **informação de estado de conexão** para as conexões em curso. Especificamente, cada vez que uma nova conexão for estabelecida através de um roteador, um novo registro de conexão deve ser adicionado à tabela de repasse do roteador. E, sempre que uma conexão for desativada, um registro deve ser removido da tabela. Observe que, mesmo que não haja tradução de números de CVs, ainda assim é necessário manter informação de estado de conexão que associe números de CVs com números das interfaces de saída. A questão de um roteador manter ou não informação de estado de conexão para cada conexão em curso é crucial — e retornaremos a ela várias vezes neste livro.

Há três fases que podem ser identificadas em um circuito virtual:

- *Estabelecimento de CV.* Durante a fase de estabelecimento, a camada de transporte remetente contata a camada de rede, especifica o endereço do receptor e espera até a rede estabelecer o CV. A camada de rede determina o caminho entre remetente e destinatário, ou seja, a série de enlaces e roteadores pelos quais todos os pacotes do CV trafegarão. A camada de rede também determina o número de CV para cada enlace ao longo do caminho e, finalmente, adiciona um registro na tabela de repasse em cada roteador ao longo do caminho. Durante o estabelecimento do CV, a camada de rede pode também reservar recursos (por exemplo, largura de banda) ao longo do caminho do CV.
- *Transferência de dados.* Como mostra a Figura 4.4, tão logo estabelecido o CV, pacotes podem começar a fluir ao longo do CV.
- *Encerramento do CV.* O encerramento começa quando o remetente (ou o destinatário) informa à camada de rede seu desejo de desativar o CV. A camada de rede então tipicamente informará o sistema final do outro lado da rede do término de conexão e atualizará as tabelas de repasse em cada um dos roteadores de pacotes no caminho para indicar que o CV não existe mais.

Há uma distinção sutil, mas importante, entre estabelecimento de CV na camada de rede e estabelecimento de conexão na camada de transporte (por exemplo, a apresentação TCP de três vias que estudamos no Capítulo 3). Estabelecer conexão na camada de transporte envolve apenas os dois sistemas finais. Durante o estabelecimento da conexão na camada de transporte, os dois sistemas finais determinam os parâmetros (por exemplo, número de seqüência inicial e tamanho da janela de controle de fluxo) de sua conexão de camada de transporte. Embora os dois sistemas finais fiquem cientes da conexão de camada de transporte, os roteadores dentro da rede ficam completamente alheios a ela. Por outro lado, com uma camada de rede de CV, *os roteadores ao longo do caminho entre os dois sistemas finais estão envolvidos no estabelecimento de CV e cada roteador fica totalmente ciente de todos os CVs que passam por ele.*

As mensagens que os sistemas finais enviam à rede para iniciar ou encerrar um CV e as mensagens passadas entre os roteadores para estabelecer o CV (isto é, modificar estado de conexão em tabelas de roteadores) são conhecidas como **mensagens de sinalização** e os protocolos usados para trocar essas mensagens freqüentemente são denominados **protocolos de sinalização**. O estabelecimento de CV está ilustrado na Figura 4.4. Não abordaremos protocolos de sinalização de CVs neste livro; [Black, 1997] apresenta uma discussão geral sobre sinalização em redes orientadas para conexão e [ITU-T Q.2931, 1994] mostra a especificação do protocolo de sinalização Q.2931 do ATM.

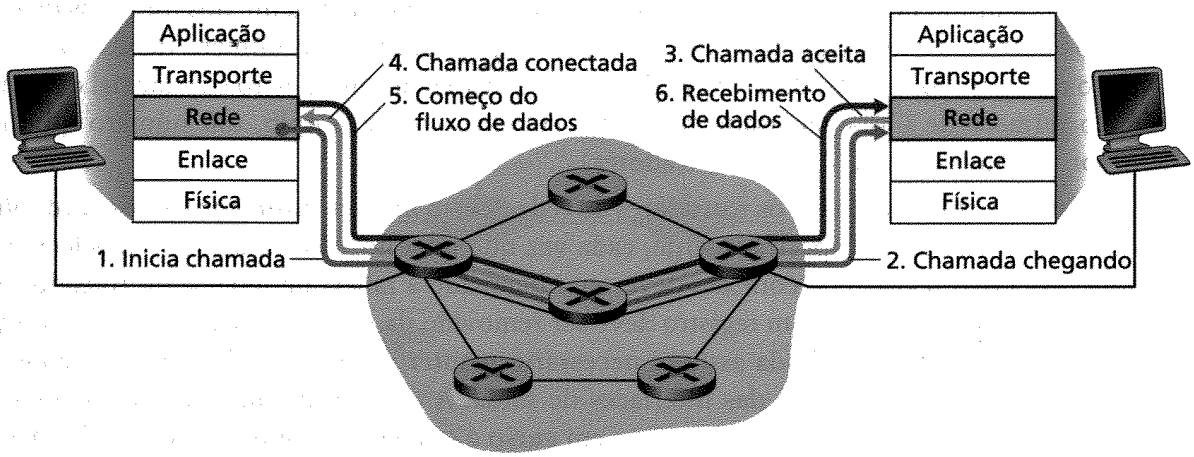


Figura 4.4 Estabelecimento de circuito virtual

4.2.2 Redes de datagramas

Em uma **rede de datagramas**, toda vez que um sistema final quer enviar um pacote, ele marca o pacote com o endereço do sistema final de destino e então o envia para dentro da rede. Como mostra a Figura 4.5, isso é feito sem o estabelecimento de nenhum CV. Roteadores em uma rede de datagramas não mantêm nenhuma informação de estado sobre CVs (porque não há nenhum CV!).

Ao ser transmitido da fonte ao destino, um pacote passa por uma série de roteadores. Cada um desses roteadores usa o endereço de destino do pacote para repassá-lo. Especificamente, cada roteador tem uma tabela de repasse que mapeia endereços de destino para interfaces de enlaces; quando um pacote chega ao roteador, este usa o endereço de destino do pacote para procurar a interface de enlace de saída apropriada na tabela de repasse. Então, o roteador transmite o pacote para aquela interface de enlace de saída.

Para entender melhor a operação de consulta, vamos examinar um exemplo específico. Suponha que todos os endereços de destino tenham 32 bits (que, por acaso, é exatamente o comprimento do endereço de destino em um datagrama IP). Uma implementação de força bruta da tabela de repasse teria um registro para cada endereço de destino possível. Porém, como há mais de quatro bilhões de endereços possíveis, essa opção está totalmente fora de questão — exigiria uma tabela de repasse imensa.

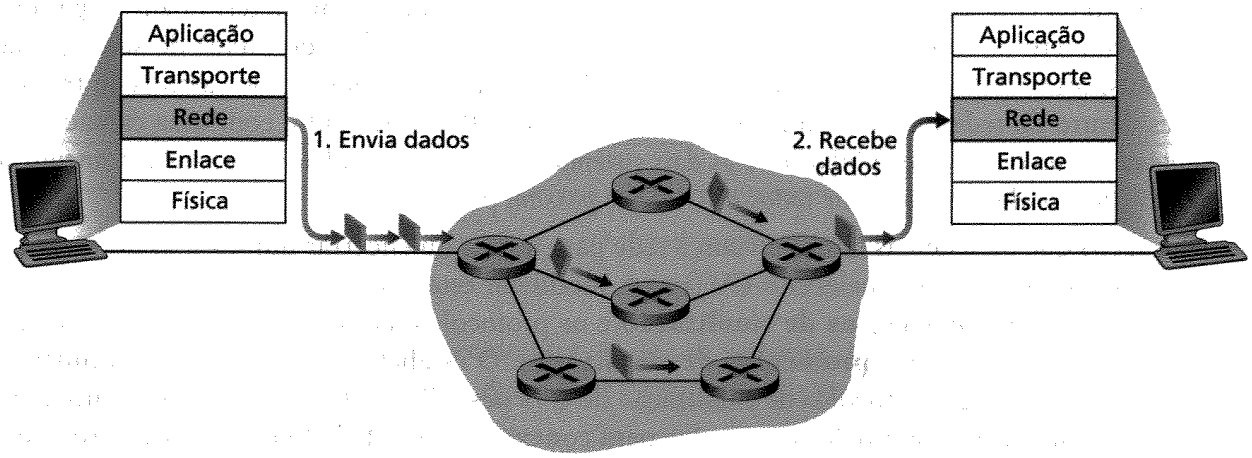


Figura 4.5 Rede de datagramas

Vamos supor ainda mais que nosso roteador tenha quatro enlaces numerados de 0 a 3, e que os pacotes devem ser repassados para as interfaces de enlace como mostrado a seguir:

Faixa de Endereços de Destino	Interface de enlace
11001000 00010111 00010000 00000000	
até	0
11001000 00010111 00010111 11111111	
11001000 00010111 00011000 00000000	
até	1
11001000 00010111 00011000 11111111	
11001000 00010111 00011001 00000000	
até	2
11001000 00010111 00011111 11111111	
senão	3

Fica claro, por este exemplo, que não é necessário ter quatro bilhões de registros na tabela de repasse do roteador. Poderíamos, por exemplo, ter a seguinte tabela de repasse com apenas quatro registros:

Prefixo do endereço	Interface de enlace
11001000 00010111 00010	0
11001000 00010111 00011000	1
11001000 00010111 00011	2
senão	3

Com esse tipo de tabela de repasse, o roteador compara um **prefixo** do endereço de destino do pacote com os registros na tabela; se houver uma concordância de prefixos, o roteador transmite o pacote para o enlace associado àquele prefixo compatível. Por exemplo, suponha que o endereço de destino do pacote seja 11001000 00010111 00010110 10100001; como o prefixo de 21 bits desse endereço é igual ao primeiro registro na tabela, o roteador transmite o pacote para a interface de enlace 0. Se o prefixo do pacote não combinar com nenhum dos três primeiros registros, o roteador envia o pacote para a interface 3. Embora isso pareça bastante simples, há aqui uma sutileza importante. Você talvez tenha notado que existe a possibilidade de um endereço de destino combinar com mais de um registro. Por exemplo, os primeiros 24 bits do endereço 11001000 00010111 00011000 10101010 combinam com o segundo registro na tabela e os primeiros 21 bits do endereço combinam com o terceiro registro da tabela. Quando há várias concordâncias de prefixos, o roteador usa a **regra da concordância do prefixo mais longo**, isto é, encontra o registro cujo prefixo tem mais bits compatíveis com os bits do endereço do pacote e envia o pacote à interface de enlace associada com esse prefixo mais longo compatível.

Evidentemente, para que essa compatibilização com o prefixo mais longo seja efetiva, cada interface de enlace de saída deve ser responsável pelo repasse de grandes blocos de endereços de destino contíguos. Veremos, na Seção 4.4, que endereços de Internet normalmente são designados hierarquicamente, de modo que essa propriedade da contigüidade prevalece nas tabelas de repasse de grande parte dos roteadores. Mesmo assim, a comunidade de pesquisadores da Internet demonstra uma certa preocupação com o aumento das lacunas nos espaços de endereços, que resultam em blocos contíguos cada vez menores e tabelas de repasse cada vez maiores. (Veja [Maennel, 2002], [RFC 3221] e o quadro Princípios na Prática na Seção 4.4.)

Embora em redes de datagramas os roteadores não mantenham nenhuma informação de estado de conexão, ainda assim mantêm informação de estado de repasse em suas tabelas de repasse. Todavia, a escala temporal da mudança dessas informações de estado é relativamente lenta. Na verdade, as tabelas de repasse em uma rede de datagramas são modificadas pelos algoritmos de roteamento que normalmente atualizam uma tabela de repasse em intervalos de um a cinco minutos, aproximadamente. A tabela de repasse de um roteador em uma rede de CVs é modificada sempre que é estabelecida uma nova conexão através do roteador ou sempre que uma conexão existente é desativada. Em um roteador de backbone de nível 1, isso poderia acontecer facilmente em uma escala temporal de microssegundos.

Como em redes de datagramas as tabelas de repasse podem ser modificadas a qualquer momento, uma série de pacotes enviados de um sistema final para outro pode seguir caminhos diferentes através da rede e muitos podem chegar fora da ordem. [Paxson, 1997] e [Jaiswal, 2003] apresentam interessantes estudos de medição da reordenação de pacotes e de outros fenômenos na Internet pública.

4.2.3 Origens das redes de circuitos virtuais e de datagramas

A evolução das redes de datagramas e de circuitos virtuais reflete as origens dessas redes. A idéia de um circuito virtual como princípio fundamental de organização tem suas raízes no mundo da telefonia (que usa circuitos reais). Como em redes de CV os roteadores mantêm estado de chamada e estado por chamada, esse tipo de rede é consideravelmente mais complexo do que uma rede de datagramas. ([Molinero, 2002] faz uma comparação interessante entre as complexidades de redes de comutação de circuitos e de comutação de pacotes.) Isso também está de acordo com a herança da telefonia. A complexidade das redes telefônicas estava, necessariamente, dentro da própria rede, já que elas conectavam sistemas finais não inteligentes tais como telefones de disco. (Para os mais jovens que não o conhecem, um telefone de disco é um telefone analógico [isto é, não digital] sem teclas —, tem somente um mostrador com números e um dispositivo mecânico denominado disco.)

Por sua vez, a Internet como uma rede de datagramas surgiu da necessidade de conectar computadores. Como esses sistemas finais são mais sofisticados, os arquitetos da Internet preferiram construir um modelo de serviço de camada de rede o mais simples possível. Como já vimos nos Capítulos 2 e 3, funcionalidades adicionais (por exemplo, entrega na ordem, transferência confiável de dados, controle de congestionamento e resolução de nomes DNS) são implementadas em uma camada mais alta, nos sistemas finais. Isso inverte o modelo da rede de telefonia, com algumas conseqüências interessantes:

- O modelo de serviço de camada de rede resultante da Internet, que oferece garantias mínimas (nenhuma!) de serviço (e que, por conseguinte impõe exigências mínimas sobre a camada de rede) também facilita a interconexão de redes que usam tecnologias de camada de enlace muito diferentes (por exemplo, satélite, Ethernet, fibra ou rádio) e cujas taxas de transmissão e características de perda também são muito diferentes. Vamos abordar detalhadamente a interconexão de redes IP na Seção 4.4.
- Como vimos no Capítulo 2, aplicações como e-mail, a Web e até mesmo um serviço centrado na camada de rede como o DNS são implementadas em hospedeiros (servidores) na borda da rede. A capacidade de adicionar um novo serviço simplesmente ligando um hospedeiro à rede e definindo um novo protocolo de camada de aplicação (como o HTTP) permitiu que novas aplicações como a Web fossem distribuídas pela Internet em um período notavelmente curto.

Como veremos no Capítulo 7, há uma considerável discussão na comunidade da Internet sobre como a arquitetura da camada de rede da Internet deve evoluir para suportar serviços de tempo real como multimídia. Uma comparação interessante entre a arquitetura de rede ATM orientada para CVs e a próxima geração proposta para a arquitetura da Internet é dada em [Crowcroft, 1995].

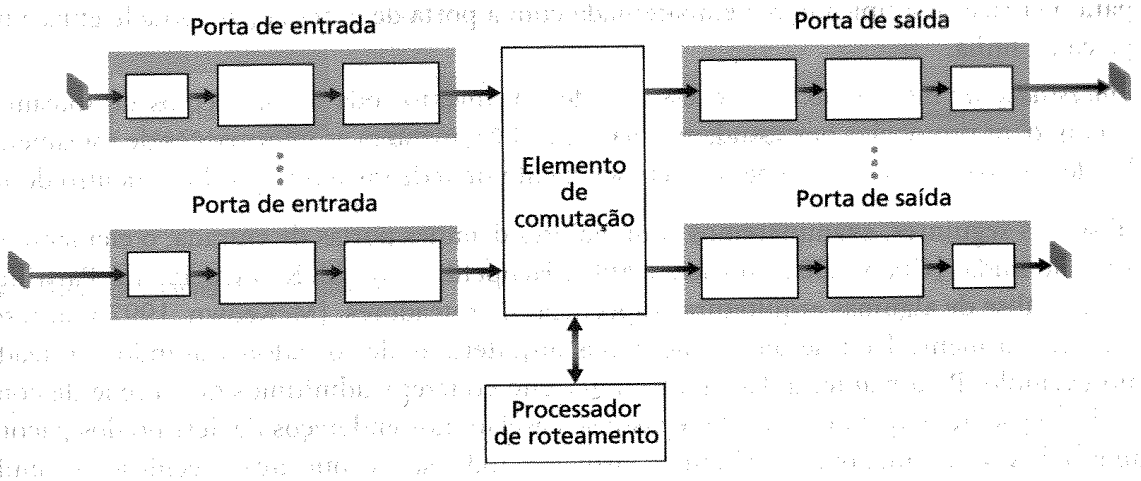


Figura 4.6 Arquitetura de roteador

4.3 O que há dentro de um roteador?

Agora que já tivemos uma visão geral das funções e serviços da camada de rede, voltaremos nossa atenção para a função de repasse — a transferência, propriamente dita, de pacotes dos enlaces de entrada até os enlaces de saída adequados de um roteador. Já estudamos algumas questões de repasse na Seção 4.2, a saber, endereçamento e compatibilização com o prefixo mais longo. Nesta seção examinaremos arquiteturas específicas de roteadores para transferir pacotes de enlaces de entrada para enlaces de saída. Neste caso, nossa análise é necessariamente breve, pois seria necessário um curso completo para tratar do projeto de roteadores com profundidade. Conseqüentemente, faremos um esforço especial nesta seção para indicar material que apresente esse tópico com mais profundidade. Mencionamos, de passagem, que os pesquisadores e profissionais de redes de computadores usam as palavras *repasse* e *comutação* indistintamente; nós usaremos ambos os termos neste livro.

Uma visão de alto nível da arquitetura de um roteador genérico é mostrada na Figura 4.6. Quatro componentes de um roteador podem ser identificados:

- *Portas de entrada.* A porta de entrada tem diversas funções. Ela realiza as funções de camada física (a caixa mais à esquerda da porta de entrada e a caixa mais à direita da porta de saída na Figura 4.6) de terminar um enlace físico de entrada em um roteador. Realiza também as funções de camada de enlace (representadas pelas caixas do meio nas portas de entrada e de saída) necessárias para interoperar com as funções da camada de enlace do outro lado do enlace de entrada. Realiza ainda uma função de exame e de repasse (a caixa mais à direita da porta de entrada e a caixa mais à esquerda da porta de saída), de modo que um pacote repassado ao elemento de comutação do roteador surja na porta de saída apropriada. Pacotes de controle (por exemplo, pacotes carregando informações de protocolo de roteamento) são repassados de uma porta de entrada até o processador de roteamento. Na prática, várias portas são freqüentemente reunidas em uma única **placa de linha** no interior de um roteador.
- *Elemento de comutação.* O elemento de comutação conecta as portas de entrada do roteador às suas portas de saída. Ele está integralmente contido no interior do roteador — uma rede dentro de um roteador de rede!
- *Portas de saída.* Uma porta de saída armazena os pacotes que foram repassados a ela através do elemento de comutação e, então, os transmite até o enlace de saída. Assim, a porta de saída realiza o inverso da funcionalidade da camada de enlace e da camada física da porta de entrada. Quando um enlace é bidirecional (isto é, carrega um tráfego em ambas as direções), uma porta de saída

para o enlace será tipicamente emparelhada com a porta de entrada para aquele enlace na mesma placa de linha.

Processador de roteamento. O processador de roteamento roda os protocolos de roteamento (por exemplo, os protocolos que estudamos na Seção 4.6), mantém as informações de roteamento e tabelas de repasse e executa funções de gerenciamento de rede (veja o Capítulo 9) dentro do roteador.

Nas subseções seguintes, examinaremos com maiores detalhes portas de entrada, o elemento de comutação e portas de saída. [Chao, 2001; Turner, 1998; Giacopelli, 1990; McKeown, 1997a; Partridge, 1998] fornecem uma análise de algumas arquiteturas específicas de roteadores. [McKeown, 1997b] apresenta uma visão geral particularmente fácil de ler de modernas arquiteturas de roteadores usando o roteador Cisco 12000 como exemplo. Para manter a discussão em terreno concreto, admitimos que a rede de computadores é uma rede de pacotes e que decisões de repasse se baseiam nos endereços de destino dos pacotes (e não em um número de CV em uma rede de circuitos virtuais). Todavia, os conceitos e técnicas são similares aos de uma rede de circuitos virtuais.

4.3.1 Portas de entrada

Uma visão mais detalhada da funcionalidade de porta da entrada é apresentada na Figura 4.7. Como discutido anteriormente, as funções de terminação de linha e de processamento de enlace realizadas pela porta de entrada implementam as camadas física e de enlace associadas a um enlace de entrada individual do roteador. O módulo examinar/repassar da porta de entrada é fundamental para a função de repasse do roteador. Em muitos roteadores, é aqui que o roteador determina a porta de saída para a qual o pacote que está chegando será repassado pelo elemento de comutação. A escolha da porta de saída é feita usando a informação contida na tabela de repasse. Embora a tabela de repasse seja calculada pelo processador de roteamento, uma cópia da tabela é comumente armazenada em cada porta de entrada e atualizada, quando necessário, pelo processador de roteamento. Com cópias locais da tabela de repasse, as decisões de repasse podem ser tomadas localmente, em cada porta de entrada, sem chamar o processador de roteamento centralizado. Esse repasse *descentralizado* evita a criação de um gargalo de processamento de repasse em um único ponto no interior do roteador.

História

Cisco Systems: dominando o núcleo da rede

Até março de 2004, a Cisco empregava mais de 30 mil pessoas e tinha uma capitalização de mercado de cerca de 150 bilhões de dólares. A empresa dominava o mercado de roteadores da Internet e, nos últimos anos, voltou-se para o mercado de telefonia por Internet, em que concorre de igual para igual com empresas fabricantes de equipamentos de telefonia como Lucent, Alcatel, Nortel e Siemens. Como surgiu essa gigantesca empresa de rede? Tudo começou em 1984 (há apenas 20 anos) na sala de estar de um apartamento no Vale do Silício. Len Bosak e sua esposa Sandy Lerner trabalhavam na universidade de Stanford quando tiveram a idéia de construir e vender roteadores de Internet a instituições acadêmicas e de pesquisa. Sandy Lerner teve a idéia do nome "Cisco" (uma abreviação de San Francisco) e também criou o logotipo da empresa, que é uma ponte. A sede da empresa era originalmente a sala de estar do casal que, no início, financiou o projeto com cartões de crédito e trabalhos extras. No final de 1986, a receita mensal da Cisco chegou a 250 mil dólares. No final de 1987, a Cisco conseguiu atrair capital de risco — dois milhões de dólares da Sequoia Capital em troca de um terço da empresa. Nos anos seguintes, a Cisco continuou a crescer e a conquistar cada vez mais participação de mercado. Ao mesmo tempo, o relacionamento entre Bosak/Lerner e a administração da empresa começou a ficar tenso. A Cisco abriu seu capital em 1990 e, nesse mesmo ano, Lerner e Bosak saíram da empresa.

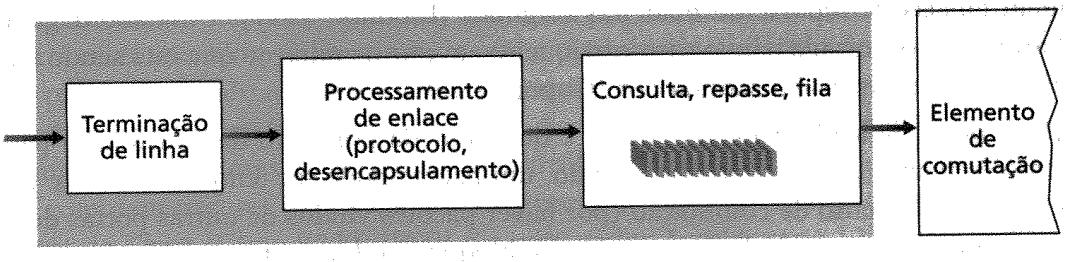


Figura 4.7 Processamento na porta de entrada

Em roteadores com capacidade limitada de processamento na porta de entrada, a porta pode simplesmente repassar o pacote para o processador de roteamento centralizado, que, então, realizará o exame da tabela de repasse e transmitirá o pacote à porta de saída apropriada. Esta é a abordagem adotada quando uma estação de trabalho ou um servidor funciona como um roteador. Aqui, o processador de roteamento é, na realidade, apenas a CPU da estação de trabalho, enquanto a porta de entrada é, na verdade, apenas uma placa de interface de rede (por exemplo, uma placa Ethernet).

Dada a existência de uma tabela de repasse, o exame é conceitualmente simples — basta procurar o registro mais longo compatível com o endereço de destino, como descrito na Seção 4.2.2. Porém, na prática, as coisas não são assim tão simples. Talvez o principal fator de complicação seja o fato de os roteadores de backbone terem de operar em altas velocidades, rodando milhões de exames por segundo. Na verdade, é desejável que o processamento da porta de entrada tenha a capacidade de operar à **velocidade da linha**, isto é, que o exame possa ser feito em tempo menor do que o necessário para receber um pacote na porta de entrada. Nesse caso, o processamento de entrada de um pacote recebido pode ser concluído antes do término da operação de recebimento subseqüente. Para ter uma idéia das exigências de desempenho para uma consulta, considere que um enlace do tipo OC48 opere à velocidade de 2,5 Gbps. Com pacotes de comprimento de 256 bytes, isso implica uma taxa de aproximadamente um milhão de consultas por segundo.

Devido à necessidade de operar nas altas velocidades de enlace atuais, é impossível uma busca linear em uma tabela de repasse extensa. Uma técnica mais razoável é armazenar os registros da tabela de repasse em uma estrutura de árvore de dados. Cada nível da árvore pode ser imaginado como um bit no endereço de destino. Para examinar um endereço, basta simplesmente começar no nó da raiz da árvore. Se o primeiro bit do endereço for zero, então a subárvore da esquerda conterá o registro da tabela de repasse para o endereço de destino; caso contrário, ele estará na subárvore da direita. A subárvore apropriada é, então, percorrida usando os bits de endereço remanescentes — se o próximo bit do endereço for zero, será escolhida a subárvore da esquerda da subárvore inicial; caso contrário, será escolhida a subárvore da direita. Dessa maneira, podemos examinar o registro da tabela de repasse em N etapas, em que N é o número de bits do endereço. (Note que isso é, essencialmente, uma busca binária em um espaço de endereço de tamanho 2^N.) Um melhoramento das técnicas de busca binária é descrito por [Srinivasan, 1999], e um levantamento geral de algoritmos de classificação de pacotes pode ser encontrado em [Gupta, 2001].

Mas, mesmo quando $N = 32$ etapas (por exemplo, um endereço IP de 32 bits), a velocidade de consulta pela busca binária não é suficientemente rápida para os requisitos atuais de roteamento de backbone. Por exemplo, admitindo um acesso à memória a cada etapa, menos de um milhão de consultas de endereços por segundo podem ser realizadas com tempos de acesso à memória de 40 nanossegundos. Assim, diversas técnicas têm sido pesquisadas para aumentar as velocidades de consulta. **Memórias de conteúdo endereçável** (*content addressable memories* — CAMs) permitem que um endereço IP de 32 bits seja apresentado à CAM, que então devolve o conteúdo do registro da tabela de repasse para o endereço em tempo essencialmente constante. O roteador da série 8500 da Cisco [Cisco 8500, 1999] tem uma CAM de 64K para cada porta de entrada. Outra técnica para acelerar a consulta é manter registros de tabela de repasse acessados recentemente em armazenamento intermediário (cache) [Feldmeier, 1988]. Nesse caso, a preocupação é o tamanho potencial do cache. Mais recentemente, foram propostas estruturas de dados ainda mais velozes, que permitem que os registros da tabela de repasse sejam localizados em $\log(N)$ etapas [Waldvogel, 1997] ou

que comprimem tabelas de repasse de maneiras originais [Brodnik, 1997]. Uma abordagem da consulta baseada em hardware e otimizada para o caso corriqueiro em que o endereço que está sendo examinado tem 24 bits significativos ou menos é discutida por [Gupta, 1998].

Assim que a porta de saída para um pacote é determinada por meio da consulta, o pacote pode ser repassado para o elemento de comutação. Contudo, um pacote pode ser temporariamente impedido (bloqueado) de entrar no elemento de comutação (por haver pacotes vindos de outras portas de entrada usando o elemento naquele momento). Assim, um pacote bloqueado deve entrar na fila da porta de entrada e então ser programado para atravessar o elemento de comutação mais tarde. Vamos examinar com mais detalhes o bloqueio, a formação de fila e o escalonamento de pacotes (nas portas de entrada e de saída) dentro de um roteador na Seção 4.3.4.

4.3.2 Elemento de comutação

O elemento de comutação está no coração de um roteador. É por meio do elemento de comutação que os pacotes são comutados (isto é, repassados) de uma porta de entrada para uma porta de saída. A comutação pode ser realizada de inúmeras maneiras, como mostra a Figura 4.8.

Comutação por memória. Os primeiros e mais simples roteadores quase sempre eram computadores tradicionais nos quais a comutação entre as portas de entrada e de saída era realizada sob o controle direto da CPU (processador de roteamento). As portas de entrada e de saída funcionavam como dispositivos tradicionais de entrada/saída de um sistema operacional tradicional. Uma porta de entrada na qual um pacote estivesse entrando primeiramente sinalizaria ao processador de roteamento por meio de uma interrupção. O pacote era então copiado da porta de entrada para a memória do processador. O processador de roteamento então extraía o endereço de destino do cabeçalho, consultava a porta de saída apropriada na tabela de repasse e copiava o pacote para os buffers da porta de saída. Note que, se a largura de banda da memória for tal que B pacotes/segundo possam ser escritos ou lidos na memória, então a vazão total de repasse (a velocidade total com que os pacotes são transferidos de portas de entrada para portas de saída) deverá ser menor do que $B/2$.

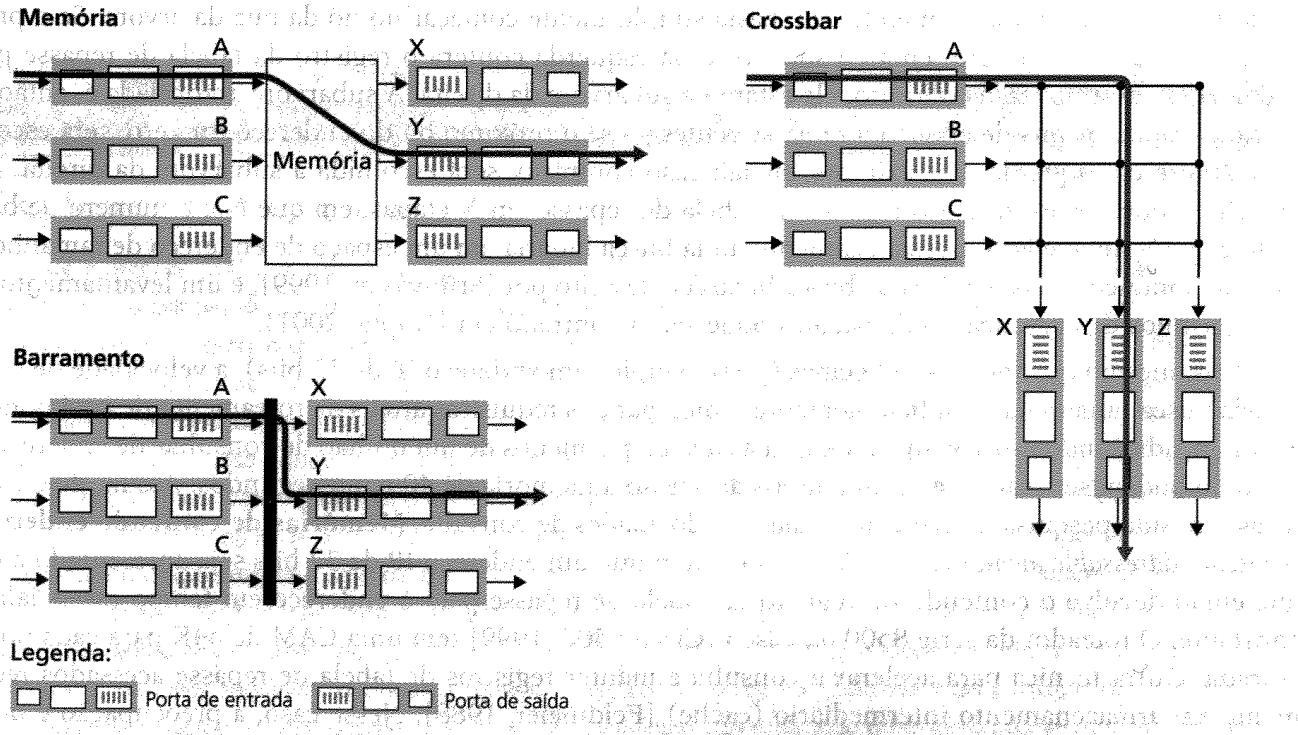


Figura 4.8 Três técnicas de comutação

Muitos roteadores modernos também comutam por memória. Contudo, uma importante diferença entre esses roteadores e os antigos é que a consulta do endereço de destino e o armazenamento do pacote na localização adequada da memória são realizados por processadores nas placas de linha de entrada. Em certos aspectos, roteadores que comutam por memória se parecem muito com multiprocessadores de memória compartilhada, nos quais os processadores de uma placa de linha comutam pacotes para a memória da porta de saída adequada. Os comutadores série 8500 Catalyst da Cisco [Cisco 8500, 1999] e o comutador Accelar série 1200 da Bay Networks comutam pacotes por uma memória compartilhada. Um modelo abstrato para estudar as propriedades da comutação por memória e uma comparação com outras formas de comutação podem ser encontrados em [Iyer, 2002].

Comutação por um barramento. Nessa abordagem, as portas de entrada transferem um pacote diretamente para a porta de saída por um barramento compartilhado sem a intervenção do processador de roteamento (note que na comutação por memória o pacote também deve atravessar o barramento do sistema para vir da memória e ir para ela.) Embora o processador de roteamento não esteja envolvido na transferência por barramento, como o barramento é compartilhado, somente um pacote por vez pode ser transferido por meio do barramento. Um pacote que chega a uma porta de entrada e encontra o barramento ocupado com a transferência de outro pacote tem bloqueada sua passagem pelo elemento de comutação e é colocado na fila da porta de entrada. Como cada pacote deve atravessar o único barramento, a largura de banda de comutação do roteador fica limitada à velocidade do barramento.

Dado que larguras de banda de barramento acima de 1 Gbps são possíveis com a tecnologia atual, a comutação por barramento muitas vezes é suficiente para roteadores que operam em redes de acesso e redes de empresas (por exemplo, redes locais [LANs] e redes corporativas). A comutação por barramento foi adotada em uma série de produtos de roteamento atuais, entre eles o Cisco 1900 [Cisco Switches, 1999], que comuta pacotes por um barramento de troca de pacotes (*packets exchange bus*) de 1 Gbps. O sistema CoreBuilder 5000 da 3Com [Kapoor, 1997] interconecta portas que residem em módulos de comutação diferentes por seu barramento de dados PacketChannel com largura de banda de 2 Gbps.

Comutação por uma rede de interconexão. Um modo de vencer a limitação da largura de banda de um barramento único compartilhado é usar uma rede de interconexão mais sofisticada, tal como as que eram utilizadas no passado para interconectar processadores em uma arquitetura de computadores multiprocessadores. Um comutador do tipo crossbar é uma rede de interconexão que consiste em $2n$ barramentos. Esses barramentos conectam n portas de entrada com n portas de saída, como mostra a Figura 4.8. Um pacote que chega a uma porta de entrada percorre o barramento horizontal ligado à porta de entrada até interceptar o barramento vertical que leva à porta de saída desejada. Se o barramento vertical que leva à porta de saída estiver livre, o pacote será transferido para a porta de saída. Se o barramento vertical estiver sendo usado para transferir um pacote de outra porta de entrada para essa mesma porta de saída, o pacote que chegou ficará bloqueado e deverá entrar na fila da porta de entrada.

Elementos de comutação Delta e Omega também foram propostos como rede de interconexão entre portas de entrada e de saída. Consulte [Tobagi, 1990] para um levantamento de arquiteturas de comutação. Os comutadores da Família 12000 da Cisco [Cisco 12000, 1998] usam uma rede de interconexão que fornece até 60 Gbps pelo elemento de comutação. Uma tendência atual no projeto de rede de interconexão [Keshav, 1998] é fragmentar um datagrama IP de comprimento variável em células de comprimento fixo e, então, marcar e comutar as células de comprimento fixo por meio da rede de interconexão. As células são então montadas novamente reproduzindo o pacote original na porta de saída. A célula de comprimento fixo e a *tag* de controle interno podem simplificar e acelerar consideravelmente a comutação de pacotes pela rede de interconexão.

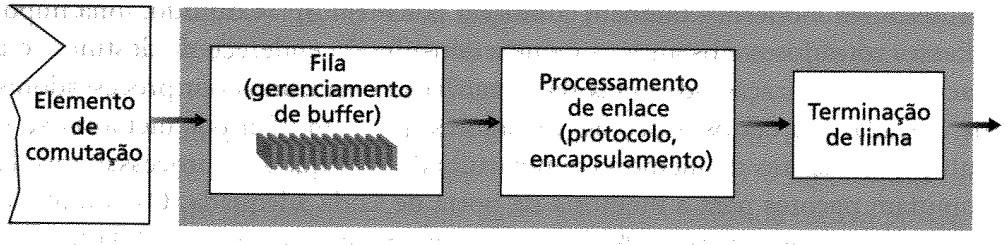


Figura 4.9 Processamento de porta de saída

4.3.3 Portas de saída

O processamento de portas de saída mostrado na Figura 4.9 toma os pacotes que foram armazenados na memória da porta de saída e os transmite pelo enlace de saída. O processamento do protocolo de enlace e a terminação da linha são as funcionalidades de camada de enlace e de camada física do lado remetente que interagem com a porta de entrada do outro lado do enlace de saída, como discutido na Seção 4.3.1. As funcionalidades de fila e de gerenciamento de buffer são necessárias quando o elemento de comutação entrega pacotes à porta de saída a uma taxa que excede a taxa do enlace de saída. Vamos examinar a seguir o enfileiramento na porta de saída.

4.3.4 Onde ocorre formação de fila?

Se examinarmos a funcionalidade da porta de entrada e da porta de saída e as configurações mostradas na Figura 4.8, veremos que filas de pacotes podem se formar tanto nas portas de entrada como nas portas de saída. É importante considerar essas filas com um pouco mais de detalhes, já que, à medida que elas ficam maiores, o espaço de buffer do roteador será eventualmente exaurido e ocorrerá **perda de pacote**. Lembre-se de que, em nossas discussões anteriores, dissemos que pacotes eram perdidos dentro da rede ou descartados em um roteador. E é ali, nessas filas dentro de um roteador, que esses pacotes são descartados ou perdidos. O local real da perda do pacote (seja nas filas da porta de entrada, seja nas filas da porta de saída) dependerá da carga de tráfego, da velocidade relativa do elemento de comutação e da taxa da linha, como discutiremos a seguir.

Suponha que as taxas da linha de entrada e as taxas da linha de saída sejam idênticas e que haja n portas de entrada e n portas de saída. Defina a **taxa do elemento de comutação** como a taxa na qual o elemento de comutação pode movimentar pacotes de portas de entrada à portas de saída. Se essa taxa for no mínimo n vezes a taxa da linha de entrada, então não ocorrerá formação de fila nas portas de entrada. Isso porque, mesmo no pior caso em que todas as n linhas de entrada estiverem recebendo pacotes, o comutador poderá transferir n pacotes da porta de entrada para a porta de saída no tempo que levar para cada uma das n portas de entrada receber (simultaneamente) um *único* pacote. Mas o que pode acontecer nas portas de saída? Vamos ainda supor que a taxa do elemento de comutação seja no mínimo n vezes as taxas das linhas. No pior caso, os pacotes que chegarem a cada uma das n portas de entrada serão destinados à *mesma* porta de saída. Nesse caso, no tempo que leva para receber (ou enviar) um único pacote, n pacotes chegarão a essa porta de saída. Uma vez que a porta de saída pode transmitir somente um único pacote em cada unidade de tempo (o tempo de transmissão do pacote), os n pacotes que chegarem terão de entrar na fila (esperar) para transmissão pelo enlace de saída. Então, mais n pacotes poderão chegar durante o tempo que leva para transmitir apenas um dos n pacotes que estavam na fila anteriormente. E assim por diante. Finalmente, o número de pacotes na fila pode ficar grande o bastante para exaurir o espaço de memória na porta de saída, caso em que os pacotes são descartados.

A formação de fila na porta de saída está ilustrada na Figura 4.10. No tempo t, um pacote chegou a cada uma das portas de entrada, cada um deles destinado à porta de saída que está mais acima na figura. Admitindo taxas da linha idênticas e um comutador operando a uma taxa três vezes maior do que a da linha, uma unidade de tempo mais tarde (isto é, no tempo necessário para receber ou enviar um pacote), todos os

três pacotes originais foram transferidos para a porta de saída e estão em fila aguardando transmissão. Na unidade de tempo seguinte, um desses três pacotes terá sido transmitido pelo enlace de saída. Em nosso exemplo, dois *novos* pacotes chegaram do lado de entrada do comutador; um desses pacotes é destinado àquela mesma porta de saída que está mais acima na figura.

Uma conseqüência da fila na porta de saída é que um **escalonador de pacotes** na porta de saída deve escolher para transmissão um pacote dentre os que estão na fila. Essa seleção pode ser feita com base em uma regra simples baseada na própria ordem da fila (FCFS) ou por uma regra de escalonamento mais sofisticada, tal como a fila ponderada justa (*weighted fair queuing* — WFQ), que compartilha o enlace de saída com justiça entre as diferentes conexões fim-a-fim que têm pacotes na fila para transmissão. O escalonamento de pacotes desempenha um papel crucial no fornecimento de **garantia de qualidade de serviço**. Examinaremos o escalonamento de pacotes extensivamente no Capítulo 7. Uma discussão sobre disciplinas de escalonamento de pacotes na porta de saída pode ser encontrada em [Cisco Queue, 1995].

De modo semelhante, se não houver memória suficiente para armazenar um pacote que está chegando, será preciso tomar a decisão de descartar esse pacote (política conhecida como **descarte do final da fila**) ou remover um ou mais pacotes já enfileirados para liberar lugar para o pacote recém-chegado. Em alguns casos pode ser vantajoso descartar um pacote (ou marcar o seu cabeçalho) *antes* de o buffer ficar cheio, para dar um sinal de congestionamento ao remetente. Várias políticas de descarte e marcação de pacotes (conhecidas coletivamente como algoritmos de **gerenciamento ativo de fila** [*active queue management* — AQM]) foram propostas e analisadas [Labrador, 1999; Hollot, 2002]. Um dos algoritmos AQM mais estudados e implementados é o algoritmo de **detecção aleatória rápida** (*random early detection* — RED). Esse algoritmo mantém uma média ponderada do comprimento da fila de saída. Se o comprimento médio da fila for menor do que um valor limite mínimo, min_{th}, quando um pacote chegar, será admitido na fila. Inversamente, se a fila estiver cheia ou se o comprimento médio da fila for maior do que um valor limite máximo, max_{th}, quando um pacote chegar, será marcado ou descartado. Finalmente, se o pacote chegar e encontrar uma fila de comprimento médio no intervalo $[min_{th}, max_{th}]$, o pacote será marcado ou descartado com uma probabilidade que normalmente é alguma função do comprimento médio da fila, de min_{th} e de max_{th}. Foram propostas inúmeras funções probabilísticas para atuação nos pacotes e várias versões do RED foram modeladas, simuladas e/ou implementadas analiticamente. [Christiansen, 2001] e [Floyd, 2004] oferecem visões gerais e indicações de leituras adicionais.

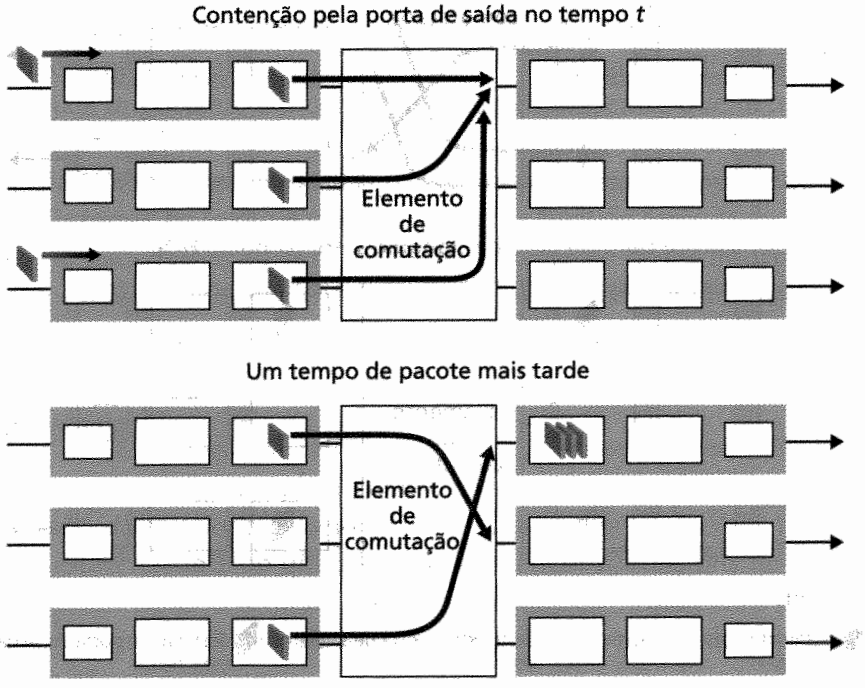


Figura 4.10 Formação de fila na porta de saída

Se o elemento de comutação não for suficientemente veloz (em relação às taxas da linha de entrada) para transmitir sem atraso *todos* os pacotes que chegam através dele, então poderá haver formação de fila também nas portas de entrada, pois os pacotes devem se juntar às filas nas portas de entrada para esperar sua vez de ser transferidos através do elemento de comutação até a porta de saída. Para ilustrar uma importante conseqüência dessa fila, considere um elemento de comutação do tipo 'crossbar' e suponha que (1) todas as velocidades de enlace sejam idênticas, (2) um pacote possa ser transferido de qualquer uma das portas de entrada até uma dada porta de saída no mesmo tempo que leva para um pacote ser recebido em um enlace de entrada e (3) pacotes sejam movimentados de uma dada fila de entrada até sua fila de saída desejada no modo FCFS. Vários pacotes podem ser transferidos em paralelo, contanto que suas portas de saída sejam diferentes. Entretanto, se dois pacotes que estão à frente das duas filas de entrada forem destinados à mesma fila de saída, então um dos pacotes ficará bloqueado e terá de esperar na fila de entrada — o elemento comutador só pode transferir um pacote por vez até uma porta de saída.

A parte superior da Figura 4.11 apresenta um exemplo em que dois pacotes (mais escuros) à frente de suas filas de entrada são destinados à mesma porta de saída mais alta à direita. Suponha que o elemento de comutação escolha transferir o pacote que está à frente da fila mais alta à esquerda. Nesse caso, o pacote escuro na fila mais baixa à esquerda tem de esperar. Mas não é apenas este último que tem de esperar; também tem de esperar o pacote claro que está na fila atrás dele (no retângulo inferior à esquerda), mesmo que não haja *nenhuma* contenção pela porta de saída do meio à direita (que é o destino do pacote claro). Esse fenômeno é conhecido como **bloqueio de cabeça de fila** (*head-of-the-line* — HOL *blocking*) em um comutador com fila de entrada — um pacote que está na fila de entrada deve esperar pela transferência através do elemento de comutação (mesmo que sua porta de saída esteja livre) porque ele está bloqueado por um outro pacote na cabeça da fila. [Karol, 1987] demonstra que, devido ao bloqueio HOL, o comprimento da fila de entrada cresce sem limites (informalmente, isso equivale a dizer que haverá significativas perdas de pacotes) em determinadas circunstâncias assim que a taxa de chegada de pacotes no enlace de entrada alcançar apenas 58 por cento de sua capacidade. Uma série de soluções para o bloqueio HOL é discutida por [McKeown, 1997b].

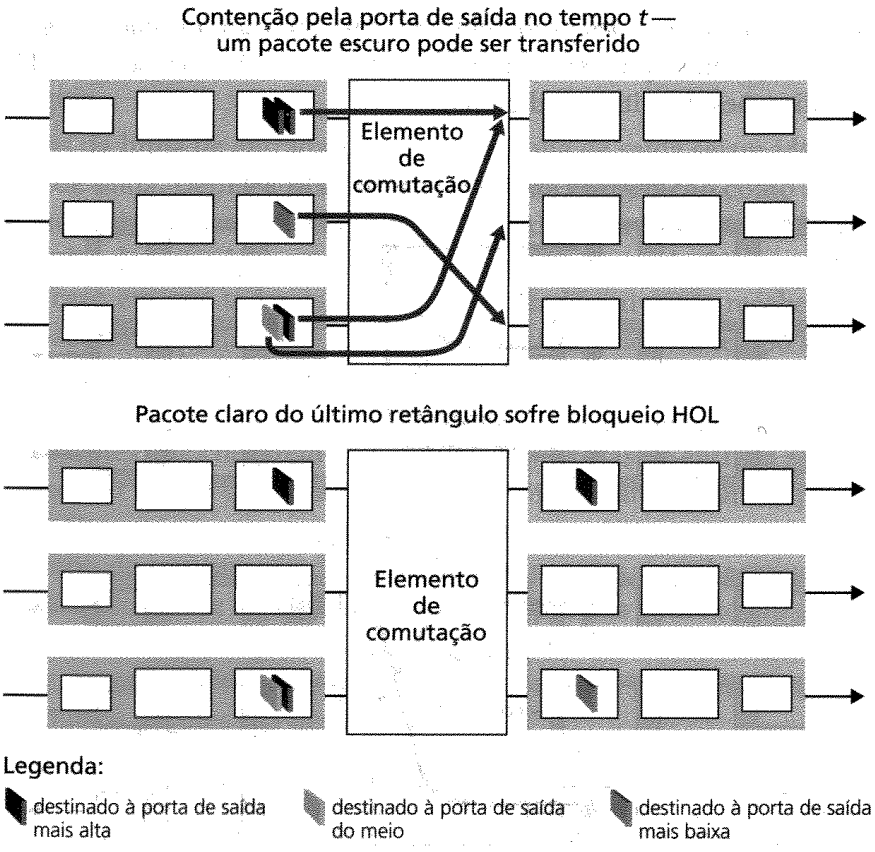


Figura 4.11 Bloqueio de cabeça de fila em um comutador com fila de entrada

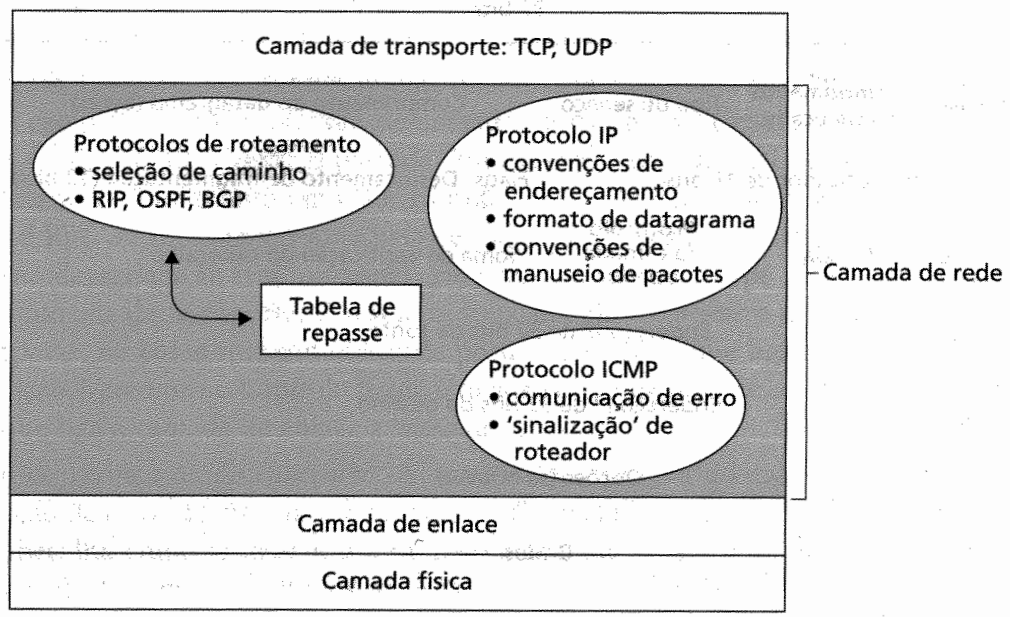


Figura 4.12 Contemplando o interior da camada de rede da Internet

4.4 O Protocolo da Internet (IP): repasse e endereçamento na Internet

Até agora discutimos endereçamento e repasse na camada de rede, sem referências a nenhuma rede de computadores específica. Nesta seção, voltaremos nossa atenção a como são feitos o endereçamento e o repasse na Internet. Veremos que o endereçamento e o repasse na Internet são componentes importantes do Protocolo da Internet (IP). Há duas versões do protocolo IP em uso hoje. Examinaremos primeiramente a versão mais utilizada do IP, a versão 4, que normalmente é denominada simplesmente IPv4 [RFC 791]. Examinaremos a versão 6 do IP [RFC 2373; RFC 2460], que foi proposta para substituir o IPv4, no final desta seção.

Mas, antes de iniciar nosso ataque ao IP, vamos voltar um pouco atrás e considerar os componentes que formam a camada de rede da Internet. Como mostra a Figura 4.12, a camada de rede da Internet tem três componentes mais importantes. O primeiro componente é o protocolo IP, que é o tópico desta seção. O segundo componente mais importante é o componente de roteamento, que determina o caminho que um datagrama segue desde a origem até o destino. Mencionamos anteriormente que protocolos de roteamento calculam as tabelas de repasse que são usadas para transmitir pacotes pela rede. Estudaremos os protocolos de roteamento da Internet na Seção 4.6. O componente final da camada de rede é um dispositivo para comunicação de erros em datagramas e para atender requisições de certas informações de camada de rede. Examinaremos o protocolo de comunicação de erro e de informações da Internet, ICMP (*Internet Control Message Protocol*), na Seção 4.4.3.

4.4.1 Formato do datagrama

Lembre-se de que um pacote de camada de rede é denominado um *datagrama*. Iniciamos nosso estudo do IP com uma visão geral da sintaxe e da semântica do datagrama IPv4. Você talvez esteja pensando que nada poderia ser mais desinteressante do que a sintaxe e a semântica dos bits de um pacote. Mesmo assim, o datagrama desempenha um papel central na Internet — todos os estudantes e profissionais de rede precisam vê-lo, absorvê-lo e dominá-lo. O formato do datagrama IPv4 é mostrado na Figura 4.13. Os principais campos desse datagrama são os seguintes:

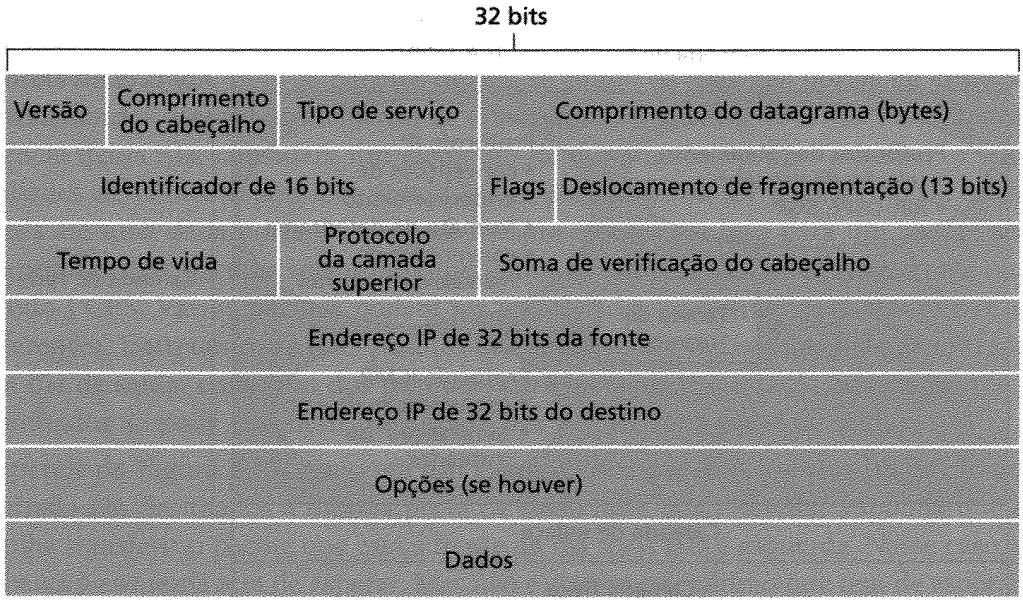


Figura 4.13 Formato do datagrama IPv4

- *Número da versão.* Esses quatro bits especificam a versão do protocolo IP do datagrama. Examinando o número da versão, o roteador pode determinar como interpretar o restante do datagrama IP. Diferentes versões de IP usam diferentes formatos de datagramas. O formato do datagrama para a versão atual do IP (IPv4) é mostrado na Figura 4.13. O formato do datagrama para a nova versão do IP (IPv6) será discutido no final desta seção.

- *Comprimento do cabeçalho.* Como um datagrama IPv4 pode conter um número variável de opções (incluídas no cabeçalho do datagrama IPv4), esses quatro bits são necessários para determinar onde, no datagrama IP, os dados realmente começam. A maior parte dos datagramas IP não contém opções; portanto, o datagrama IP típico tem um cabeçalho de 20 bytes.

- *Tipo de serviço.* Os bits de tipo de serviço (*type of service* — TOS) foram incluídos no cabeçalho do IPv4 para poder diferenciar os diferentes tipos de datagramas IP (por exemplo, datagramas que requerem, particularmente, baixo atraso, alta vazão ou confiabilidade) que devem ser distinguidos uns dos outros. Por exemplo, poderia ser útil distinguir datagramas de tempo real (como os usados por uma aplicação de telefonia IP) de tráfego que não é de tempo real (por exemplo, FTP). Um importante fabricante de roteadores (Cisco) interpreta os primeiros três bits de TOS como definição de níveis diferenciais de serviço que podem ser oferecidos pelo roteador. O nível específico de serviço a ser fornecido é uma questão de política determinada pelo administrador do roteador. Examinaremos detalhadamente a questão do serviço diferenciado no Capítulo 7.

- *Comprimento do datagrama.* É o comprimento total do datagrama IP (cabeçalho mais dados) medido em bytes. Uma vez que esse campo tem 16 bits de comprimento, o tamanho máximo teórico do datagrama IP é 65.535 bytes. Contudo, datagramas raramente são maiores do que 1.500 bytes.

- *Identificador, flags, deslocamento de fragmentação.* Esses três campos têm a ver com a fragmentação do IP, um tópico que em breve vamos considerar detalhadamente. O interessante é que a nova versão do IP, o IPv6, não permite fragmentação em roteadores.*

- *Tempo de vida.* O campo de tempo de vida (*time-to-live* — TTL) é incluído para garantir que datagramas não fiquem circulando para sempre na rede (devido a, por exemplo, um laço de roteamen-

* A fragmentação em IPv6 exige o uso de um cabeçalho adicional no datagrama que pode ser usado pela fonte do datagrama. (N. do R. T.)

to de longa duração). Esse campo é decrementado de uma unidade cada vez que o datagrama é processado por um roteador. Se o campo TTL chegar a 0, o datagrama deve ser descartado.

Protocolo. Esse campo é usado somente quando um datagrama IP chega a seu destino final. O valor desse campo indica o protocolo de camada de transporte específico ao qual a porção de dados desse datagrama IP deverá ser passada. Por exemplo, um valor 6 indica que a porção de dados será passada ao TCP, enquanto um valor 17 indica que os dados serão passados ao UDP. Consulte [RFC 1700 e RFC 3232] para ver uma lista de todos os valores possíveis. Note que o número do protocolo no datagrama IP tem um papel que é análogo ao do campo de número de porta no segmento da camada de transporte. O número do protocolo é o adesivo que liga as camadas de rede e de transporte, ao passo que o número de porta é o adesivo que liga as camadas de transporte e de aplicação. Veremos no Capítulo 5 que o quadro de camada de enlace também tem um campo especial que liga a camada de enlace à camada de rede.

Soma de verificação do cabeçalho. A soma de verificação do cabeçalho auxilia um roteador na detecção de erros de bits em um datagrama IP recebido. A soma é calculada tratando cada 2 bytes do cabeçalho como se fossem um número e somando esses números usando complementos aritméticos de 1. Como discutimos na Seção 3.3, o complemento de 1 dessa soma, conhecida como soma de verificação da Internet, é armazenado no campo de soma de verificação. Um roteador calculará o valor da soma de verificação para cada datagrama IP recebido e detectará uma condição de erro se o valor da soma de verificação carregado no cabeçalho do datagrama não for igual à soma de verificação calculada. Roteadores normalmente descartam datagramas quando um erro é detectado. Note que a soma de verificação deve ser recalculada e armazenada novamente em cada roteador, pois o campo TTL e, possivelmente, também os campos de opções podem mudar. Uma discussão interessante sobre algoritmos rápidos para calcular a soma de verificação da Internet é encontrada em [RFC 1071]. Uma pergunta que sempre é feita nesse ponto é: por que o TCP/IP faz verificação de erro nas camadas de transporte e de rede? Há várias razões para essa repetição. Em primeiro lugar, note que, na camada IP, a soma de verificação é calculada somente para o cabeçalho IP, enquanto no TCP/UDP a soma de verificação é calculada para todo o segmento TCP/IP. Em segundo lugar, o TCP/UDP e o IP não precisam necessariamente pertencer à mesma pilha de protocolos. O TCP pode, em princípio, rodar sobre um protocolo diferente (por exemplo, ATM) e o IP pode carregar dados que não serão passados ao TCP/UDP.

Endereços IP de fonte e de destino. Quando uma fonte cria um datagrama, insere seu endereço IP no campo de endereço de fonte IP e insere o endereço do destino final no campo de endereço de destinatário IP. Muitas vezes o hospedeiro da fonte determina o endereço do destinatário por meio de uma consulta ao DNS, como discutimos no Capítulo 2. Discutiremos endereçamento IP detalhadamente na Seção 4.4.2.

Opções. O campo de opções permite que um cabeçalho IP seja ampliado. A intenção é que as opções de cabeçalho sejam usadas raramente — por isso a decisão de poupar sobrecarga não incluindo a informação em campos de opções em todos os cabeçalhos de datagrama. Contudo, a mera existência de opções, na realidade, complica as coisas — uma vez que cabeçalhos de datagramas podem ter comprimentos variáveis, não é possível determinar *a priori* onde começará o campo de dados. Além disso, uma vez que alguns datagramas podem requerer processamento de opções e outros não, a quantidade de tempo necessária para processar um datagrama IP em um roteador pode variar bastante. Essas considerações se tornam particularmente importantes para o processamento do IP em roteadores e hospedeiros de alto desempenho. Por essas e outras razões, as opções IP foram descartadas no cabeçalho da versão IPv6, como discutimos na Seção 4.4.4.

Dados (carga útil). Finalmente, chegamos ao último e mais importante campo — a razão de ser do datagrama! Em muitas circunstâncias, o campo de dados do datagrama IP contém o segmento da camada de transporte (TCP ou UDP) a ser entregue ao destino. Contudo, o campo de dados pode carregar outros tipos de dados, como mensagens ICMP (discutidas na Seção 4.4.3).

Note que um datagrama IP tem um total de 20 bytes de cabeçalho (admitindo-se que não haja opções). Se o datagrama carregar um segmento TCP, então cada datagrama (não fragmentado) carrega um total de 40 bytes de cabeçalho (20 bytes de cabeçalho IP, mais 20 bytes de cabeçalho TCP) juntamente com a mensagem de camada de aplicação.

Fragmentação do datagrama IP

Veremos no Capítulo 5 que nem todos os protocolos de camada de enlace podem transportar pacotes do mesmo tamanho. Alguns protocolos podem transportar datagramas grandes, ao passo que outros podem transportar apenas pacotes pequenos. Por exemplo, quadros Ethernet não podem conter mais do que 1.500 bytes de dados, enquanto quadros para alguns enlaces de longa distância não podem conter mais do que 576 bytes. A quantidade máxima de dados que um quadro de camada de enlace pode carregar é denominada unidade máxima de transmissão (*maximum transmission unit* — MTU). Como cada datagrama IP é encapsulado dentro do quadro de camada de enlace para ser transportado de um roteador até o roteador seguinte, a MTU do protocolo de camada de enlace estabelece um limite estrito para o comprimento de um datagrama IP. Ter uma limitação estrita para o tamanho de um datagrama IP não é grande problema. O problema é que cada um dos enlaces ao longo da rota entre remetente e destinatário pode usar diferentes protocolos de camada de enlace, e cada um desses protocolos pode ter diferentes MTUs.

Para entender melhor a questão do repasse, imagine que *você* é um roteador que está interligando diversos enlaces, cada um rodando diferentes protocolos de camada de enlace com diferentes MTUs. Suponha que você receba um datagrama IP de um enlace, verifique sua tabela de repasse para determinar o enlace de saída e perceba que esse enlace de saída tem uma MTU que é menor do que o comprimento do datagrama IP. É hora de entrar em pânico — como você vai comprimir esse datagrama IP de tamanho excessivo no campo de carga útil do quadro da camada de enlace? A solução para esse problema é fragmentar os dados do datagrama IP em dois ou mais datagramas IP menores e, então, enviar esses datagramas menores pelo enlace de saída. Cada um desses datagramas menores é denominado um **fragmento**.

Fragmentos precisam ser reconstruídos antes que cheguem à camada de transporte no destino. Na verdade, tanto o TCP quanto o UDP esperam receber da camada de rede segmentos completos, não fragmentados. Os projetistas do IPv4 perceberam que a reconstrução de datagramas nos roteadores introduziria uma complicação significativa no protocolo e colocaria um freio no desempenho do roteador. (Se você fosse um roteador, iria querer reconstruir fragmentos além de tudo mais que tem de fazer?) Seguindo o princípio de manter a simplicidade do núcleo da rede, os projetistas do IPv4 decidiram alocar a tarefa de reconstrução de datagramas aos sistemas finais, e não aos roteadores da rede.

Quando um hospedeiro destinatário recebe uma série de datagramas da mesma fonte, ele precisa determinar se alguns desses datagramas são fragmentos de um datagrama original de maior tamanho. Se alguns desses datagramas forem fragmentos, o hospedeiro ainda deverá determinar quando recebeu o último fragmento e como os fragmentos recebidos devem ser reconstruídos para voltar à forma do datagrama original. Para permitir que o hospedeiro destinatário realize essas tarefas de reconstrução, os projetistas do IP (versão 4) criaram campos de *identificação*, *flag* e *deslocamento* de fragmentação no datagrama IP. Quando um datagrama é criado, o hospedeiro remetente marca o datagrama com um número de identificação, bem como com os endereços da fonte e do destino. O hospedeiro remetente incrementa o número de identificação para cada datagrama que envia. Quando um roteador precisa fragmentar um datagrama, cada datagrama resultante (isto é, cada fragmento) é marcado com o endereço da fonte, o endereço do destino e o número de identificação do datagrama original. Quando o destinatário recebe uma série de datagramas do mesmo hospedeiro remetente, pode examinar os números de identificação dos datagramas para determinar quais deles são, na verdade, fragmentos de um mesmo datagrama de tamanho maior. Como o IP é um serviço não confiável, é possível que um ou mais desses fragmentos jamais cheguem ao destino. Por essa razão, para que o hospedeiro de destino fique absolutamente seguro de que recebeu o último fragmento do datagrama original, o último datagrama tem um bit de flag ajustado para 0, ao passo que todos os outros fragmentos têm um bit de flag ajustado para 1. Além disso, para que o hospedeiro destinatário possa determinar se está faltando algum fragmento (e possa reconstruir os fragmentos na ordem correta), o campo de deslocamento é usado para especificar a localização exata do fragmento no datagrama IP original.

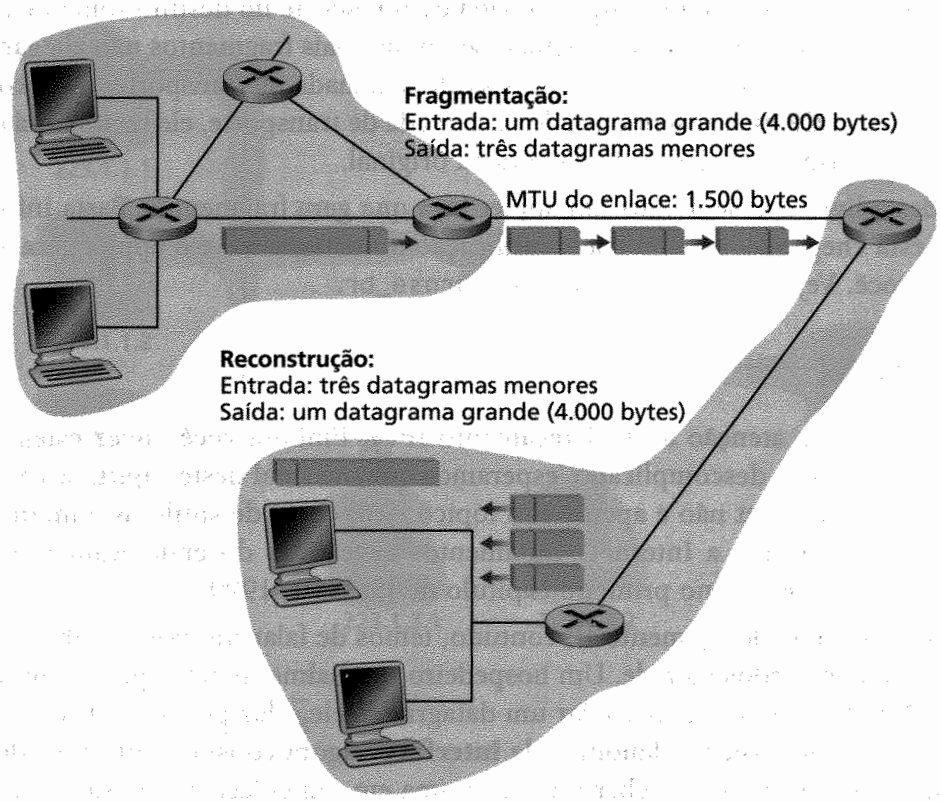


Figura 4.14 Fragmentação e reconstrução IP

A Figura 4.14 apresenta um exemplo. Um datagrama de 4 mil bytes (20 bytes de cabeçalho IP, mais 3.980 bytes de carga útil IP) chega a um roteador e deve ser repassado a um enlace com MTU de 1.500 bytes. Isso implica que os 3.980 bytes de dados do datagrama original devem ser alocados a três fragmentos separados (cada qual é também um datagrama IP). Suponha que o datagrama original esteja marcado com o número de identificação 777. As características dos três fragmentos são mostradas na Tabela 4.2. Os valores da Tabela 4.2 refletem a exigência de que a quantidade de dados da carga útil original em todos os fragmentos, exceto o último, seja um múltiplo de 8 bytes, e que o valor de deslocamento seja especificado em unidades de porções de 8 bytes.

Fragmento	Bytes	ID	Deslocamento	Flag
1º fragmento	1.480 bytes no campo de dados do datagrama IP	identificação = 777	deslocamento = 0 (o que significa que os dados devem ser inseridos a partir do byte 0)	flag = 1 (o que significa que há mais)
2º fragmento	1.480 bytes de dados	identificação = 777	deslocamento = 185 (o que significa que os dados devem ser inseridos a partir do byte 1.480. Note que 185 x 8 = 1.480)	flag = 1 (o que significa que há mais)
3º fragmento	1.020 bytes de dados (= 3.980–1.480–1.480)	identificação = 777	deslocamento = 370 (o que significa que os dados devem ser inseridos a partir do byte 2.960. Note que 370 x 8 = 2.960)	flag = 0 (o que significa que esse é o último fragmento)

Tabela 4.2 Fragmentos IP

A carga útil do datagrama é passada para a camada de transporte no destino somente após a camada IP ter reconstruído totalmente o datagrama IP original. Se um ou mais fragmentos não chegarem ao destino, o datagrama incompleto será descartado e não será passado à camada de transporte. Mas, como aprendemos no capítulo anterior, se o TCP estiver sendo usado na camada de transporte, ele recuperará essa perda fazendo com que a fonte retransmita os dados do datagrama original.

No site Web deste livro apresentamos um applet Java que gera fragmentos. Basta informar o tamanho do datagrama que está chegando, a MTU e a identificação do datagrama, e o applet gera automaticamente os fragmentos para você. Veja `http://www.aw.com/kurose_br`.

4.4.2 Endereçamento IPv4

Agora voltaremos nossa atenção ao endereçamento IPv4. Embora você talvez esteja pensando que o endereçamento seja um tópico descomplicado, esperamos que, no final deste capítulo, você se convença de que o endereçamento na Internet não é apenas um tópico rico, cheio de sutilezas e interessante, mas também de crucial importância para a Internet. Tratamentos excelentes do endereçamento IPv4 podem ser encontrados em [Semeria, 1996] e no primeiro capítulo de [Stewart, 1999].

Antes de discutirmos o endereçamento IP, contudo, temos de falar um pouco sobre como hospedeiros e roteadores estão interconectados na rede. Um hospedeiro normalmente tem apenas um único enlace com a rede. Quando o IP no hospedeiro quer enviar um datagrama, ele o faz por meio desse enlace. A fronteira entre o hospedeiro e o enlace físico é denominada **interface**. Agora considere um roteador e suas interfaces. Como a tarefa de um roteador é receber um datagrama em um enlace e repassá-lo a algum outro enlace, ele necessariamente estará ligado a dois ou mais enlaces. A fronteira entre o roteador e qualquer um desses enlaces também é denominada uma interface. Assim, um roteador tem múltiplas interfaces, uma para cada um de seus enlaces. Como todos os hospedeiros e roteadores podem enviar e receber datagramas IP, o IP exige que cada interface tenha seu próprio endereço IP. Desse modo, um endereço IP está tecnicamente associado com uma interface, e não com um hospedeiro ou um roteador que contém aquela interface.

Cada endereço IP tem comprimento de 32 bits (equivalente a 4 bytes). Portanto, há um total de 2^{32} endereços IP possíveis. Fazendo uma aproximação de 2^{10} por 10^3, é fácil ver que há cerca de 4 bilhões de endereços IP possíveis. Esses endereços são escritos em **notação decimal separada por pontos** (*dotted-decimal notation*), na qual cada byte do endereço é escrito em sua forma decimal e separado dos outros bytes do endereço por um ponto. Por exemplo, considere o endereço IP 193.32.216.9. O 193 é o número decimal equivalente aos primeiros 8 bits do endereço; o 32 é o decimal equivalente ao segundo conjunto de 8 bits do endereço e assim por diante. Por conseguinte, o endereço 193.32.216.9, em notação binária, é:

11000001 00100000 11011000 00001001

Cada interface em cada hospedeiro e roteador da Internet global tem de ter um endereço IP globalmente exclusivo (exceto as interfaces por trás de NATs, como discutiremos no final desta seção). Contudo, esses endereços não podem ser escolhidos de qualquer maneira. Uma parte do endereço IP de uma interface será determinada pela sub-rede à qual ela está conectada.

A Figura 4.15 fornece um exemplo de endereçamento IP e interfaces. Nessa figura, um roteador (com três interfaces) é usado para interconectar sete hospedeiros. Examine os endereços IP atribuídos às interfaces de hospedeiros e roteadores; há diversas particularidades a notar. Todos os três hospedeiros da parte superior esquerda da Figura 4.15 — e também a interface do roteador ao qual estão ligados — têm um endereço IP na forma 223.1.1.xxx, ou seja, todos eles têm os mesmos 24 bits mais à esquerda em seus endereços IP. As quatro interfaces também estão interconectadas por uma rede *que não contém nenhum roteador*. (Essa rede poderia ser, por exemplo, uma LAN Ethernet, caso em que as interfaces seriam conectadas por um hub Ethernet ou um comutador Ethernet; veja o Capítulo 5). Na terminologia IP, essa rede que interconecta três interfaces de hospedeiros e uma interface de roteador forma uma **sub-rede** [RFC 950]. (Na lite-

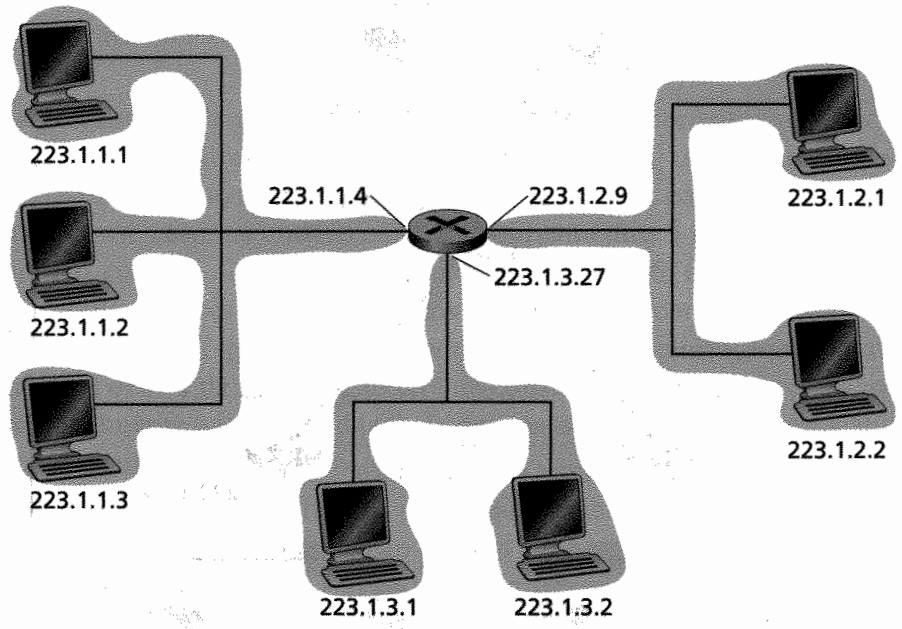


Figura 4.15 Endereços de interfaces e sub-redes

ratura da Internet, uma sub-rede também é denominada uma *rede IP* ou simplesmente uma *rede*). O endereçamento IP designa um endereço a essa sub-rede: 223.1.1.0/24, no qual a notação /24, às vezes conhecida como uma **máscara de rede**, indica que os 24 bits mais à esquerda do conjunto de 32 bits definem o endereço da sub-rede. Assim, a sub-rede 223.1.1.0/24 consiste em três interfaces de hospedeiros (223.1.1.1, 223.1.1.2 e 223.1.1.3) e uma interface de roteador (223.1.1.4). Quaisquer hospedeiros adicionais ligados à sub-rede 223.1.1.0/24 seriam obrigados a ter um endereço na forma 223.1.1.xxx. Há duas sub-redes adicionais mostradas na Figura 4.15: a sub-rede 223.1.2.0/24 e a sub-rede 223.1.3.0/24. A Figura 4.16 ilustra as três sub-redes IP presentes na Figura 4.15.

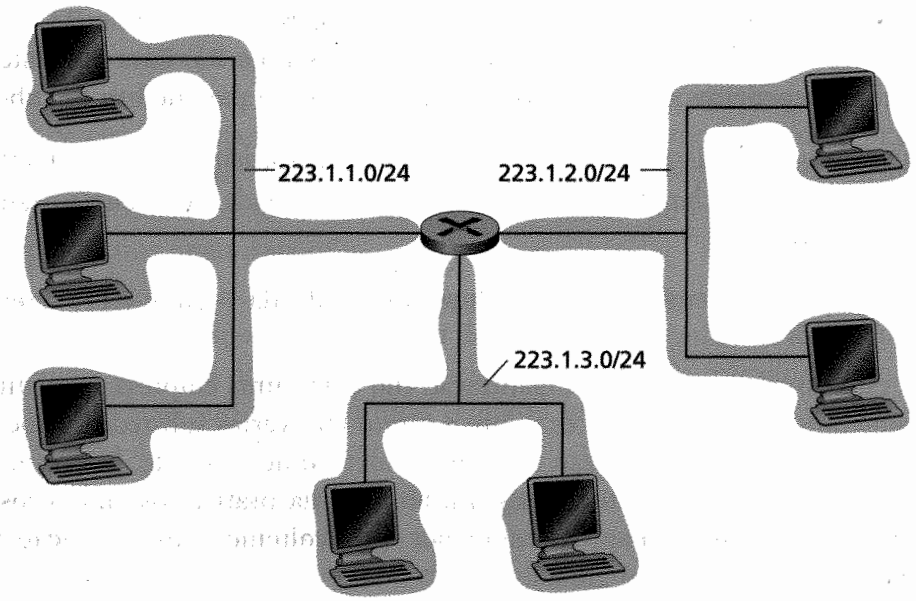


Figura 4.16 Endereços de sub-redes

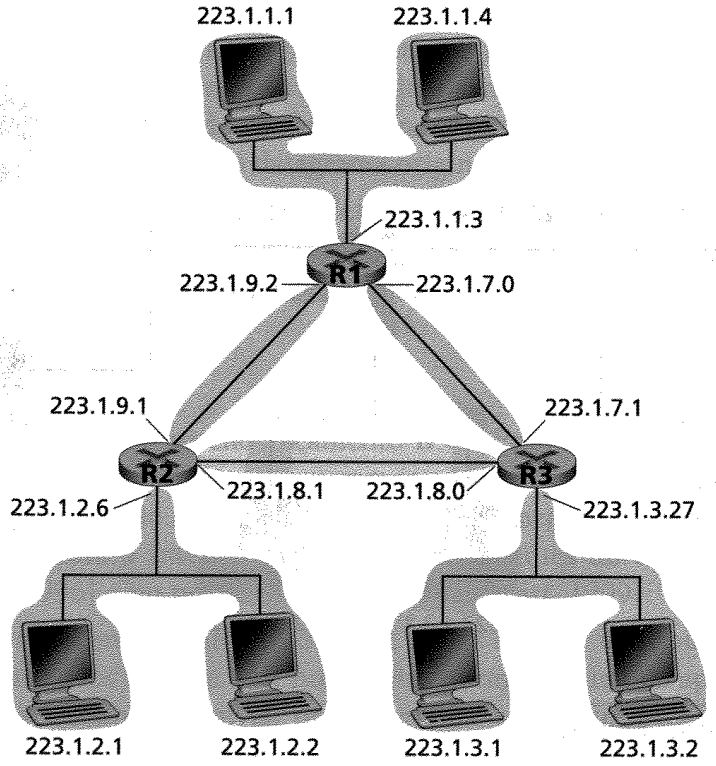


Figura 4.17 Três roteadores interconectando seis sub-redes

A definição IP de uma sub-rede não está restrita a segmentos Ethernet que conectam múltiplos hospedeiros a uma interface de roteador. Para entender melhor isso, considere a Figura 4.17, que mostra três roteadores interconectados por enlaces ponto a ponto. Cada roteador tem três interfaces, uma para cada enlace ponto a ponto e uma para o enlace *broadcast* que conecta diretamente o roteador a um par de hospedeiros. Que sub-redes IP estão presentes aqui? Três sub-redes, 223.1.1.0/24, 223.1.2.0/24 e 223.1.3.0/24, semelhantes às sub-redes que encontramos na Figura 4.15. Mas note que também há três sub-redes adicionais nesse exemplo: uma sub-rede, 223.1.9.0/24, para as interfaces que conectam os roteadores R1 e R2; outra sub-rede, 223.1.8.0/24, para as interfaces que conectam os roteadores R2 e R3, e uma terceira sub-rede, 223.1.7.0/24, para as interfaces que conectam os roteadores R3 e R1. Para um sistema geral interconectado de roteadores e hospedeiros, podemos usar a seguinte receita para definir as sub-redes no sistema.

> *Para determinar as sub-redes, destaque cada interface de seu hospedeiro ou roteador, criando ilhas de redes isoladas com interfaces fechando as terminações das redes isoladas. Cada uma dessas redes isoladas é denominada* ***sub-rede****.*

Se aplicarmos esse procedimento ao sistema interconectado da Figura 4.17, teremos seis ilhas ou sub-redes.

Fica claro, com essa discussão, que uma organização (tal como uma empresa ou instituição acadêmica) que tenha vários segmentos Ethernet e enlaces ponto a ponto terá várias sub-redes, e todos os equipamentos e dispositivos em uma dada sub-rede terão o mesmo endereço de sub-rede. Em princípio, as diferentes sub-redes poderiam ter endereços bem diferentes. Entretanto, na prática, os endereços de sub-rede freqüentemente têm muito em comum. Para entender por que, voltemos nossa atenção ao modo como o endereçamento é manipulado na Internet global.

A estratégia de atribuição de endereços da Internet é conhecida como **Roteamento Interdomínio sem Classes** (*Classless Interdomain Routing* — CIDR), que se pronuncia como a palavra *cider* (cidra) em inglês [RFC 1519]. O CIDR generaliza a noção de endereçamento de sub-rede. Como acontece com o endereça-

mento de sub-redes, o endereço IP de 32 bits é dividido em duas partes e, mais uma vez, tem a forma decimal com pontos de separação *a.b.c.d/x*, em que *x* indica o número de bits existentes na primeira parte do endereço.

Os *x* bits mais significativos de um endereço na forma *a.b.c.d/x* constituem a parcela da rede do endereço IP e normalmente são denominados **prefixo** (ou *prefixo de rede*). Uma organização normalmente recebe um bloco de endereços contíguos, isto é, uma faixa de endereços com um prefixo comum (ver quadro Princípios na prática). Nesse caso, os endereços IP de equipamentos e dispositivos dentro da organização compartilharão o prefixo comum. Quando estudarmos, na Seção 4.6, o protocolo de roteamento BGP da Internet, veremos que somente esses *x* bits indicativos do prefixo são considerados por roteadores que estão fora da rede da organização. Isto é, quando um roteador de fora da organização repassar um datagrama cujo endereço de destino esteja dentro da organização terá de considerar somente os *x* bits indicativos do endereço. Isso reduz consideravelmente o tamanho da tabela de repasse nesses roteadores, visto que um *único* registro da forma *a.b.c.d/x* será suficiente para transmitir pacotes para *qualquer* destino dentro da organização.

Os restantes $(32 - x)$ bits de um endereço podem ser considerados como os bits que distinguem os equipamentos e dispositivos *dentro* da organização e todos eles têm o mesmo prefixo de rede. Esses serão os bits considerados no repasse de pacotes em roteadores *dentro* da organização. Esses bits de ordem mais baixa podem (ou não) ter uma estrutura adicional de sub-rede tal como aquela discutida anteriormente. Por exemplo, suponha que os primeiros 21 bits do endereço *a.b.c.d/21*, por assim dizer, 'ciderizado', especificam o prefixo da rede da organização e são comuns aos endereços IP de todos os hospedeiros da organização. Os 11 bits restantes então identificam os hospedeiros específicos da organização. A estrutura interna da organização poderia ser tal que esses 11 bits mais à direita seriam usados para criar uma sub-rede dentro da organização, como discutido anteriormente. Por exemplo, *a.b.c.d/24* poderia se referir a uma sub-rede específica dentro da organização.

Antes da adoção do CIDR, os tamanhos das parcelas de um endereço IP estavam limitados a 8, 16 ou 24 bits, um esquema de endereçamento conhecido como **endereçamento de classes cheias**, já que sub-redes com endereços de sub-rede de 8, 16 e 24 eram conhecidas como redes de classe A, B e C, respectivamente. A exigência de que a parcela da sub-rede de um endereço IP tenha exatamente 1, 2 ou 3 bytes há muito tempo se mostrou problemática para suportar o rápido crescimento do número de organizações com sub-redes de pequeno e médio portes. Uma sub-rede de classe C (/24) poderia acomodar apenas $2^8 - 2 = 254$ hospedeiros (dois dos $2^8 = 256$ endereços são reservados para uso especial) — muito pequena para inúmeras organizações. Contudo, uma sub-rede de classe B (/16), que suporta até 65.634 hospedeiros, seria demasiadamente grande. Com o endereçamento de classes cheias, uma organização com, digamos, dois mil hospedeiros, recebia um endereço de sub-rede de classe B (/16), o que resultava no rápido esgotamento do espaço de endereços de classe B e na má utilização do espaço de endereço alocado. Por exemplo, uma organização que usasse um endereço de classe B para seus dois mil hospedeiros, recebia espaço de endereços suficiente para até 65.534 interfaces — o que deixava mais de 63 mil endereços sem uso e que não poderiam ser utilizados por outras organizações.

Seríamos omissos se não mencionássemos ainda um outro tipo de endereço IP, o endereço de *broadcast* 255.255.255.255. Quando um hospedeiro emite um datagrama com endereço de destino 255.255.255.255, a mensagem é entregue a todos os hospedeiros na mesma sub-rede. Os roteadores também têm a opção de repassar a mensagem para suas sub-redes vizinhas (embora usualmente não o façam.) Examinaremos um exemplo que mostra como o *broadcast* IP é usado quando discutirmos o protocolo DHCP no Capítulo 5.

Agora que já estudamos o endereçamento IP detalhadamente, precisamos saber, antes de qualquer coisa, como hospedeiros e sub-redes obtêm seus endereços. Vamos começar examinando como uma organização obtém um bloco de endereços para seus equipamentos e, então, veremos como um equipamento (tal como um hospedeiro) recebe um endereço do bloco de endereços da organização.

Obtenção de um bloco de endereços

Para obter um bloco de endereços IP para utilizar dentro da sub-rede de uma organização, um administrador de rede poderia, em primeiro lugar, contatar seu ISP, que forneceria endereços a partir de um bloco maior de endereços que já estão alocados ao ISP. Por exemplo, o próprio ISP pode ter recebido o bloco de endereços 200.23.16.0/20. O ISP, por sua vez, pode dividir seu bloco de endereços em oito blocos de endereços contíguos, do mesmo tamanho, e dar um desses blocos de endereços a cada uma de um conjunto de oito organizações suportadas por ele, como demonstrado a seguir. (Sublinhamos a parte da sub-rede desses endereços para melhor visualização.)

Bloco do ISP	200.23.16.0/20	11001000 00010111 00010000	00000000
Organização 0	200.23.16.0/23	11001000 00010111 00010000	00000000
Organização 1	200.23.18.0/23	11001000 00010111 00010010	00000000
Organização 2	200.23.20.0/23	11001000 00010111 00010100	00000000
. . .		. . .	
Organização 7	200.23.30.0/23	11001000 00010111 00011110	00000000

Princípios na prática

Esse exemplo de um ISP que conecta oito organizações com a Internet também ilustra de maneira elegante como endereços 'ciderizados' cuidadosamente alocados facilitam o roteamento. Suponha, como mostra a Figura 4.18, que o ISP (que chamaremos de ISP Fly-By-Night) anuncie ao mundo exterior que devem ser enviados a ele quaisquer datagramas cujos primeiros 20 bits de endereço sejam iguais a 200.23.16.0/20. O resto do mundo não precisa saber que dentro do bloco de endereços 200.23.16.0/20 existem, na verdade, oito outras organizações, cada qual com suas próprias sub-redes. Essa capacidade de usar um único prefixo de rede para anunciar várias redes é freqüentemente denominada **agregação de endereços** (e também **agregação de rotas** ou **resumo de rotas**).

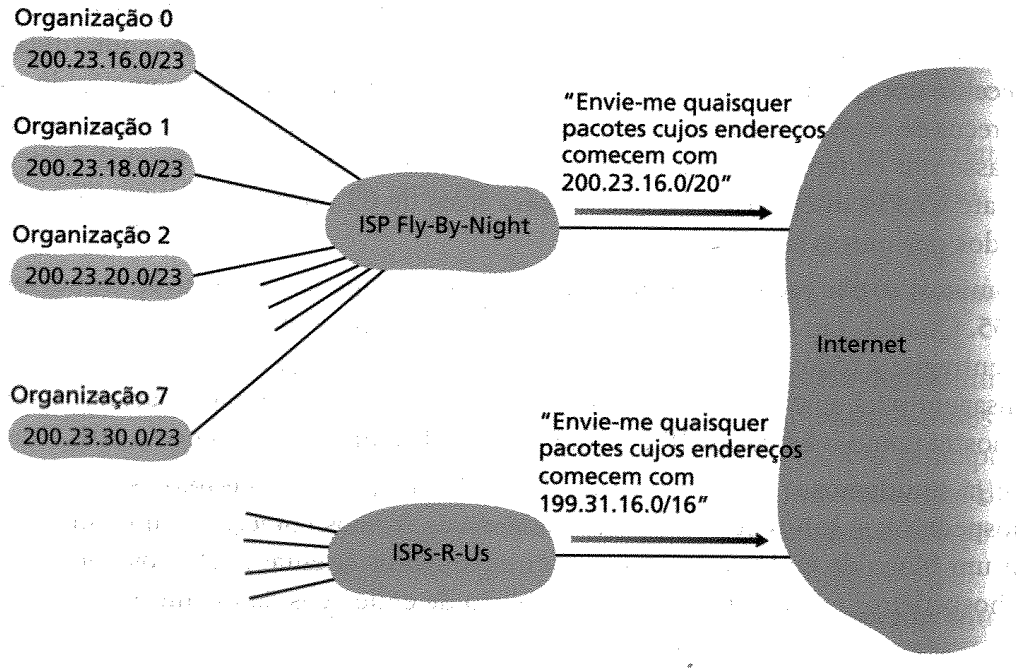


Figura 4.18 Endereçamento hierárquico e agregação de rotas

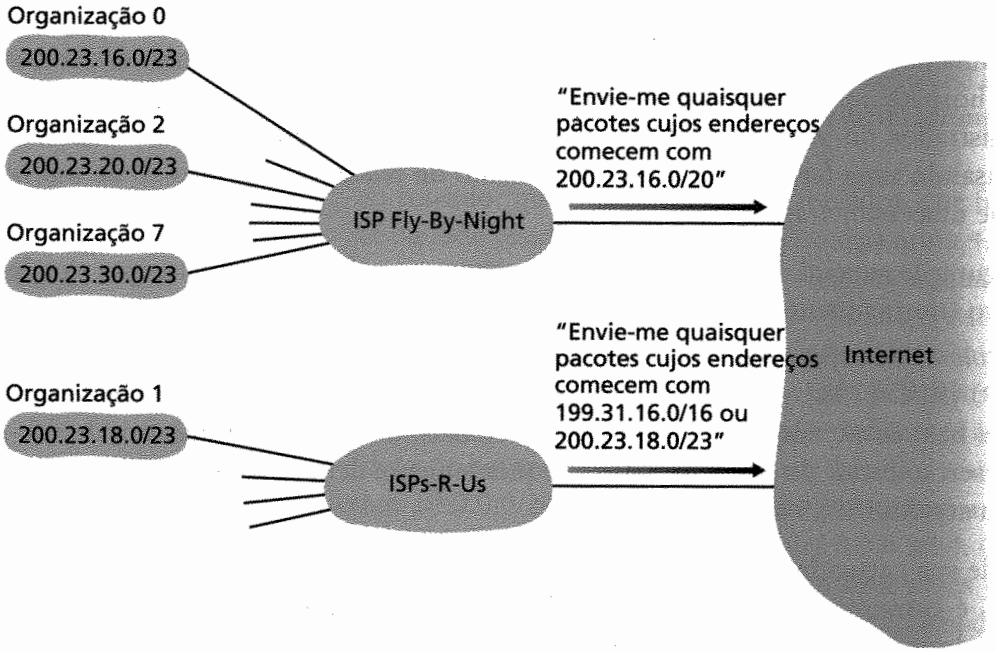


Figura 4.19 ISPs-R-Us tem uma rota mais específica para a Organização 1

A agregação de endereços funciona muito bem quando os endereços são alocados em blocos ao ISP e, então, pelo ISP às organizações clientes. Mas o que acontece quando os endereços não são alocados dessa maneira hierárquica? O que aconteceria, por exemplo, se o ISP Fly-By-Night adquirisse o ISPs-R-Us e então ligasse a Organização 1 com a Internet por meio de sua subsidiária ISPs-R-Us? Como mostra a Figura 4.18, a subsidiária ISPs-R-Us é dona do bloco de endereços 199.31.0.0/16, mas os endereços IP da Organização 1 infelizmente estão fora desse bloco de endereços. O que deveria ser feito nesse caso? Com certeza, a Organização 1 poderia renumerar todos os seus roteadores e hospedeiros para que seus endereços ficassem dentro do bloco de endereços do ISPs-R-Us. Mas essa é uma solução dispendiosa, e a Organização 1 poderia muito bem preferir mudar para uma outra subsidiária no futuro. A solução normalmente adotada é a Organização 1 manter seus endereços IP em 200.23.18.0/23. Nesse caso, como mostra a Figura 4.19, o ISP Fly-By-Night continua a anunciar o bloco de endereços 200.23.16.0/20 e o ISPs-R-Us continua a anunciar 199.31.0.0/16. Contudo, o ISPs-R-Us agora anuncia também o bloco de endereços para a Organização 1, 200.23.18.0/23. Quando outros roteadores da Internet virem os blocos de endereços 200.23.16.0/20 (do ISP Fly-By-Night) e 200.23.18.0/23 (do ISPs-R-Us) e quiserem rotear para um endereço no bloco 200.23.18.0/23, usarão a regra de compatibilização com o prefixo mais longo (veja Seção 4.2.2) e rotearão para o ISP-R-Us, já que ele anuncia o prefixo de endereço mais longo (mais específico) que combina com o endereço de destino.

Embora a obtenção de um conjunto de endereços de um ISP seja um modo de conseguir um bloco de endereços, não é o único. Evidentemente, também deve haver um modo de o próprio ISP obter um bloco de endereços. Há uma autoridade global que tenha a responsabilidade final de gerenciar o espaço de endereços IP e alocar blocos de endereços a ISPs e outras organizações? É claro que há! Endereços IP são administrados sob a autoridade da Internet Corporation for Assigned Names and Numbers (ICANN) [ICANN, 2004], com base em diretrizes estabelecidas no RFC 2050. O papel da ICANN, uma organização sem fins lucrativos [NTIA, 1998], não é somente alocar endereços IP, mas também administrar os servidores de nome raiz (DNS). Essa organização também tem a controvertida tarefa de atribuir nomes de domínio e resolver disputas de nomes de domínio. A ICANN aloca endereços a serviços regionais de registro da Internet (por exemplo, ARIN, RIPE, APNIC e LACNIC), que, juntas, formam a Address Supporting Organization da ICANN [ASO-ICANN, 2004], que é responsável pela alocação/administração de endereços dentro de suas regiões.

Obtenção de um endereço de hospedeiro

Tão logo tenha obtido um bloco de endereços, uma organização pode atribuir endereços IP individuais às interfaces de hospedeiros e roteadores em sua organização. No caso dos endereços de interface de roteadores, o administrador do sistema configura manualmente os endereços IP no roteador (quase sempre remotamente, usando uma ferramenta de gerenciamento de rede). O endereço IP de um hospedeiro pode ser designado de duas maneiras:

- *Configuração manual.* Um administrador de sistema configura manualmente o endereço IP no hospedeiro (normalmente em um arquivo).
- *Protocolo de Configuração Dinâmica de Hospedeiros* (*Dynamic Host Configuration Protocol* — DHCP) [RFC 2131]. O DHCP permite que um hospedeiro obtenha um endereço IP automaticamente, bem como adquira informações adicionais, tais como sua máscara de sub-rede, o endereço do seu roteador do primeiro salto (freqüentemente denominado default gateway) e o endereço do seu servidor DNS local.

Por causa de sua capacidade de automatizar os aspectos relativos à rede da conexão de um hospedeiro à rede, o DHPC é comumente denominado um **protocolo *plug and play***. Essa capacidade o torna *muito* atraente para o administrador de rede que, caso contrário, teria de executar essas tarefas manualmente! O DHPC também está conquistando ampla utilização em redes residenciais de acesso à Internet e em LANs sem fio, nas quais hospedeiros entram e saem da rede com freqüência. Um administrador de rede pode configurar o DHCP para que um dado hospedeiro receba um endereço IP permanente. Assim, cada vez que esse hospedeiro se juntar à rede, receberá o mesmo endereço IP. Mas muitos ISPs de empresas e residenciais não têm endereços IP suficientes para todos os seus hospedeiros. Quando isso acontece, o DHCP é utilizado para atribuir a cada um dos hospedeiros conectados um **endereço IP temporário.** Como exemplo, considere um ISP residencial que tem dois mil clientes, porém, nunca mais de 400 deles estão on-line ao mesmo tempo. O ISP não precisa de um bloco de dois mil endereços para administrar todos os seus dois mil clientes — usando um servidor DHCP para designar endereços dinamicamente, ele precisa de um bloco de apenas 512 endereços (por exemplo, um bloco da forma a.b.c.d/23). À medida que hospedeiros entram e saem da rede, o servidor DHCP precisa atualizar sua lista de endereços IP disponíveis. Toda vez que um hospedeiro se conectar à rede, o servidor DHPC designa a ele um endereço arbitrário do seu reservatório corrente de endereços disponíveis; toda vez que um hospedeiro sair, o endereço é devolvido ao reservatório.

Uma outra razão importante por que o DHCP conquistou uma utilização tão ampla, foi o advento da computação móvel. Considere, por exemplo, um estudante que leva seu laptop do seu quarto para a biblioteca, para a sala de aula. É provável que ele se conectará a uma nova sub-rede em cada um desses lugares e, por conseguinte, precisará de um novo endereço IP em cada um deles. O DHCP é ideal para essa situação, pois há muitos usuários em trânsito e os endereços são utilizados apenas por um tempo limitado.

Na verdade, o protocolo DHCP abrange a fronteira entre a rede e as camadas de rede e de enlace da pilha de protocolo de cinco camadas da Internet. Portanto, adiaremos a discussão detalhada da implementação do serviço DHCP para o Capítulo 5, que estuda a camada de enlace.

Tradução de endereços na rede (NAT)

Dada a nossa discussão sobre endereços de Internet e o formato do datagrama IPv4, agora estamos bem cientes de que todo equipamento que utiliza IP precisa de um endereço IP. Isso aparentemente significa que, com a proliferação de sub-redes de pequenos escritórios e de escritórios residenciais (*small office home office* — SOHO), sempre que um desses escritórios quiser instalar uma LAN para conectar várias máquinas, o ISP precisará alocar uma faixa de endereços para atender a todas as máquinas que usam IP do escritório em questão. Se a sub-rede ficasse maior (por exemplo, as crianças da casa não somente têm seus próprios computadores, mas também compraram PDAs, telefones com IP e jogos Game Boys em rede), seria preciso alocar um bloco de endereços maior. Mas, e se o ISP já tiver alocado as porções contíguas da faixa de endere-

ços utilizada atualmente por esse escritório residencial? E, antes de mais nada, qual é o proprietário típico de uma residência que quer (ou precisaria) saber como administrar endereços IP? Felizmente, há uma abordagem mais simples da alocação de endereços que vem conquistando uma utilização cada vez mais ampla nesses tipos de cenários: a **tradução de endereços de rede (*network address translation* — NAT)** [RFC 2663; RFC 3022].

A Figura 4.20 mostra a operação de um roteador que utiliza NAT. Esse roteador, que fica na residência do cidadão, tem uma interface que faz parte da rede residencial do lado direito da Figura 4.20. O endereçamento dentro da rede residencial é exatamente como vimos anteriormente — todas as quatro interfaces da rede têm o mesmo endereço de sub-rede, 10.00.0/24. O espaço de endereço 10.0.0.0/8 é uma das três porções do espaço de endereço IP reservado pelo [RFC 1918] para uma rede privada, ou um **domínio** com endereços privados, tal como a rede residencial da Figura 4.20. *Um domínio com endereços privados* refere-se a uma rede cujos endereços somente têm significado para equipamentos pertencentes àquela rede. Para ver por que isso é importante, considere o fato de haver centenas de milhares de redes residenciais, muitas delas utilizando o mesmo espaço de endereço 10.0.0.0/24. Equipamentos que pertencem a uma determinada rede residencial podem enviar pacotes entre si utilizando o endereçamento 10.0.0.0/24. Contudo, é claro que pacotes repassados da rede residencial para a Internet global não podem usar esses endereços (nem como endereço de fonte, nem como endereço de destino) porque há centenas de milhares de redes utilizando esse bloco de endereços. Isto é, os endereços 10.0.0.0/24 somente podem ter significado dentro daquela rede residencial. Mas, se endereços privados têm significado apenas dentro de uma dada rede, como o endereçamento é administrado quando pacotes são recebidos ou enviados para a Internet global, onde os endereços são necessariamente exclusivos? A resposta será dada pelo estudo da NAT. O roteador que usa NAT *não parece* um roteador para o mundo externo, pois se comporta como um equipamento *único* com um *único* endereço IP. Na Figura 4.20, todo o tráfego que sai do roteador residencial para a Internet tem um endereço IP de fonte 138.76.29.7, e todo o tráfego que entra nessa rede tem de ter endereço de destino 138.76.29.7. Em essência, o roteador que usa NAT está ocultando do mundo exterior os detalhes da rede residencial. (A propósito, você talvez esteja imaginando onde os computadores de redes residenciais obtêm seus endereços e onde o roteador obtém seu endereço IP exclusivo. A resposta é quase sempre a mesma — DHCP! O roteador obtém seu endereço do servidor DHPC do ISP e roda um servidor DHPC para fornecer endereços a computadores que estão dentro do espaço de endereços NAT da rede residencial controlado pelo DHCP).

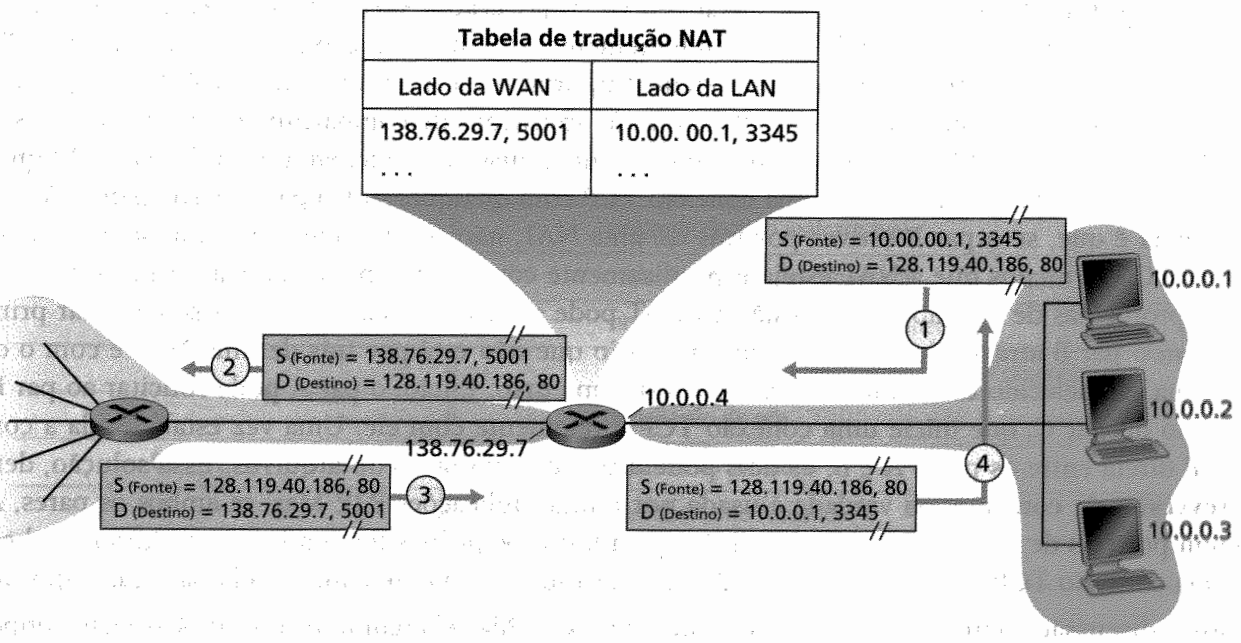


Figura 4.20 Tradução de endereços de rede

Se todos os datagramas que chegam ao roteador NAT provenientes da WAN tiverem o mesmo endereço IP de destino (especificamente, o endereço da interface do roteador NAT do lado da WAN), então como o roteador sabe para qual hospedeiro interno deve repassar um dado datagrama? O truque é utilizar uma **tabela de tradução NAT** no roteador NAT e incluir nos registros da tabela números de portas, bem como endereços IP.

Considere o exemplo da Figura 4.20. Suponha que um usuário que está utilizando o hospedeiro 10.0.0.1 da rede residencial requisite uma página Web de algum servidor Web (porta 80) cujo endereço IP é 128.119.40.186. O hospedeiro 10.0.0.1 escolhe o número de porta de fonte (arbitrário) 3345 e envia os datagramas para dentro da LAN. O roteador NAT recebe o datagrama, gera um novo número de porta de fonte, 5001, para o datagrama, substitui o endereço IP da fonte por seu endereço IP do lado da WAN, 138.76.29.7, e substitui o número de porta de fonte original, 3345, pelo novo número de porta de fonte, 5001. Ao gerar um novo número de porta de fonte, o roteador NAT pode selecionar qualquer número de porta de fonte que não esteja correntemente na tabela de tradução NAT. (Note que, como o comprimento de um campo de número de porta é 16 bits, o protocolo NAT pode suportar mais de 60 mil conexões simultâneas com um único endereço IP do lado da WAN para o roteador!). A NAT no roteador também adiciona um registro à sua tabela de tradução NAT. O servidor Web, totalmente alheio ao fato de que o datagrama que está chegando com uma requisição HTTP foi manipulado pelo roteador NAT, responde com um datagrama cujo endereço de destino é o endereço IP do roteador NAT, e cujo número de porta de destino é 5001. Quando esse datagrama chega ao roteador NAT, o roteador indexa a tabela de tradução NAT usando o endereço IP de destino e o número de porta de destino para obter o endereço IP (10.0.0.1) e o número de porta de destino (3345) adequados para o browser na rede residencial. O roteador então reescreve o endereço de destino e o número de porta de destino do datagrama e o repassa para a rede residencial.

A NAT conquistou ampla aceitação nos últimos anos. Mas devemos mencionar que muitos puristas da comunidade da IETF têm grandes restrições à NAT. Em primeiro lugar, argumentam, a finalidade dos números de portas é endereçar processos, e não hospedeiros. (Realmente, a violação dessa regra pode causar problemas para servidores que rodam em redes residenciais, pois, como vimos no Capítulo 2, processos servidores esperam pela chegada de requisições em números de portas bem conhecidos.) Em segundo lugar, argumentam que roteadores devem processar pacotes somente até a camada 3. Discutem, em terceiro lugar, que o protocolo NAT viola o argumento denominado fim-a-fim; isto é, hospedeiros devem falar diretamente uns com os outros, sem a interferência de nós que modifiquem endereços IP e números de portas. E, em quarto lugar, argumentam que deveríamos usar o IPv6 (veja Seção 4.4.4) para resolver a escassez de endereços IP, e não tentar resolver o problema imprudentemente com uma solução temporária como a NAT. Mas, gostemos ou não, a NAT tornou-se um componente importante da Internet. Outro problema importante da NAT é que ela interfere com aplicações P2P, incluindo aplicações de compartilhamento de arquivos P2P e aplicações de voz sobre IP. Lembre-se de que, no Capítulo 2, dissemos que, em uma aplicação P2P, qualquer par A participante deve poder iniciar uma conexão TCP com qualquer outro par B participante. A essência do problema é que, se o par B estiver por trás de uma NAT, não poderá agir como um servidor e aceitar conexões TCP (a menos que a NAT esteja especificamente configurada para a aplicação P2P). Como veremos nos problemas de fixação, essa questão da NAT pode ser contornada se o par A se conectar primeiramente com o par B através de um par C intermediário que não está por trás de uma NAT e com o qual o par B tenha estabelecido uma conexão TCP que está em curso. Então, o par A pode solicitar ao par B, por intermédio do par C, que inicie uma conexão TCP diretamente com ele. Uma vez estabelecida a conexão P2P TCP direta entre os pares A e B, os dois podem trocar mensagens ou arquivos. Essa solução, denominada **reversão de conexão**, na verdade é usada por muitas aplicações P2P. Mas, se ambos os pares, A e B, estiverem por trás de suas próprias NATs, então, para todos os propósitos práticos, será impossível estabelecer uma conexão TCP entre os dois se a NAT não for especificamente configurada para cada aplicação.

Nossa discussão sobre a NAT foi necessariamente breve. Mas ela tem muitos outros aspectos importantes, entre eles NAT estática *versus* NAT dinâmica e os efeitos da NAT sobre protocolos de camada mais alta e sobre a segurança. Se quiser mais detalhes, consulte uma discussão de prós e contras da NAT em [Cisco NAT, 2004; Phifer, 2000].

4.4.3 Protocolo de Mensagens de Controle da Internet (ICMP)

Lembre-se de que a camada de rede da Internet tem três componentes principais: o protocolo IP, discutido na seção anterior; os protocolos de roteamento da Internet (entre eles RIP, OSPF e BGP), que serão estudados na Seção 4.6; e o ICMP, objeto desta seção.

O ICMP, especificado no RFC 792, é usado por hospedeiros e roteadores para comunicar informações de camada de rede entre si. A utilização mais comum do ICMP é para comunicação de erros. Por exemplo, ao rodar uma sessão Telnet, FTP ou HTTP, é possível que você já tenha encontrado uma mensagem de erro como "Rede de destino inalcançável". Essa mensagem teve sua origem no ICMP. Em algum ponto, um roteador IP não conseguiu descobrir um caminho para o hospedeiro especificado em sua aplicação Telnet, FTP ou HTTP. O roteador criou e enviou uma mensagem ICMP do tipo 3 a seu hospedeiro indicando o erro.

O ICMP é freqüentemente considerado parte do IP, mas, em termos de arquitetura, está logo acima do IP, pois mensagens ICMP são carregadas dentro de datagramas IP. Isto é, mensagens ICMP são carregadas como carga útil IP, exatamente como segmentos TCP ou UDP, que também são carregados como carga útil IP. De maneira semelhante, quando um hospedeiro recebe um datagrama IP com ICMP especificado como protocolo de camada superior, ele demultiplexa o conteúdo do datagrama para ICMP, exatamente como demultiplexaria o conteúdo de um datagrama para TCP ou UDP.

Mensagens ICMP têm um campo de tipo e um campo de código. Além disso, contêm o cabeçalho e os primeiros 8 bytes do datagrama IP que causou a criação da mensagem ICMP em primeiro lugar (de modo que o remetente pode determinar o datagrama que causou o erro). Alguns tipos de mensagens ICMP selecionadas são mostradas na Figura 4.21. Note que mensagens ICMP não são usadas somente para sinalizar condições de erro.

O conhecido programa ping envia uma mensagem ICMP do tipo 8 código 0 para o hospedeiro especificado. O hospedeiro de destino, ao ver a solicitação de eco, devolve uma resposta de eco ICMP do tipo 0 código 0. A maior parte das implementações de TCP/IP suportam o servidor `ping` diretamente no sistema operacional; isto é, o servidor não é um processo. O Capítulo 11 de [Stevens, 1990] fornece o código-fonte para o programa `ping` cliente. Note que o programa cliente tem de ser capaz de instruir o sistema operacional para que ele gere uma mensagem ICMP do tipo 8 código 0.

Tipo de mensagem ICMP	Código	Descrição
0	0	resposta de eco (para ping)
3	0	rede de destino inalcançável
3	1	hospedeiro de destino inalcançável
3	2	protocolo de destino inalcançável
3	3	porta de destino inalcançável
3	6	rede de destino desconhecida
3	7	hospedeiro de destino desconhecido
4	0	redução da fonte (controle de congestionamento)
8	0	solicitação de eco
9	0	anúncio do roteador
10	0	descoberta do roteador
11	0	TTL expirado
12	0	cabeçalho IP inválido

Figura 4.21 Tipos de mensagens ICMP

Uma outra mensagem ICMP interessante é a de redução de fonte. Essa mensagem é pouco usada na prática. Sua finalidade original era realizar controle de congestionamento — permitir que um roteador congestionado enviasse uma mensagem ICMP de redução de fonte a um hospedeiro para obrigar esse hospedeiro a reduzir sua velocidade de transmissão. Vimos no Capítulo 3 que o TCP tem seu próprio mecanismo de controle de congestionamento, que funciona na camada de transporte sem usar realimentação da camada de rede tal como a mensagem ICMP de redução de fonte.

No Capítulo 1, apresentamos o programa Traceroute, que nos permite acompanhar a rota de um hospedeiro a qualquer outro hospedeiro no mundo. O interessante é que o Traceroute é implementado com mensagens ICMP. Para determinar os nomes e endereços de roteadores entre a fonte e o destino, o Traceroute da fonte envia uma série de datagramas comuns ao destino. O primeiro desses datagramas tem um TTL de 1, o segundo tem um TTL de 2, o terceiro tem um TLL de 3 e assim por diante. A fonte também aciona temporizadores para cada um dos datagramas. Quando o enésimo datagrama chega ao enésimo roteador, o enésimo roteador observa que o TTL do datagrama acabou de expirar. Segundo as regras do protocolo IP, o roteador descarta o datagrama e envia uma mensagem ICMP de aviso à fonte (tipo 11 código 0). Essa mensagem de aviso inclui o nome do roteador e seu endereço IP. Quando chega à fonte, a mensagem obtém, do temporizador, o tempo de viagem de ida e volta e, da mensagem ICMP, o nome e o endereço IP do enésimo roteador.

Como uma fonte Traceroute sabe quando parar de enviar segmentos UDP? Lembre-se de que a fonte incrementa o campo do TTL para cada datagrama que envia. Assim, um dos datagramas eventualmente conseguirá chegar ao hospedeiro de destino. Como esse datagrama contém um segmento UDP com um número de porta improvável, o hospedeiro de destino devolve à fonte uma mensagem ICMP indicando que a porta não pôde ser alcançada (mensagem tipo 3, código 3). Quando recebe essa mensagem ICMP particular, o hospedeiro da fonte sabe que não precisa enviar mais pacotes de sondagem. (Na verdade, o programa Traceroute padrão envia conjuntos de três pacotes com o mesmo TTL; assim, o Traceroute provê três resultados para cada TTL.)

Desse modo, o hospedeiro da fonte fica a par do número e das identidades de roteadores que estão entre ele e o hospedeiro de destino e o tempo de viagem de ida e volta entre os dois hospedeiros. Note que o programa cliente Traceroute tem de ser capaz de instruir o sistema operacional para que este gere datagramas UDP com valores específicos de TTL. Também tem de poder ser avisado por seu sistema operacional quando chegam mensagens ICMP.

Agora que você entende como o Traceroute funciona, é provável que queira voltar e brincar um pouco com ele.

4.4.4 IPv6

No começo da década de 1990, a IETF iniciou um esforço para desenvolver o sucessor do protocolo IPv4. Uma motivação primária para esse esforço foi o entendimento de que o espaço de endereços IP de 32 bits estava começando a escassear, com novas sub-redes e nós IP sendo anexados à Internet (e ainda recebendo endereços IP exclusivos) a uma velocidade estonteante. Para atender a essa necessidade de maior espaço para endereços IP, um novo protocolo IP, o IPv6, foi desenvolvido. Os projetistas do IPv6 também aproveitaram essa oportunidade para ajustar e ampliar outros aspectos do IPv4 com base na experiência operacional acumulada sobre esse protocolo.

O momento em que todos os endereços IPv4 estariam alocados (e, por conseguinte, mais nenhuma sub-rede poderia ser ligada à Internet) foi objeto de considerável debate. Com base nas tendências correntes sobre alocação de endereços existentes na época, os dois líderes do grupo de trabalho Expectativa de Tempo de Vida dos Endereços (Address Lifetime Expectations) da IETF estimaram que os endereços se esgotariam em 2008 e 2018, respectivamente [Solensky, 1996]. Em 1996, o Registro Americano para Números da Internet (American Registry for Internet Numbers — ARIN) informou que já tinham sido alocados todos os endereços da classe A do IPv4, 62 por cento dos endereços da classe B e 37 por cento dos

endereços da classe C [ARIN, 1996]. Embora essas estimativas e números sugerissem que havia um tempo considerável até que o espaço de endereços IPv4 fosse exaurido, ficou claro que seria necessário um tempo expressivo para disponibilizar uma nova tecnologia em escala tão gigantesca. Assim, foi dado início ao esforço denominado Próxima Geração do IP (Next Generation IP — IPng) [Bradner, 1996; RFC 1752]. O resultado desse esforço foi a especificação IP versão 6 (IPv6) [RFC 2460]. (Uma pergunta que sempre é feita é o que aconteceu com o IPv5. Foi proposto inicialmente que o protocolo ST-2 se tornasse o IPv5, porém, mais tarde, este protocolo foi descartado em favor do RSVP, que discutiremos no Capítulo 7.)

Excelentes fontes de informação sobre o IPv6 são a página IP Next Generation [Hinden, 2004] e um excelente livro sobre o assunto é de [Huitema, 1998].

Formato do datagrama IPv6

O formato do datagrama IPv6 é mostrado na Figura 4.22. As mudanças mais importantes introduzidas no IPv6 ficam evidentes no formato do datagrama:

- *Capacidade de endereçamento expandida.* O IPv6 aumenta o tamanho do endereço IP de 32 bits para 128 bits. Isso garante que o mundo não ficará sem endereços IP. Agora cada grão de areia do planeta pode ter um endereço IP. Além dos endereços *multicast* e *unicast*, o IPv6 introduziu um novo tipo de endereço, denominado **endereço anycast**, que permite que um datagrama seja entregue a qualquer hospedeiro de um grupo. (Essa característica poderia ser usada, por exemplo, para enviar uma mensagem HTTP GET ao site mais próximo de um conjunto de sites espelhados que contenham um dado documento.)
- *Cabeçalho aprimorado de 40 bytes.* Como discutiremos adiante, vários campos IPv4 foram descartados ou tornaram-se opcionais. O cabeçalho de comprimento fixo de 40 bytes resultante permite processamento mais veloz do datagrama IP. Uma nova codificação de opções permite um processamento de opções mais flexível.
- *Rotulação de fluxo e prioridade.* O IPv6 tem uma definição dúbia de **fluxo**. O RFC 1752 e o RFC 2460 declaram que isso permite "rotular pacotes que pertencem a fluxos particulares para os quais o remetente requisita tratamento especial, tal como um serviço de qualidade não padrão ou um serviço de tempo real". Por exemplo, a transmissão de áudio e vídeo poderia ser tratada como um fluxo. Por outro lado, as aplicações mais tradicionais, como transferência de arquivos e e-mail, poderiam não ser tratadas como fluxos. É possível que o tráfego carregado por um usuário de alta prioridade (digamos, alguém que paga por um serviço melhor de tráfego) seja também tratado como um fluxo. O que fica claro, contudo, é que os projetistas do IPv6 prevêem a possível necessidade de poder diferenciar os fluxos, mesmo que o exato significado de fluxo ainda não tenha sido determinado. O cabeçalho IPv6 também tem um campo de 8 bits para classe de tráfego. Esse campo, como o campo TOS do IPv4, pode ser usado para dar prioridade a certos datagramas dentro de um fluxo ou a datagramas de certas aplicações (por exemplo, pacotes ICMP) em relação a datagramas de outras aplicações (por exemplo, notícias pela rede).

Como foi observado anteriormente, uma comparação entre as Figuras 4.22 e 4.13 revela uma estrutura mais simples e mais aprimorada para o datagrama IPv6. Os seguintes campos são definidos no IPv6:

- *Versão.* Esse campo de 4 bits identifica o número da versão do IP. Não é surpresa que o IPv6 tenha o valor 6 nesse campo. Note que colocar 4 nesse campo não cria um datagrama IPv4 válido. (Se criasse, a vida seria bem mais simples — veja a discussão mais adiante, referente à transição do IPv4 para o IPv6.)
- *Classe de tráfego.* Esse campo de 8 bits tem função semelhante à do campo TOS que vimos no IPv4.
- *Rótulo de fluxo.* Como discutimos antes, esse campo de 20 bits é usado para identificar um fluxo de datagramas.

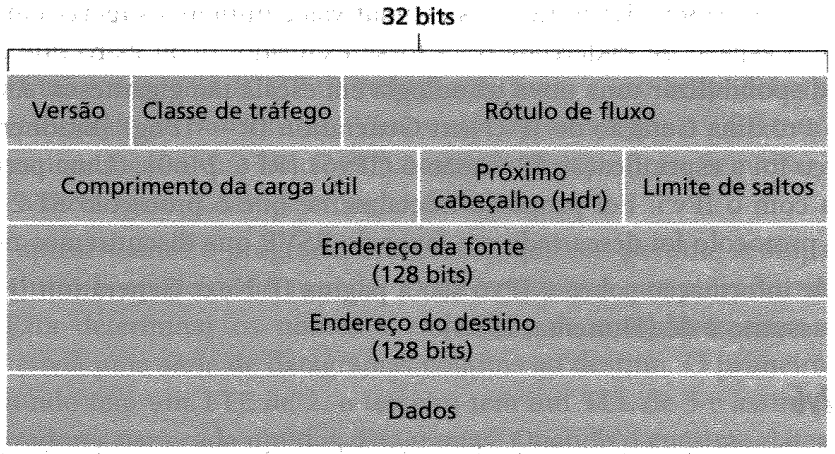


Figura 4.22 Formato do datagrama IPv6

- *Comprimento da carga útil*. Esse valor de 16 bits é tratado como um número inteiro sem sinal que dá o número de bytes no datagrama IPv6 que se segue ao pacote do cabeçalho, que tem tamanho fixo de 40 bytes.
- *Próximo cabeçalho*. Esse campo identifica o protocolo ao qual o conteúdo (campo de dados) desse datagrama será entregue (por exemplo, TCP ou UDP). Usa os mesmos valores do campo de protocolo no cabeçalho IPv4.
- *Limite de saltos*. O conteúdo desse campo é decrementado de um para cada roteador que repassa o datagrama. Se a contagem do limite de saltos chegar a zero, o datagrama será descartado.
- *Endereços de fonte e de destino*. Os vários formatos do endereço de 128 bits do IPv6 são descritos no RFC 2373.
- *Dados*. Esta é a parte da carga útil do datagrama IPv6. Quando o datagrama alcança seu destino, a carga útil pode ser extraída do datagrama IP e passada adiante para o protocolo especificado no campo próximo cabeçalho.

Nessa discussão, apresentamos a finalidade dos campos que estão incluídos no datagrama IPv6. Quando comparamos o formato do datagrama IPv6 da Figura 4.22 com o formato do datagrama IPv4 que vimos na Figura 4.13, notamos que diversos campos que aparecem no datagrama IPv4 não estão presentes no datagrama IPv6:

- *Fragmentação/remontagem*. O IPv6 não permite fragmentação e remontagem em roteadores intermediários; essas operações podem ser realizadas somente pela fonte e pelo destino. Se um datagrama IPv6 recebido por um roteador for muito grande para ser repassado pelo enlace de saída, o roteador simplesmente descartará o datagrama e devolverá ao remetente uma mensagem ICMP de erro "Pacote muito grande" (veja a seguir). O remetente pode então reenviar os dados usando um datagrama IP de tamanho menor. Fragmentação e remontagem são operações que tomam muito tempo; retirar essas funcionalidades dos roteadores e colocá-las nos sistemas finais acelera consideravelmente o repasse IP para dentro da rede.
- *Soma de verificação do cabeçalho*. Como os protocolos de camada de transporte (por exemplo, TCP e UDP) e de enlace de dados (por exemplo, Ethernet) nas camadas da Internet realizam soma de verificação, os projetistas do IP provavelmente acharam que essa funcionalidade era tão redundante na camada de rede que podia ser retirada. Mais uma vez, o processamento rápido de pacotes IP era uma preocupação principal. Lembre-se de que em nossa discussão sobre o IPv4 na Seção 4.4.1 vimos que, como o cabeçalho IPv4 contém um campo TTL (semelhante ao campo de limite de saltos no IPv6), a soma de verificação do cabeçalho IPv4 precisava ser recalculada em cada rotea-

dor. Como acontece com a fragmentação e a remontagem, esta também era uma operação de alto custo no IPv4.

Opções. O campo de opções não faz mais parte do cabeçalho-padrão do IP. Contudo, ele ainda não saiu de cena. Em vez disso, passou a ser um dos possíveis próximos cabeçalhos que poderão ser apontados pelo cabeçalho do IPv6. Em outras palavras, o campo de opções também poderá ser o próximo cabeçalho dentro de um pacote IP, exatamente como os cabeçalhos dos protocolos TCP e UDP. A remoção do campo de opções resulta em um cabeçalho IP de tamanho fixo de 40 bytes.

Lembre-se de que dissemos na Seção 4.4.3 que o protocolo ICMP é usado pelos nós IP para informar condições de erro e fornecer informações limitadas (por exemplo, a resposta de eco para uma mensagem ping) a um sistema final. Uma nova versão do ICMP foi definida para o IPv6 no RFC 2463. Além da reorganização das definições existentes de tipos e códigos ICMP, o ICMPv6 adicionou novos tipos e códigos requeridos pela nova funcionalidade do IPv6. Entre eles estão incluídos o tipo "Pacote muito grande" e um código de erro "opções IPv6 não reconhecidas". Além disso, o ICMPv6 incorpora a funcionalidade do IGMP (Internet Group Management Protocol — protocolo de gerenciamento de grupos da Internet), que estudaremos na Seção 4.7. O IGMP, que é usado para gerenciar a adesão ou a saída de um hospedeiro de grupos *multicast*, era anteriormente um protocolo separado do ICMP no IPv4.

Transição do IPv4 para o IPv6

Agora que vimos os detalhes técnicos do IPv6, vamos tratar de um assunto muito prático: como a Internet pública, que é baseada no IPv4, fará a transição para o IPv6? O problema é que, enquanto os novos sistemas habilitados para IPv6 podem ser inversamente compatíveis, isto é, podem enviar, rotear e receber datagramas IPv4, os sistemas habilitados para IPv4 não podem manusear datagramas IPv6. Há várias opções possíveis.

Uma opção seria determinar um "dia da conversão" — uma data e um horário determinados em que todas as máquinas da Internet seriam desligadas e atualizadas, passando do IPv4 para o IPv6. A última transição importante de tecnologia (do uso do NPC para o uso do TCP para serviço confiável de transporte) ocorreu há quase 20 anos. E, mesmo naquela época [RFC 801], quando a Internet era pequenina e ainda era gerenciada por um número reduzido de 'sabichões', ficou claro que esse "dia da conversão" não era possível. Um dia como esse, envolvendo centenas de milhares de máquinas e milhões de gerenciadores de rede, é ainda mais impensável hoje. O RFC 2893 descreve duas abordagens (que podem ser usadas independentemente ou em conjunto) para a integração gradual dos hospedeiros e roteadores IPv4 ao mundo IPv6 (com a meta de longo prazo de fazer a transição de todos os nós IPv4 para IPv6).

Provavelmente, a maneira mais direta de introduzir nós habilitados ao IPv6 seja uma abordagem de **pilha dupla**, em que nós IPv6 também tenham uma implementação IPv4 completa. Esse nó, denominado nó IPv6/IPv4 no RFC 2893, estaria habilitado a enviar e receber tanto datagramas IPv4 quanto datagramas IPv6. Ao interagir com um nó IPv4, um nó IPv6/IPv4 poderá usar datagramas IPv4; ao interagir com um nó IPv6, poderá utilizar IPv6. Nós IPv6/IPv4 devem ter endereços IPv6 e IPv4. Além disso, devem poder determinar se outro nó é habilitado para IPv6 ou somente para IPv4. Esse problema pode ser resolvido usando o DNS (veja o Capítulo 2), que poderá retornar um endereço IPv6 se o nome do nó a ser resolvido for capacitado para IPv6. Caso contrário, ele retornará um endereço IPv4. É claro que, se o nó que estiver emitindo a requisição DNS for habilitado apenas para IPv4, o DNS retornará apenas um endereço IPv4.

Na abordagem de pilha dupla, se o remetente ou o destinatário forem habilitados apenas para IPv4, um datagrama IPv4 deverá ser usado. Como resultado, é possível que dois nós habilitados para IPv6 acabem, em essência, enviando datagramas IPv4 um para o outro. Isso é ilustrado na Figura 4.23. Suponha que o nó A utiliza IPv6 e queira enviar um datagrama IP ao nó F, que também utiliza IPv6. Os nós A e B podem trocar um datagrama IPv6. Contudo, o nó B deve criar um datagrama IPv4 para enviar a C. É claro que o campo de dados do datagrama IPv6 pode ser copiado para o campo de dados do datagrama IPv4 e que o mapeamento adequado de endereço também pode ser feito. No entanto, ao realizar a conversão de IPv6 para IPv4,

haverá campos IPv6 específicos no datagrama IPv6 (por exemplo, o campo do identificador de fluxo) que não terão contrapartes em IPv4. As informações contidas nesses campos serão perdidas. Assim, mesmo que E e F possam trocar datagramas IPv6, os datagramas IPv4 que chegarem a E e D não conterão todos os campos que estavam no datagrama IPv6 original enviado de A.

Uma alternativa para a abordagem de pilha dupla, também discutida no RFC 1933, é conhecida como **implantação de túnel**. O túnel pode resolver o problema observado anteriormente, permitindo, por exemplo, que E receba o datagrama IPv6 originado por A. A idéia básica por trás da implementação do túnel é a seguinte. Suponha que dois nós IPv6 (por exemplo, B e E na Figura 4.23) queiram interagir usando datagramas IPv6, mas estão conectados um ao outro por roteadores intervenientes IPv4. Referimo-nos ao conjunto de roteadores intervenientes IPv4 entre dois roteadores IPv6 como um **túnel**, como ilustrado na Figura 4.24.

Com a implementação do túnel, o nó IPv6 no lado remetente do túnel (por exemplo, B) pega o datagrama IPv6 inteiro e o coloca no campo de dados (carga útil) de um datagrama IPv4. Esse datagrama IPv4 é então endereçado ao nó IPv6 no lado receptor do túnel (por exemplo, E) e enviado ao primeiro nó do túnel (por exemplo, C). Os roteadores IPv4 intervenientes no túnel roteiam esse datagrama IPv4 entre eles, exatamente como fariam com qualquer outro datagrama, alheios ao fato de que o datagrama IPv4 contém um datagrama IPv6 completo. O nó IPv6 do lado receptor do túnel eventualmente recebe o datagrama IPv4 (pois ele é o destino do datagrama IPv4!), determina que o datagrama IPv4 contém um datagrama IPv6, extrai o datagrama IPv6 e, então, roteia o datagrama IPv6 exatamente como o faria se tivesse recebido o datagrama IPv6 de um vizinho IPv6 diretamente ligado a ele.

Encerramos esta seção mencionando que a adoção do IPv6 está demorando para decolar [Lawton, 2001]. Lembre-se de que uma das principais motivações para o IPv6 era o esgotamento de endereços IPv4 disponíveis. Vimos na Seção 4.4.2 que avanços como endereços IPv4 "ciderizados", DHCP e NAT contribuíram para a solução desse problema, pelo menos no curto prazo. É possível, entretanto, que a proliferação de dispositivos como telefones IP e outros dispositivos portáteis possam dar o empurrão que falta para a aplicação mais ampla do IPv6.

O programa europeu Third Generation Partnership [3GPP, 2004] especificou o IPv6 como o padrão de endereçamento para multimídia móvel. Mesmo que o IPv6 não tenha sido disponibilizado amplamente nos primeiros nove anos de sua jovem vida, está bem claro que é preciso uma visão de longo prazo. O sistema de números de telefones existente hoje demorou várias décadas para se firmar, mas está vigente há quase meio século e não dá nenhum sinal de que pretende sair de cena. De modo semelhante, pode levar algum tempo para o IPv6 se firmar, mas, depois, provavelmente ele também será utilizado durante muito tempo. Brian Carpenter, antigo diretor do Internet Architecture Board [IAB, 2004], e autor de diversos RFCs relacionados ao IPv6, declarou: "Eu sempre considerei a adoção do IPv6 como um processo de 15 anos, começando em 1995." [Lawton, 2001]. Pelas contas de Carpenter, já estamos perto dos dois-terços desse tempo!

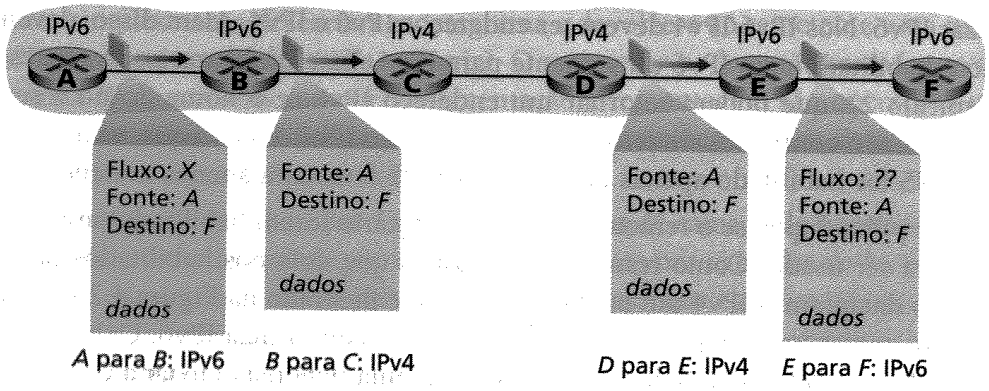


Figura 4.23 Abordagem de pilha dupla

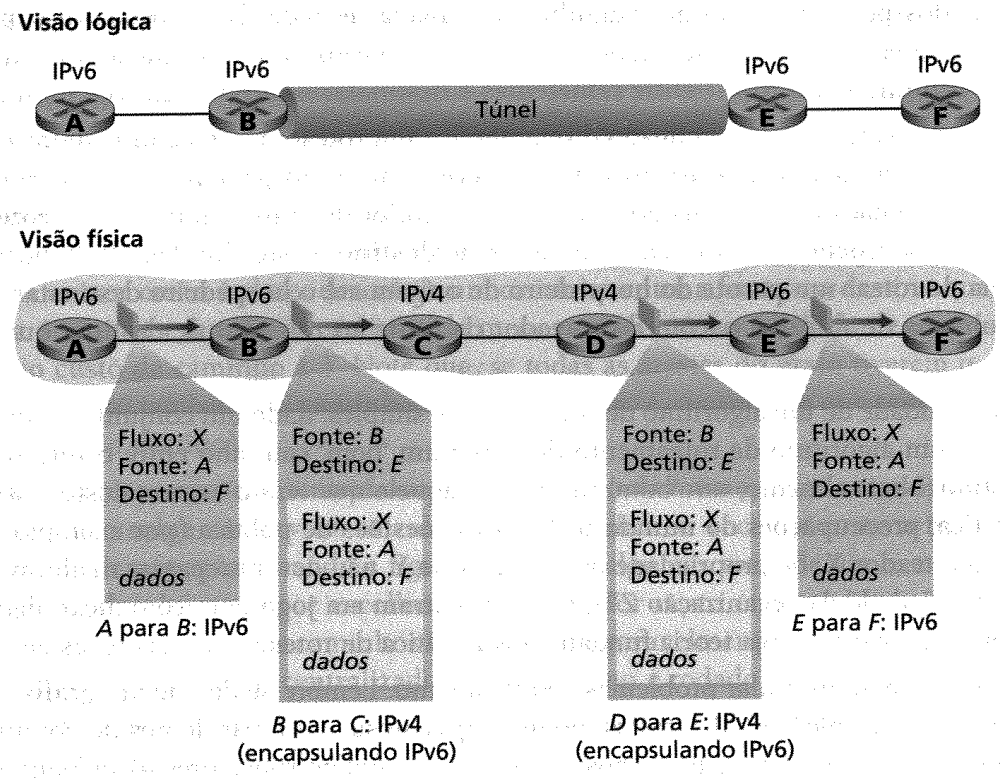


Figura 4.24 Implementação de túnel

Uma lição importante que podemos aprender com a experiência do IPv6 é que há enorme dificuldade para mudar protocolos de camada de rede. Desde o início da década de 1990, numerosos novos protocolos de camada de rede foram anunciados como a próxima maior revolução da Internet, mas a maioria desses protocolos teve pouca aceitação até agora. Dentre eles, estão o IPv6, os protocolos *multicast* (veja a Seção 4.7) e os protocolos de reserva de recursos (Capítulo 7). Na verdade, introduzir novos protocolos na camada de rede é como substituir as fundações de uma casa — é difícil de fazer sem demolir a casa inteira ou, no mínimo, retirar os moradores temporariamente da residência. Por outro lado, a Internet vem testemunhando a rápida disponibilização de novos protocolos na camada de aplicação. Os exemplos clássicos são, é claro, a Web, a mensagem instantânea e o compartilhamento de arquivos P2P. Outros exemplos são a recepção de áudio e vídeo em tempo real e os jogos distribuídos. Introduzir novos protocolos de camada de aplicação é como pintar uma casa — é relativamente fácil de fazer e, se você escolher uma cor atraente, outras casas da vizinhança vão imitá-lo. Em resumo, no futuro podemos esperar mudanças na camada de rede da Internet, mas essas mudanças provavelmente ocorrerão dentro de uma escala de tempo bem mais lenta do que as que ocorrerão na camada de aplicação.

4.5 Algoritmos de roteamento

Exploramos, neste capítulo, a função de repasse da camada de rede mais do que qualquer outra. Aprendemos que, quando um pacote chega a um roteador, o roteador indexa uma tabela de repasse e determina a interface de enlace para a qual o pacote deve ser dirigido. Aprendemos também que algoritmos de roteamento que rodam em roteadores de rede trocam e calculam as informações que são utilizadas para configurar essas tabelas de repasse. A interação entre algoritmos de roteamento e tabelas de repasse foi ilustrada na Figura 4.2. Agora que já nos aprofundamos um pouco na questão do repasse, voltaremos nossa atenção para o outro tópico importante desse capítulo, a saber, a função crítica da camada de rede, o roteamento. Quer ofereça um serviço de datagramas (quando pacotes diferentes entre um determinado par fonte-destino podem pegar rotas diferentes) ou um serviço de VCs (quando todos os pacotes entre uma fonte e um

destino determinados pegarão o mesmo caminho), a camada de rede deve, mesmo assim, determinar o caminho que os pacotes percorrem entre remetentes e destinatários. Veremos que a tarefa do roteamento é determinar bons caminhos (ou rotas) entre remetentes e destinatários através da rede de roteadores.

Normalmente um hospedeiro está ligado diretamente a um roteador, o **roteador default** para esse hospedeiro (também denominado **roteador do primeiro salto** para o hospedeiro). Sempre que um hospedeiro emitir um pacote, o pacote será transferido para seu roteador default. Denominamos **roteador da fonte** o roteador default do hospedeiro de origem e **roteador de destino** o roteador default do hospedeiro de destino. O problema de rotear um pacote do hospedeiro de origem até o hospedeiro destinatário se reduz, claramente, ao problema de rotear o pacote do roteador da fonte ao roteador de destino, que é o foco desta seção.

Portanto, a finalidade de um algoritmo de roteamento é simples: dado um conjunto de roteadores conectados por enlaces, um algoritmo de roteamento descobre um 'bom' caminho entre o roteador de fonte e o roteador de destino. Normalmente, um 'bom' caminho é aquele que tem o 'menor custo'. No entanto, veremos que, na prática, preocupações do mundo real, como questões de política (por exemplo, uma regra que determina que "o roteador X, de propriedade da organização Y, não deverá repassar nenhum pacote originário da rede de propriedade da organização Z"), também entram em jogo para complicar algoritmos conceitualmente simples e elegantes, cuja teoria fundamenta a prática de roteamento nas redes de hoje.

Um grafo é usado para formular problemas de roteamento. Lembre-se de que um **grafo** $G = (N,E)$ é um conjunto N de nós e uma coleção E de arestas, no qual cada aresta é um par de nós do conjunto N. No contexto do roteamento da camada de rede, os nós do grafo representam roteadores — os pontos nos quais são tomadas decisões de repasse de pacotes — e as arestas que conectam esses nós representam os enlaces físicos entre esses roteadores. Uma abstração gráfica de uma rede de computadores está ilustrada na Figura 4.25.

Para ver alguns grafos que representam mapas de redes reais, consulte [Dodge, 2004, Cheswick, 2000]. Para uma discussão sobre quão bem os diferentes modelos fundamentados em grafos modelam a Internet, consulte [Zegura, 1997, Faloutsos, 1999].Como ilustrado na Figura 4.25, uma aresta também tem um valor que representa seu custo. Normalmente, o custo de uma aresta pode refletir o tamanho físico do enlace correspondente (por exemplo, um enlace transoceânico poderia ter um custo mais alto do que um enlace terrestre de curta distância), a velocidade do enlace ou o custo monetário a ele associado. Para nossos objetivos, consideraremos os custos da aresta apenas como um dado e não nos preocuparemos com o modo como eles são determinados. Para qualquer aresta (x,y) em E, denominamos $c(x,y)$ o custo da aresta entre os nós x e y. Se o par (x,y) não pertencer a E, estabelecemos $c(x,y) = \infty$. Além disso, sempre consideraremos somente grafos não direcionados (isto é, grafos cujas arestas não têm uma direção), de modo que a aresta (x,y) é a mesma que a aresta (y,x) e $c(x,y) = c(y,x)$. Dizemos também que y é um **vizinho** do nó x se (x,y) pertencer a E.

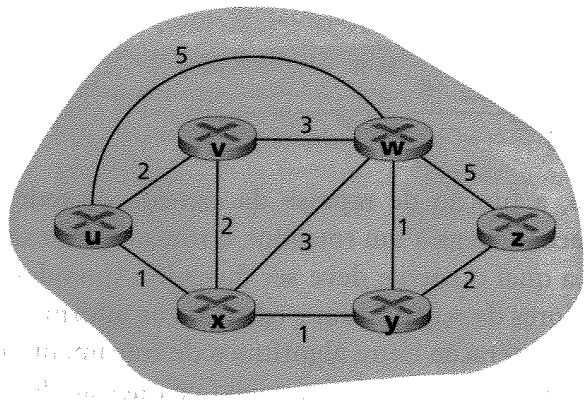


Figura 4.25 Modelo abstrato de grafo de uma rede de computadores

Dado que são atribuídos custos às várias arestas na abstração do grafo, uma meta natural de um algoritmo de roteamento é identificar o caminho de menor custo entre fontes e destinos. Para tornar esse problema mais preciso, lembremos que um **caminho** em um grafo $G = (N,E)$ é uma seqüência de nós $(x_1, x_2,..., x_p)$ tal que cada um dos pares (x_1, x_2), $(x_2, x_3),...,(x_{p-1}, x_p)$ são arestas em E. O custo de um caminho $(x_1, x_2,..., x_p)$ é simplesmente a soma de todos os custos das arestas ao longo do caminho, ou seja, $c(x_1, x_2) + c(x_2, x_3) + ... + c(x_{p-1},x_p)$. Dados quaisquer dois nós x e y, normalmente há muitos caminhos entre os dois, e cada caminho tem um custo. Um ou mais desses caminhos é um **caminho de menor custo**. Por conseguinte, o problema do menor custo é claro: descobrir um caminho entre a fonte e o destino que tenha o menor custo. Na Figura 4.25, por exemplo, o caminho de menor custo entre o nó da fonte u e o nó de destino w é (u, x, y, w), cujo custo de caminho é 3. Note que, se todas as arestas do grafo tiverem o mesmo custo, o caminho de menor custo também é o **caminho mais curto** (isto é, o caminho que tem o menor número de enlaces entre a fonte e o destino).

Apenas como simples exercício, tente descobrir o caminho de menor custo entre os nós u e z na Figura 4.25 e reflita um pouco sobre como você calculou esse caminho. Se você for como a maioria das pessoas, descobriu o caminho de u a z examinando a figura, traçando algumas rotas de u a z, e se convencendo, de algum modo, que o caminho que você escolheu tinha o menor custo entre todos os caminhos possíveis. (Você verificou todos os 17 possíveis caminhos entre u e z? Provavelmente, não!) Esse cálculo é um exemplo de um algoritmo de roteamento centralizado — o algoritmo é rodado em um local, o seu cérebro, com informações completas sobre a rede. De modo geral, uma maneira possível de classificar algoritmos de roteamento é como globais ou centralizados.

- Um **algoritmo de roteamento global** calcula o caminho de menor custo entre uma fonte e um destino usando conhecimento completo e global sobre a rede. Em outras palavras, o algoritmo considera como dados de cálculo a conectividade entre todos os nós e todos os custos dos enlaces. E isso exige que o algoritmo obtenha essas informações, de algum modo, antes de realmente realizar o cálculo. O cálculo em si pode ser rodado em um local (um algoritmo de roteamento global centralizado) ou duplicado em vários locais. Contudo, a principal característica distintiva, nesse caso, é que um algoritmo global tem informação completa sobre conectividade e custo de enlaces. Na prática, algoritmos com informação global de estado são freqüentemente denominados **algoritmos de estado de enlace** (*link-state* — LS), já que devem estar a par dos custos de cada enlace na rede. Estudaremos algoritmos de estado de enlace na Seção 4.5.1
- Em um **algoritmo de roteamento descentralizado**, o cálculo do caminho de menor custo é realizado de modo iterativo e distribuído. Nenhum nó tem informação completa sobre os custos de todos os enlaces da rede. Em vez disso, cada nó começa sabendo apenas os custos dos enlaces diretamente ligados a ele. Então, por meio de um processo iterativo de cálculo e de troca de informações com seus nós vizinhos (isto é, nós que estão na outra extremidade dos enlaces aos quais ele próprio está ligado), um nó gradualmente calcula o caminho de menor custo até um destino ou um conjunto de destinos. O algoritmo de roteamento descentralizado que estudaremos logo adiante na Seção 4.5.2 é denominado **algoritmo de vetor de distâncias** (*distance-vector algorithm* — DV) porque cada nó mantém um vetor de estimativas de custos (distâncias) de um nó até todos os outros nós da rede.

Uma segunda maneira geral de classificar algoritmos de roteamento é como estáticos ou dinâmicos. Em **algoritmos de roteamento estáticos**, as rotas mudam muito lentamente ao longo do tempo, muitas vezes como resultado de intervenção humana (por exemplo, uma pessoa editando manualmente a tabela de repasse do roteador). **Algoritmos de roteamento dinâmicos** mudam os caminhos de roteamento à medida que mudam as cargas de tráfego ou a topologia da rede. Um algoritmo dinâmico pode ser rodado periodicamente ou como reação direta à mudanças de topologia ou de custo dos enlaces. Ao mesmo tempo que são mais sensíveis à mudanças na rede, algoritmos dinâmicos também são mais suscetíveis a problemas como loops de roteamento e oscilação em rotas.

Uma terceira maneira de classificar algoritmos de roteamento é como sensíveis à carga ou insensíveis à carga. Em um **algoritmo sensível à carga**, custos de enlace variam dinamicamente para refletir o nível corrente de congestionamento no enlace subjacente. Se houver um alto custo associado com um enlace que está correntemente congestionado, um algoritmo de roteamento tenderá a escolher rotas que evitem esse enlace congestionado. Embora antigos algoritmos da ARPAnet fossem sensíveis à carga [McQuillan, 1980], foram encontradas várias dificuldades [Huitema, 1998]. Os algoritmos de roteamento utilizados na Internet hoje (como RIP, OSPF e BGP) são **insensíveis à carga**, pois o custo de um enlace não reflete explicitamente seu nível de congestionamento corrente (nem o de congestionamento mais recente).

4.5.1 O algoritmo de roteamento de estado de enlace (LS)

Lembre-se de que em um algoritmo de estado de enlace a topologia da rede e todos os custos de enlace são conhecidos, isto é, estão disponíveis como dados para o algoritmo de estado de enlace. Na prática, isso se consegue fazendo com que cada nó transmita pacotes de estado de enlace a todos os outros nós da rede, sendo que cada um desses pacotes contém as identidades e os custos dos enlaces ligados a ele. Na prática (por exemplo, com o protocolo de roteamento OSPF da Internet, discutido na Seção 4.6.1) isso freqüentemente é conseguido com um algoritmo de **transmissão broadcast de estado de enlace** [Perlman, 1999]. Algoritmos de transmissão *broadcast* serão estudados na Seção 4.7. O resultado da transmissão *broadcast* dos nós é que todos os nós têm uma visão idêntica e completa da rede. Cada nó pode, então, rodar o algoritmo de estado de enlace e calcular o mesmo conjunto de caminhos de menor custo como todos os outros nós.

O algoritmo de roteamento de estado de enlace que apresentamos adiante é conhecido como *algoritmo de Dijkstra*, o nome de seu inventor. Um algoritmo que guarda relações muito próximas com ele é o algoritmo de Prim. Consulte [Cormen, 1990] para uma discussão geral sobre algoritmos de grafo. O algoritmo de Dijkstra calcula o caminho de menor custo entre um nó (a fonte, que chamaremos de u) e todos os outros nós da rede. É um algoritmo iterativo e tem a propriedade de, após a k-ésima iteração, conhecer os caminhos de menor custo para k nós de destino e, dentre os caminhos de menor custo até todos os nós de destino, esses k caminhos terão os k menores custos. Vamos definir a seguinte notação:

- $D(v)$: custo do caminho de menor custo entre o nó da fonte e o destino v até essa iteração do algoritmo.
- $p(v)$: nó anterior (vizinho de v) ao longo do caminho de menor custo corrente desde a fonte até v.
- N': subconjunto de nós; v pertence a N' se o caminho de menor custo entre a fonte e v for inequivocamente conhecido.

O algoritmo de roteamento global consiste em uma etapa de inicialização seguida de um loop. O número de vezes que o loop é rodado é igual ao número de nós na rede. Ao terminar, o algoritmo terá calculado os caminhos mais curtos desde o nó da fonte u até todos os outros nós da rede.

Algoritmo de estado de enlace para o nó da fonte u

```
1  Inicialização:
2          N' = {u}
3          para todos os nós v
4                  se v for um vizinho de u
5                          então D(v)= c(u,v)
6                          senão D(v) = ∞
7
8  Loop
9          encontre w não em N', tal que D(w) é um mínimo
```

```
10          adicione w a N'
11          atualize D(v) para cada vizinho v de w e não em N':
12                  D(v) = min( D(v), D(w) + c(w, v) )
13          /*o novo custo para v é o velho custo para v
14          ou o custo do menor caminho conhecido para w mais o custo de w para v */
15  até N'= N
```

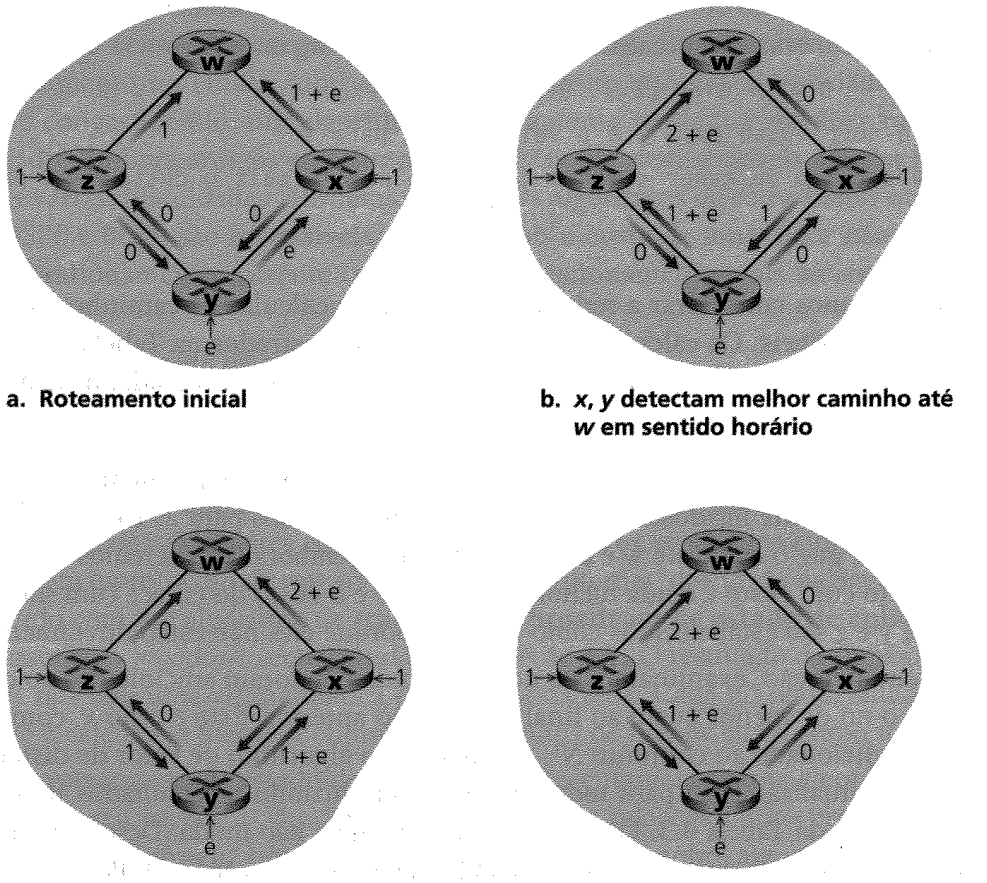
Como exemplo, vamos considerar a rede da Figura 4.25 e calcular os caminhos de menor custo de *u* até todos os destinos possíveis. Os cálculos do algoritmo estão ilustrados na Tabela 4.3, na qual cada linha fornece os valores das variáveis do algoritmo ao final da iteração. Vamos considerar detalhadamente alguns dos primeiros estágios:

- No estágio de inicialização, os caminhos de menor custo correntemente conhecidos de *u* até os vizinhos diretamente ligados a ele (*v*, *w* e *x*) são inicializados para 2, 1 e 5, respectivamente. Note, em particular, que o custo até w é estabelecido em 5 (embora logo veremos que, na realidade, existe um trajeto cujo custo é ainda menor), já que este é o custo do enlace (um salto) direto *u* a *w*. Os custos até *y* e *z* são estabelecidos como infinito, porque eles não estão diretamente conectados a *u*.
- Na primeira iteração, examinamos os nós que ainda não foram adicionados ao conjunto N' e descobrimos o nó de menor custo ao final da iteração anterior. Este é o nó *x*, com um custo de 1, e, assim, *x* é adicionado ao conjunto N'. A linha 12 do algoritmo de vetor de distâncias (LS) é então rodada para atualizar $D(v)$ para todos os nós *v*, produzindo os resultados mostrados na segunda linha (Etapa 1) da Tabela 4.3. O custo do caminho até *v* não muda. Descobriremos que o custo do caminho até *w* através do nó *x* (que era 5 ao final da inicialização), é 4. Por conseguinte, esse caminho de custo mais baixo é selecionado e o predecessor de *w* ao longo do caminho mais curto a partir de *u* é definido como *x*. De maneira semelhante, o custo até *y* (através de *x*) é computado como 2 e a tabela é atualizada de acordo com isso.
- Na segunda iteração, verificamos que os nós *v* e *y* são os que têm os caminhos de menor custo (2); decidimos o empate arbitrariamente e adicionamos *y* ao conjunto N' de modo que N' agora contém *u*, *x* e *y*. O custo dos nós remanescentes que ainda não estão em N' (isto é, nós *v*, *w* e *z*) são atualizados pela linha 12 do algoritmo LS, produzindo os resultados mostrados na terceira linha da Tabela 4.3.
- E assim por diante...

Quando o algoritmo LS termina, temos, para cada nó, seu predecessor ao longo do caminho de menor custo a partir do nó da fonte. Temos também *o predecessor* para cada um desses predecessores; desse modo, podemos construir o caminho inteiro desde a fonte até todos os destinos. Então, a tabela de repasse em um nó, por exemplo, *u*, pode ser construída a partir dessas informações, armazenando, para cada destino, o nó do salto seguinte no caminho de menor custo de *u* até o destino.

Etapa	N'	D(v), p(v)	D(w), p(w)	D(x), p(x)	D(y), p(y)	D(z), p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					

Tabela 4.3 Execução do algoritmo de estado de enlace na rede da Figura 4.25



a. Roteamento inicial

b. *x, y* detectam melhor caminho até *w* em sentido horário

c. *x, y, z* detectam melhor caminho até *w* em sentido anti-horário

d. *x, y, z,* detectam melhor caminho até *w* em sentido horário

Figura 4.26 Oscilações com roteamento de estado de enlace (LS)

Qual é a complexidade do cálculo desse algoritmo? Isto é, dados n nós (sem contar a fonte), quanto cálculo é preciso efetuar no pior caso para descobrir os caminhos de menor custo entre a fonte e todos os destinos? Na primeira iteração, precisamos pesquisar todos os n nós para determinar o nó w, que não está em N', e que tem o custo mínimo. Na segunda iteração, precisamos verificar $n - 1$ nós para determinar o custo mínimo. Na terceira iteração, $n - 2$ nós. E assim por diante. Em termos gerais, o número total de nós que precisamos pesquisar em todas as iterações é $n(n + 1)/2$, e, assim, dizemos que a complexidade da implementação desse algoritmo de estado de enlace para o pior caso é de ordem n ao quadrado: $O(n^2)$. (Uma implementação mais sofisticada desse algoritmo, que utiliza uma estrutura de dados conhecida como uma pilha, pode descobrir o mínimo na linha 9 em tempo logarítmico e não-linear, reduzindo assim a complexidade.)

Antes de concluirmos nossa discussão sobre o algoritmo LS, vamos considerar uma patologia que pode surgir. A Figura 4.26 mostra uma topologia de rede simples em que os custos dos enlaces são iguais à carga transportada pelo enlace, refletindo, por exemplo, o atraso que seria experimentado. Nesse exemplo, os custos dos enlaces não são simétricos, isto é, $c(u,v)$ é igual a $c(v,u)$ somente se a carga transportada em ambas as direções do enlace (u,v) for a mesma. Nesse exemplo, o nó z origina uma unidade de tráfego destinada a w, o nó x também origina uma unidade de tráfego destinada a w e o nó y injeta uma quantidade de tráfego igual a e também destinada a w. O roteamento inicial é mostrado na Figura 4.26(a) com os custos dos enlaces correspondentes à quantidade de tráfego carregada.

Quando o algoritmo LS é rodado novamente, o nó y determina — baseado nos custos dos enlaces mostrados na Figura 4.26(a) — que o caminho em sentido horário até w tem um custo de 1, ao passo que o caminho em sentido anti-horário até w (que estava sendo usado) tem um custo de $1 + e$. Por conseguinte, o

caminho de menor custo de y até w é agora em sentido horário. De maneira semelhante, x determina que seu novo caminho de menor custo até w é também em sentido horário, resultando nos custos mostrados na Figura 4.26(b). Na próxima vez em que o algoritmo LS é rodado, os nós x, y, e z detectam um caminho de custo zero até w na direção anti-horária, e todos dirigem seu tráfego para as rotas anti-horárias. Na próxima vez em que o algoritmo LS é rodado, os nós x, y, e z então dirigem seu tráfego para as rotas em sentido horário.

O que pode ser feito para evitar essas oscilações (que podem ocorrer com qualquer algoritmo, e não apenas com um algoritmo LS, que use uma métrica de enlace baseada em congestionamento ou em atraso)? Uma solução seria tornar obrigatório que os custos dos enlaces não dependessem da quantidade de tráfego carregada — uma solução inaceitável, já que um dos objetivos do roteamento é evitar enlaces muito congestionados (por exemplo, enlaces com grande atraso). Outra solução seria assegurar que nem todos os roteadores rodassem o algoritmo LS ao mesmo tempo. Esta parece ser uma solução mais razoável, já que é de esperar que, mesmo que os roteadores rodem o algoritmo LS com idêntica periodicidade, o instante de execução do algoritmo não seja o mesmo em cada nó. O interessante é que recentemente os pesquisadores descobriram que os roteadores da Internet podem se auto-sincronizar [Floyd Synchronization, 1994], isto é, mesmo que inicialmente rodem o algoritmo com o mesmo período, mas em diferentes momentos, o instante de execução do algoritmo pode eventualmente se tornar, e permanecer, sincronizado nos roteadores. Um modo de evitar essa auto-sincronização é cada roteador variar aleatoriamente o instante em que envia um anúncio de enlace.

Agora que examinamos o algoritmo de estado de enlace, vamos analisar outro importante algoritmo usado hoje na prática — o algoritmo de roteamento de vetor de distâncias.

4.5.2 O algoritmo de roteamento de vetor de distâncias (DV)

Enquanto o algoritmo LS usa informação global, o algoritmo de vetor de distâncias (***distance-vector* — DV**) é iterativo, assíncrono e distribuído. É distribuído porque cada nó recebe alguma informação de um ou mais vizinhos diretamente ligados a ele, realiza cálculos e, em seguida, distribui os resultados de seus cálculos para seus vizinhos. É iterativo porque esse processo continua até que mais nenhuma informação seja trocada entre vizinhos. (O interessante é que este é um algoritmo finito — não há nenhum sinal de que o cálculo deve parar; ele apenas pára.) O algoritmo é assíncrono porque não requer que todos os nós rodem simultaneamente. Veremos que um algoritmo assíncrono, iterativo, finito e distribuído é muito mais interessante e divertido do que um algoritmo centralizado!

Antes de apresentar o algoritmo DV, é bom discutir uma relação importante que existe entre os custos dos caminhos de menor custo. Seja $d_x(y)$ o custo do caminho de menor custo do nó x ao nó y. Então, os menores custos estão relacionados segundo a famosa equação de Bellman-Ford, a saber:

$$d_x(y) = min_v\{c(x,v) + d_v(y)\}, \tag{4.1}$$

onde o min_v da equação é calculado para todos os vizinhos de x. A equação de Bellman-Ford é bastante intuitiva. Realmente, se após transitarmos de x para v tomarmos o caminho de menor custo de v a y, o custo do caminho será $c(x,v) + d_v(y)$. Como devemos começar viajando até algum vizinho v, o caminho de menor custo de x a y é o mínimo do conjunto dos $c(x,v) + d_v(y)$ calculados para todos os vizinhos v.

Mas, para aqueles que ainda se mostrem céticos quanto à validade da equação, vamos verificá-la para o nó de fonte u e o nó de destino z na Figura 4.25. O nó da fonte u tem três vizinhos: nós v, x e w. Percorrendo vários caminhos no grafo, é fácil ver que $d_v(z) = 5$, $d_x(z) = 3$ e $d_w(z) = 3$. Passando esses valores para a Equação 4.1, juntamente com os custos $c(u,v) = 2$, $c(u,x) = 1$ e $c(u,w) = 5$, temos $d_u(z) = \min\{2 + 5, 5 + 3, 1 + 3\} = 4$, que é, obviamente, verdade e que é, exatamente, o resultado conseguido com o algoritmo de Dijskstra para a mesma rede. Essa verificação rápida deve ajudá-lo a vencer qualquer ceticismo que ainda possa ter.

A equação de Bellman-Ford não é apenas uma curiosidade intelectual. Na verdade, ela tem uma importância prática significativa. Em particular, a solução da equação de Bellman-Ford fornece os registros da tabela de repasse do nó x. Para verificar isso, seja v^* qualquer nó vizinho que represente o mínimo na Equação 4.1. Então, se o nó x quiser enviar um pacote ao nó y pelo caminho de menor custo, deverá, em primeiro lugar, repassar o pacote para o nó v^*. Assim, a tabela de repasse do nó x especificaria o nó v^* como o roteador do próximo salto para o destino final y. Uma outra contribuição importante da equação de Bellman-Ford é que ela sugere a forma da comunicação vizinho para vizinho que ocorrerá no algortimo DV.

A idéia básica é a seguinte. Cada nó começa com $D_x(y)$, uma estimativa do custo do caminho de menor custo entre ele mesmo e o nó y, para todos os nós em N. Seja $\mathbf{D}_x = [D_x(y): y \text{ em } N]$ o vetor de distâncias do nó x, que é o vetor de estimativas de custo de x até todos os outros nós, y, em N. Com o algoritmo DV cada nó x mantém os seguintes dados de roteamento:

- Para cada vizinho v, o custo $c(x,v)$ de x até o vizinho diretamente ligado a ele, v
- O vetor de distâncias do nó x, isto é, $\mathbf{D}_x = [D_x(y): y \text{ em } N]$, contendo a estimativa de x para seus custos até todos os destinos, y, em N
- Os vetores de distâncias de seus vizinhos, isto é, $\mathbf{D}_v = [D_v(y): y \text{ em } N]$ para cada vizinho v de x

No algoritmo distribuído, assíncrono, cada nó envia, a intervalos regulares, uma cópia do seu vetor de distâncias a cada um de seus vizinhos. Quando um nó x recebe um novo vetor de distâncias de qualquer de seus vizinhos v, ele armazena o vetor de distâncias de v e então usa a equação de Bellman-Ford para atualizar seu próprio vetor de distâncias, como segue:

$$D_x(y) = min_v\{c(x,v) + D_v(y)\} \quad \text{para cada nó } y \text{ em } N$$

Se o vetor de distâncias do nó x tiver mudado como resultado dessa etapa de atualização, o nó x então enviará seu vetor de distâncias atualizado para cada um de seus vizinhos que, por sua vez, podem atualizar seus próprios vetores de distâncias. Parece milagre, mas, contanto que todos os nós continuem a trocar seus vetores de distâncias assincronamente, cada estimativa de custo $D_x(y)$ convergirá para $d_x(y)$, que é, na verdade, o custo do caminho de menor custo do nó x ao nó y [Bersekas, 1992]!

Algoritmo vetor de distância (DV)

Para cada nó, x:

```
1  Inicialização:
2     para todos os destinos y em N:
3          Dx(y)  = c(x,y)   /* se y não é um vizinho então c(x,y) = ∞ /*
4     para cada vizinho w
5          Dw(y) = ∞ para todos os destinos y em N
6     para cada vizinho w
7          envia um vetor de distâncias Dx =  [Dx(y): y em N] para w
8
9  loop
10    wait (até que ocorra uma mudança no custo do enlace ao vizinho w ou
11          até a recepção de um vetor de distâncias do vizinho w)
12
13    para cada y em N:
14         Dx(y) = minv{c(x,v) + Dv(y)}
15
16    if Dx(y) mudou para algum destino y
17               envia um vetor de distâncias Dx = [Dx(y): y em N] para todos os vizinhos
```

```
18
19 forever
```

O algoritmo DV mostra como um nó x atualiza sua estimativa do vetor de distâncias quando percebe um mudança de custo em um dos enlaces ligados diretamente a ele ou recebe uma atualização do vetor de distâncias de algum vizinho. Mas, para atualizar sua própria tabela de repasse para um dado destino y, o que o nó x realmente precisa saber não é a distância do caminho mais curto até y, mas qual nó vizinho $v^*(y)$ é o roteador do próximo salto ao longo do caminho mais curto até y. Como era de esperar, o roteador do próximo salto $v^*(y)$ é o vizinho v que representar o mínimo na Linha 14 do algoritmo DV. (Se houver vários vizinhos v que representem o mínimo, então $v^*(y)$ pode ser qualquer um dos vizinhos minimizadores.) Assim, nas Linhas 13-14, para cada destino y, o nó x também determina $v^*(y)$ e atualiza sua tabela de repasse para o destino y.

Lembre-se de que o algoritmo LS é um algoritmo global no sentido de que requer que cada nó obtenha, em primeiro lugar, um mapa completo da rede antes de rodar o algoritmo de Dijkstra. O algoritmo DV é *descentralizado* e não usa essa informação global. Realmente, a única informação que um nó terá são os custos dos enlaces até os vizinhos diretamente ligados a ele e as informações que recebe desses vizinhos. Cada nó espera uma atualização de qualquer vizinho (Linhas 10–11), calcula seu novo vetor de distâncias ao receber uma atualização (Linha 14) e distribui seu novo vetor de distâncias a seus vizinhos (Linhas 16–17). O algoritmo DV é utilizado em muitos protocolos de roteamento na prática, entre eles o RIP e o BGP da Internet, o ISO IDRP, o IPX da Novell, e o ARPAnet original.

A Figura 4.27 ilustra a operação do algoritmo DV para a rede simples de três nós mostrada na parte superior da figura. A operação do algoritmo é ilustrada de um modo síncrono, no qual todos os nós recebem vetores de distâncias simultaneamente de seus vizinhos, calculam seus novos vetores de distâncias e informam a seus vizinhos se esses vetores mudaram. Após estudar esse exemplo, você deve se convencer de que o algoritmo também opera corretamente em modo assíncrono, com cálculos de nós e atualizações de geração/recepção ocorrendo a qualquer instante.

A coluna mais à esquerda na figura mostra três **tabelas de roteamento** iniciais para cada um dos três nós. Por exemplo, a tabela no canto superior à esquerda é a tabela de roteamento inicial do nó x. Dentro de uma tabela de roteamento específica, cada linha é um vetor de distâncias — especificamente, a tabela de roteamento de cada nó inclui seu próprio vetor de distâncias e os vetores de cada um de seus vizinhos. Assim, a primeira linha da tabela de roteamento inicial do nó x é $\mathbf{D}_x = [D_x(x), D_x(y), D_x(z)] = [0, 2, 7]$. A segunda linha e a terceira linha nessa tabela são os vetores de distâncias recebidos mais recentemente dos nós y e z, respectivamente. Como na inicialização o nó x não recebeu nada do nó y ou z, os registros da segunda linha e da terceira linha estão definidos como infinito.

Após a inicialização, cada nó envia seu vetor de distâncias a cada um de seus dois vizinhos. Isso é ilustrado na Figura 4.27 pelas setas que vão da primeira coluna de tabelas até a segunda coluna de tabelas. Por exemplo, o nó x envia seu vetor de distâncias $\mathbf{D}_x = [0, 2, 7]$ a ambos os nós y e z. Após receber as atualizações, cada nó recalcula seu próprio vetor de distâncias. Por exemplo, o nó x calcula

$$D_x(x) = 0$$
$$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\} = \min\{2 + 0, 7 + 1\} = 2$$
$$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\} = \min\{2 + 1, 7 + 0\} = 3$$

Por conseguinte, a segunda coluna mostra, para cada nó, o novo vetor de distâncias do nó, juntamente com os vetores de distâncias que acabou de receber de seus vizinhos. Observe, por exemplo, que a estimativa do nó x para o menor custo até o nó z, $D_x(z)$, mudou de 7 para 3. Note também que, para ambos os nós y e z, o nó y representa os mínimos correspondentes. Assim, nesse estágio do algoritmo, os roteadores do próximo salto são $v^*(y) = y$ e $v^*(z) = y$.

Depois que recalculam seus vetores de distâncias, os nós enviam novamente seus vetores de distâncias recalculados a seus vizinhos (se houver uma mudança). Isso é ilustrado na Figura 4.27 pelas setas que vão da segunda coluna de tabelas até a terceira coluna de tabelas. Note que somente os nós x e z enviam atuali-

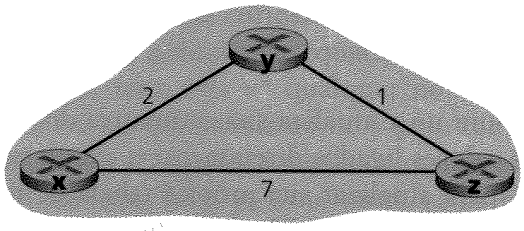


Tabela do nó x

De	Custo até x	Custo até y	Custo até z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

De	Custo até x	Custo até y	Custo até z
x	0	2	3
y	2	0	1
z	7	1	0

De	Custo até x	Custo até y	Custo até z
x	0	2	3
y	2	0	1
z	3	1	0

Tabela do nó y

De	Custo até x	Custo até y	Custo até z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

De	Custo até x	Custo até y	Custo até z
x	0	2	7
y	2	0	1
z	7	1	0

De	Custo até x	Custo até y	Custo até z
x	0	2	3
y	2	0	1
z	3	1	0

Tabela do nó z

De	Custo até x	Custo até y	Custo até z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

De	Custo até x	Custo até y	Custo até z
x	0	2	7
y	2	0	1
z	3	1	0

De	Custo até x	Custo até y	Custo até z
x	0	2	3
y	2	0	1
z	3	1	0

Tempo

Figura 4.27 Algoritmo de vetor de distâncias (DV)

zações: o vetor de distâncias do nó *y* não mudou, então o nó *y* não envia uma atualização. Após receber as atualizações, os nós então recalculam seus vetores de distâncias e atualizam suas tabelas de roteamento, que são mostradas na terceira coluna.

O processo de receber vetores de distâncias atualizados de vizinhos, recalcular os registros de tabelas de roteamento e informar aos vizinhos os custos modificados do caminho de menor custo até o destino continua até que mais nenhuma mensagem de atualização seja enviada. Nesse ponto, já que mais nenhuma mensagem de atualização é enviada, não ocorrerá mais nenhum cálculo de tabela de roteamento e o algoritmo entra em estado de inatividade; isto é, todos os nós estarão rodando a espera nas Linhas 10-11 do

algoritmo DV. O algoritmo permanece no estado de inatividade até que o custo de um enlace mude, como discutiremos a seguir.

Algoritmo de vetor de distâncias: mudanças no custo do enlace e falha no enlace

Quando um nó que está rodando o algoritmo DV detecta uma mudança no custo do enlace dele mesmo até um vizinho (Linhas 10-11), ele atualiza seu vetor de distâncias (Linhas 13-14) e, se houver uma modificação no custo do caminho de menor custo, informa a seus vizinhos (Linhas 16-17) seu novo vetor de distâncias. A Figura 4.28(a) ilustra um cenário em que o custo do enlace de y a x muda de 4 para 1. Destacamos aqui somente os registros na tabela de distâncias de y e z até o destino x.

O algoritmo DV causa a ocorrência da seguinte seqüência de eventos:

- No tempo t_0, y detecta a mudança no custo do enlace (o custo mudou de 4 para 1), atualiza seu vetor de distâncias e informa essa mudança a seus vizinhos, já que o vetor mudou.
- No tempo t_1, z recebe a atualização de y e atualiza sua própria tabela. Calcula um novo menor custo para x (cujo custo diminuiu de 5 para 2), e envia seu novo vetor de distâncias a seus vizinhos.
- No tempo t_2, y recebe a atualização de z e atualiza sua tabela de distâncias. Os menores custos de y não mudaram e, por conseguinte, y não envia nenhuma mensagem a z. O algoritmo entra em estado de inatividade.

Assim, apenas duas iterações são necessárias para o algoritmo DV alcançar o estado de inatividade. A boa notícia sobre a redução do custo entre x e y se propagou rapidamente pela rede.

Agora vamos considerar o que pode acontecer quando o custo de um enlace *aumenta*. Suponha que o custo do enlace entre x e y aumente de 4 para 60, como mostra a Figura 4.28(b).

1. Antes da mudança do custo do enlace, $D_y(x) = 4$, $D_y(z) = 1$, $D_z(y) = 1$, e $D_z(x) = 5$. No tempo t_0, y detecta uma mudança no custo do enlace (o custo mudou de 4 para 60). y calcula seu novo caminho de custo mínimo até x, de modo a ter um custo de

$$D_y(x) = \min\{c(y,x) + D_x(x), c(y,z) + D_z(x)\} = \min\{60 + 0,1 + 5\} = 6$$

   É claro que, com nossa visão global da rede, podemos ver que esse novo custo via z está errado. Mas as únicas informações que o nó y tem é que seu custo direto até x é 60 e que z disse a y que z pode chegar a x com um custo de 5. Assim, para chegar a x, y teria de fazer a rota através de z, com a expectativa de que z será capaz de chegar a x com um custo de 5. A partir de t_1, temos um **loop de roteamento** — para poder chegar a x, y faz a rota através de z, que por sua vez faz a rota através de y. Um loop de roteamento é como um buraco negro — um pacote destinado a x que chegar a y ou a z a partir do momento t_1 vai ricochetear entre um e outro desses dois nós para sempre (ou até que as tabelas de repasse sejam mudadas).
2. Tão logo o nó y tenha calculado um novo custo mínimo até x, ele informará a z esse novo vetor de distâncias no tempo t_1.

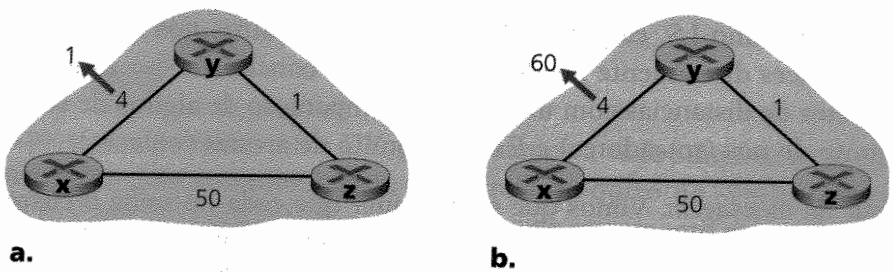


Figura 4.28 Mudanças no custo do enlace

3. Algum tempo depois de t_1, z recebe o novo vetor de distâncias de y, que indica que o custo mínimo de y até x é 6. z sabe que pode chegar até y com um custo de 1 e, por conseguinte, calcula um novo menor custo até x, $D_2(x) = \min\{50 + 0{,}1 + 6\} = 7$. Uma vez que o custo mínimo de z até x aumentou, z informa a y o seu novo vetor de distâncias em t_2.
4. De maneira semelhante, após receber o novo vetor de distâncias de z, y determina $D_y(x) = 8$ e envia a z seu vetor de distâncias. Então z determina $D_z(x) = 9$ e envia a y seu vetor de distâncias e assim por diante.

Por quanto tempo esse processo continua? Pode ter certeza de que o loop persistirá por 44 iterações (trocas de mensagens entre y e z) até que z possa, eventualmente, calcular que o custo de seu caminho via y é maior do que 50. Nesse ponto, z (finalmente!) determinará que seu caminho de menor custo até x é via sua conexão direta com x. y então fará a rota até x via z. O resultado das más notícias sobre o aumento do custo do enlace na verdade viajou devagar! O que teria acontecido se o custo do enlace $c(y, x)$ tivesse mudado de 4 para 10.000 e o custo $c(z, x)$ fosse 9999? Por causa de cenários como esses, o problema que acabamos de examinar é, às vezes, denominado problema de contagem ao infinito.

Algoritmo de vetor de distâncias: adição de reversão envenenada

O cenário específico de looping que acabamos de descrever pode ser evitado usando uma técnica denominada reversão envenenada (*poisoned reverse*). A idéia é simples — se a rota de z para chegar a x passa por y, então z deve anunciar a y que sua distância a x é infinita, isto é, z anunciará a y que $D_z(x) = \infty$ (mesmo que z saiba que, na verdade, $D_z(x) = 5$). z continuará contando essa mentirinha inocente a y enquanto a rota de z a x estiver passando por y. Enquanto y acreditar que z não tem nenhum caminho até x, y jamais tentará a rota até x por z, contanto que a rota de z a x continue a passar por y (e ele minta sobre isso).

Agora vamos ver como a reversão envenenada resolve o problema específico do looping que encontramos anteriormente na Figura 4.28(b). Como resultado da reversão envenenada, a tabela de distâncias de y indica que $Dz(x) = \infty$. Quando o custo do enlace (x, y) muda de 4 para 60 no tempo t_0, y atualiza sua tabela e continua a estabelecer rotas diretamente para x, embora com um custo mais alto do que 60, e informa z o seu novo custo até x, isto é, $D_y(x) = 60$. Após receber a atualização em t_1, z imediatamente desloca sua rota para x, para que seja via o enlace direto (z, x) a um custo de 50. Como este é um novo menor custo até x, e uma vez que o caminho não passa mais por y, z agora informa a y que $Dz(x) = 50$ em t_2. Após receber a atualização de z, y atualiza sua tabela de distâncias com $Dz(x) = 51$. E, também, uma vez que z está agora no caminho de menor custo até x, y envenena o caminho inverso de z a x, informando a z, no tempo t_3, que $D_y(x) = \infty$ (mesmo que y saiba que, na verdade, $D_y(x) = 51$).

A reversão envenenada resolve o problema geral da contagem até o infinito? Não resolve. É bom que você se convença de que loops que envolvem três ou mais nós (e não apenas dois nós imediatamente vizinhos) não serão detectados pela técnica da reversão envenenada.

Uma comparação entre os algoritmos de roteamento de estado de enlace e de vetor de distâncias

Os algoritmos DV e LS adotam abordagens complementares em relação ao cálculo do roteamento. No algoritmo DV, cada nó fala *somente* com os vizinhos diretamente conectados a ele, mas informa a esses vizinhos as estimativas de menor custo entre ele mesmo e *todos* os outros nós da rede (isto é, todos os que ele sabe que existem). No algoritmo LS, cada nó fala com *todos* os outros nós (via *broadcast*), mas informa *somente* os custos dos enlaces diretamente ligados a ele. Vamos concluir nosso estudo sobre algoritmos de estado de enlace e de vetor de distâncias com uma rápida comparação de alguns de seus atributos. Lembre-se de que N é o conjunto de nós (roteadores) e E é o conjunto de arestas (enlaces).

Complexidade da mensagem. Vimos que o LS requer que cada nó saiba o custo de cada enlace da rede. Isso exige que sejam enviadas $O(|N|\,|E|)$. E, também, sempre que o custo de um enlace muda, o novo custo deve ser enviado a todos os nós. O algoritmo DV requer troca de mensagens entre vizinhos diretamente conectados a cada iteração. Já vimos que o tempo necessário para que o algo-

ritmo convirja pode depender de muitos fatores. Quando o custo do enlace muda, o algoritmo DV propaga os resultados do custo modificado do enlace somente se o novo custo resultar em mudança no caminho de menor custo para um dos nós ligado ao enlace.

- *Velocidade de convergência.* Já vimos que nossa implementação de LS é um algoritmo $O(|N|^2)$ que requer $O(N| |E|)$ mensagens. O algoritmo DV pode convergir lentamente e pode ter loops de roteamento enquanto estiver convergindo. O algoritmo DV também tem o problema da contagem até o infinito.
- *Robustez.* O que pode acontecer se um roteador falhar, se comportar mal ou for sabotado? Sob o LS, um roteador poderia transmitir um custo incorreto para um de seus enlaces diretos (mas não para outros). Um nó poderia também corromper ou descartar quaisquer pacotes recebidos como parte de uma transmissão de estado de enlace. Mas um nó LS está calculando somente suas próprias tabelas de roteamento; os outros nós estão realizando cálculos semelhantes para si próprios. Isso significa que, sob o LS, os cálculos de rota são, de certa forma, isolados, fornecendo um grau de robustez. Sob o DV, um nó pode anunciar incorretamente caminhos de menor custo para qualquer destino, ou para todos os destinos. (Na verdade, em 1997, um roteador que estava funcionando mal em um pequeno ISP forneceu aos roteadores nacionais de backbone tabelas de roteamento errôneas. Isso fez com que outros roteadores inundassem de tráfego o roteador que estava funcionando mal. Com isso, grandes porções da Internet ficaram desconectadas durante muitas horas [Neumann, 1997].) De modo geral, notamos que, a cada iteração, um cálculo de nó em DV é passado adiante a seu vizinho e, em seguida, indiretamente ao vizinho de seu vizinho na iteração seguinte. Nesse sentido, sob o DV, um cálculo incorreto do nó pode ser difundido pela rede inteira.

No final, nenhum algoritmo *ganha* do outro. Realmente, ambos são usados na Internet.

Outros algoritmos de roteamento

Os algoritmos LS e DV que estudamos não somente são utilizados em grande escala na prática, mas também são, essencialmente, *os únicos* algoritmos utilizados hoje na Internet.

Não obstante, muitos algoritmos de roteamento foram propostos por pesquisadores nos últimos 30 anos, abrangendo desde o extremamente simples até o muito sofisticado e complexo. Há uma grande classe de algoritmos de roteamento cuja base é considerar o tráfego como fluxos entre fontes e destinos em uma rede. Nessa abordagem, o problema do roteamento pode ser formulado em termos matemáticos como um problema de otimização restrita, conhecido como problema de fluxo da rede [Bertsekas, 1991]. Ainda um outro conjunto de algoritmos de roteamento que mencionamos aqui são os derivados do mundo da telefonia. Esses **algoritmos de roteamento de comutação de circuitos** são de interesse para redes de comutação de circuitos nos casos em que devem ser reservados recursos (por exemplo, buffers ou uma fração da largura de banda do enlace) por enlace para cada conexão roteada pelo enlace. Embora a formulação do problema do roteamento talvez pareça bem diferente da formulação do problema do roteamento de menor custo que vimos neste capítulo, existem muitas semelhanças, ao menos no que se refere ao algoritmo de busca de caminhos (algoritmo de roteamento). Veja [Ash, 1998; Ross, 1995; Girard, 1990] para uma discussão detalhada dessa área de pesquisa.

4.5.3 Roteamento hierárquico

Quando estudamos os algoritmos LS e DV, consideramos a rede simplesmente como uma coleção de roteadores interconectados. Um roteador não se distinguia de outro no sentido de que todos rodavam o mesmo algoritmo de roteamento para calcular os caminhos de roteamento pela rede inteira. Na prática, esse modelo e sua visão de um conjunto homogêneo de roteadores, todos rodando o mesmo algoritmo de roteamento, é um tanto simplista por pelo menos duas razões importantes:

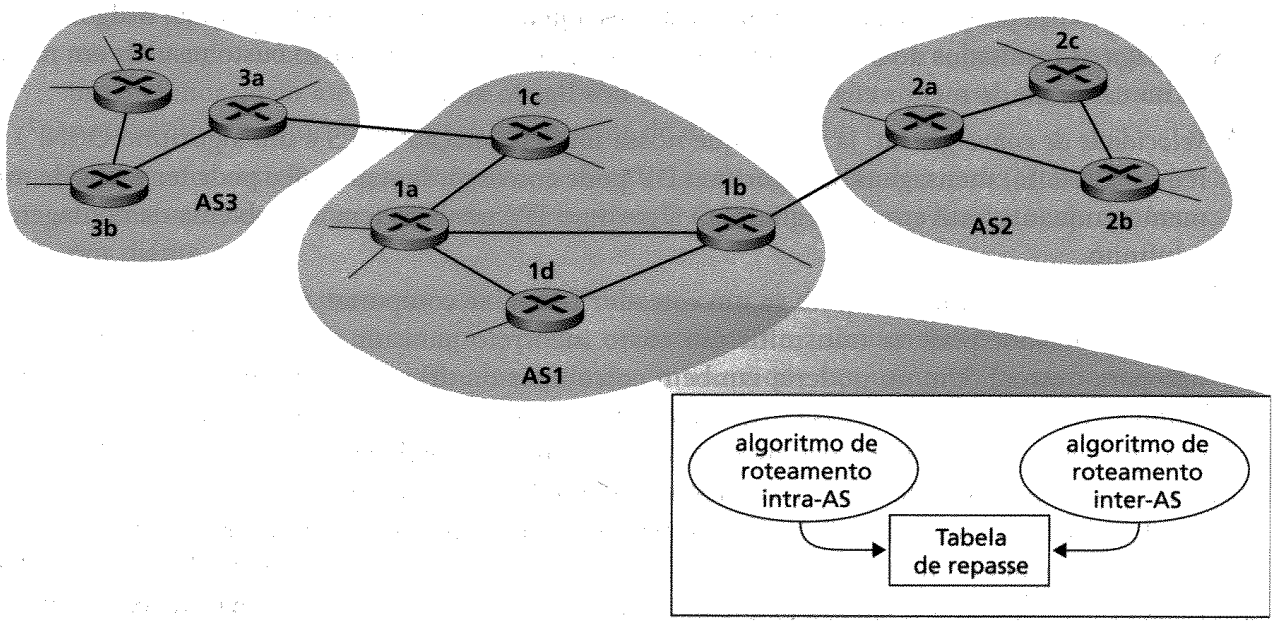


Figura 4.29 Um exemplo de sistemas autônomos interconectados

Escala. À medida que aumenta o número de roteadores, a sobrecarga relativa ao cálculo, ao armazenamento e à comunicação de informações para a tabela de roteamento (por exemplo, atualizações de estado de enlace ou de mudança no caminho de menor custo) se torna proibitiva. A Internet pública de hoje consiste em centenas de milhões de hospedeiros. Armazenar informações de roteamento para cada um desses hospedeiros evidentemente exigiria quantidades enormes de memória. Com a sobrecarga exigida para transmitir atualizações do estado de enlace entre todos os roteadores da Internet não sobraria nenhuma largura de banda para enviar pacotes de dados! Um algoritmo DV que fizesse iterações entre esse número tão grande de roteadores seguramente jamais convergiria! Fica claro que algo deve ser feito para reduzir a complexidade do cálculo de rotas em redes tão grandes como a Internet pública.

Autonomia administrativa. Embora os pesquisadores tendam a ignorar questões como o desejo das empresas de controlar seus roteadores como bem entendem (por exemplo, rodar qualquer algoritmo de roteamento que escolherem) ou ocultar do público externo aspectos da organização interna das redes, essas considerações são importantes. Idealmente, uma organização deveria poder executar e administrar sua rede como bem entendesse, mantendo a capacidade de conectar sua rede a outras redes externas.

Esses problemas podem ser resolvidos agrupando roteadores em **sistemas autônomos** (*autonomous systems* — ASs), com cada AS consistindo em um grupo de roteadores sob o mesmo controle administrativo (isto é, operados pelo mesmo ISP ou pertencentes a uma mesma rede corporativa). Todos os roteadores dentro do mesmo AS rodam o mesmo algoritmo de roteamento (por exemplo, algoritmo LS ou DV) e dispõem das informações sobre cada um dos outros — exatamente como foi o caso do modelo idealizado da seção anterior. O algoritmo de roteamento que roda dentro de um AS é denominado um **protocolo de roteamento intra-sistema autônomo**. É claro que será necessário conectar os ASs entre si e, assim, um ou mais dos roteadores em um AS terá a tarefa adicional de ficar responsável por transmitir pacotes a destinos externos ao AS — esses roteadores são denominados **roteadores de borda** (*gateway routers*).

A Figura 4.29 ilustra um exemplo simples com três ASs: AS1, AS2 E AS3. Na figura, as linhas escuras representam conexões diretas de enlaces entre pares e roteadores. As linhas mais finas e interrompidas que saem dos roteadores representam sub-redes conectadas diretamente a eles. O AS1 tem quatro roteadores, 1.a, 1.b, 1.c e 1.d, e cada qual roda o protocolo de roteamento utilizado dentro do AS1. Assim, cada um des-

ses quatro roteadores sabe como transmitir pacotes ao longo do caminho ótimo para qualquer destino dentro de AS1. De maneira semelhante, cada um dos sistemas autônomos AS2 e AS3 tem três roteadores. Note que os protocolos de roteamento intra-AS que rodam em AS1, AS2 e AS3 não precisam ser os mesmos. Note também que os roteadores 1b, 1c, 2a e 3a são roteadores de borda.

Agora já deve estar claro como os roteadores em um AS determinam caminhos de roteamento para pares fonte-destino internos ao sistema. Mas ainda há uma peça faltando nesse quebra-cabeça de roteamento fim-a-fim. Como um roteador que está dentro de algum AS sabe como rotear um pacote até um destino que está fora do AS? Essa pergunta é fácil de responder se o AS tiver somente um roteador de borda que se conecta com somente um outro AS. Nesse caso, como o algoritmo de roteamento intra-AS do sistema autônomo determinou o caminho de menor custo entre cada roteador interno e o roteador de borda, cada roteador interno sabe como deve enviar o pacote. Ao receber o pacote, o roteador de borda o repassa para o único enlace que leva ao exterior do AS. Então, o AS que está na outra extremidade do enlace assume a responsabilidade de rotear o pacote até seu destino final. Como exemplo, suponha que o roteador 2b da Figura 4.29 receba um pacote cujo destino está fora do AS2. O roteador 2b então transmite o pacote ao roteador 2a ou 2c, como especificado pela tabela de repasse do roteador 2b, que foi configurada pelo protocolo de roteamento intra-AS de AS2. O pacote eventualmente chegará ao roteador de borda 2a, que o retransmitirá ao roteador 1b. Tão logo o pacote tenha saído de 2a, termina o trabalho de AS2 com referência a esse pacote.

Portanto, o problema é fácil quando o AS de origem tem somente um enlace que leva para fora do AS. Mas, e se o AS de origem tiver dois ou mais enlaces (passando por um ou mais roteadores de borda) que levam para fora do AS? Então, o problema de saber para onde repassar o pacote torna-se significativamente mais desafiador. Por exemplo, considere um roteador em AS1 e suponha que ele recebe um pacote cujo destino está fora do AS. É claro que o roteador deveria repassar o pacote para um de seus dois roteadores de borda, 1b ou 1c, mas para qual deles? Para resolver esse problema, AS1 (1) precisa saber quais destinos podem ser alcançados via o AS2 e quais podem ser alcançados via o AS3 e (2) precisa propagar essa informação para todos os roteadores dentro de AS1, de modo que cada roteador possa configurar sua tabela de repasse para manipular destinos externos ao AS. Essas duas tarefas — obter informações sobre as condições de alcance de ASs vizinhos e propagar essas informações a todos os outros roteadores internos a AS — são gerenciadas pelo **protocolo de roteamento inter-AS.** Visto que o protocolo de roteamento inter-AS envolve comunicação entre dois ASs, esses dois ASs comunicantes devem rodar o mesmo protocolo de roteamento inter-AS. Realmente, na Internet, todos os ASs rodam o mesmo protocolo de roteamento inter-AS, denominado BGP4, que discutiremos na próxima seção. Como ilustrado na Figura 4.29, cada roteador recebe informações de um protocolo de roteamento intra-AS e de um protocolo de roteamento inter-AS, e usa as informações de ambos os protocolos para configurar sua tabela de repasse.

Como exemplo, considere uma sub-rede *x* (identificada por seu endereço "ciderizado") e suponha que AS1 sabe, por meio do protocolo de roteamento inter-AS, que a sub-rede *x* pode ser alcançada de AS3, mas *não* pode ser alcançada de AS2. Então, AS1 propaga essa informação a todos os seus roteadores. Quando o roteador 1d fica sabendo que a sub-rede *x* pode ser alcançada de AS3 e, por conseguinte, do roteador de borda 1c, determina, com base na informação fornecida pelo protocolo de roteamento intra-AS, a interface de roteador que está no caminho de menor custo entre o roteador 1d e o roteador de borda 1c. Seja *I* essa interface. O roteador 1d então pode colocar o registro (*x*, *I*), em sua tabela de repasse. (Esse exemplo, e outros apresentados nesta seção, passam as idéias gerais, mas são simplificações do que realmente acontece na Internet. Na seção seguinte daremos uma descrição mais detalhada, se bem que mais complicada, quando discutirmos o BGP.)

Continuando com nosso exemplo anterior, suponha agora que AS2 e AS3 estão conectados com outros ASs, que não aparecem no diagrama. Suponha também que AS1 fica sabendo, pelo protocolo de roteamento inter-AS, que a sub-rede *x* pode ser alcançada de AS2, via o roteador de borda 1b, e também de AS3, via o roteador de borda 1c. Então, AS1 propagaria essa informação a todos os seus roteadores, incluindo o roteador 1d. Para configurar sua tabela de repasse, o roteador 1d teria de determinar para qual roteador de borda, 1b ou 1c, deve dirigir pacotes destinados à sub-rede *x*. Uma abordagem, que é freqüentemente empregada

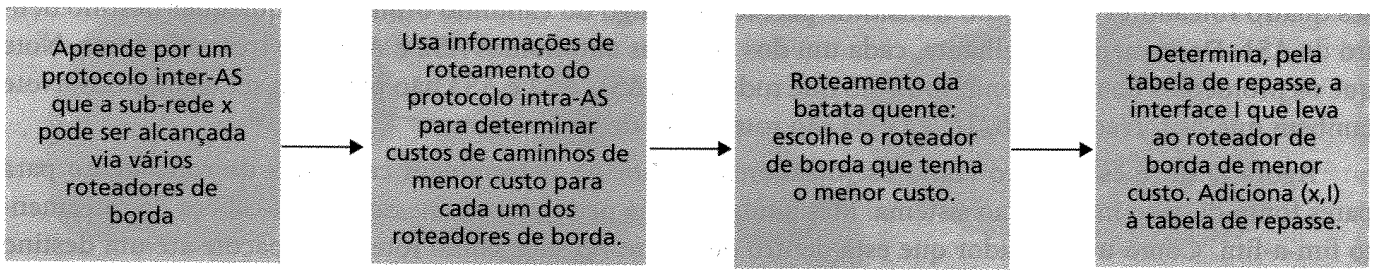


Figura 4.30 Etapas da adição de um destino fora do AS à tabela de repasse de um roteador

na prática, é utilizar o **roteamento da batata quente**. Com esse roteamento, o AS se livra do pacote (a batata quente) o mais rapidamente possível (mais exatamente, com o menor custo possível). Isso é feito obrigando um roteador a enviar o pacote ao roteador de borda que tiver o menor custo roteador-roteador de borda entre todos os roteadores de borda que têm um caminho para o destino. No contexto do exemplo que estamos examinando, o roteamento da batata quente, rodando em 1d, usaria informação recebida do protocolo de roteamento intra-AS para determinar os custos de caminho até 1b e 1c, e então escolheria o caminho de menor custo. Uma vez escolhido este caminho, o roteador 1d adiciona em sua tabela de repasse um registro para a sub-rede *x*. A Figura 4.30 resume as ações executadas no roteador 1d para adicionar à tabela de repasse o novo registro para *x*.

Quando um AS fica sabendo de um destino por meio de um AS vizinho, ele pode anunciar essa informação de roteamento a alguns de seus outros ASs vizinhos. Por exemplo, suponha que AS1 fica sabendo, por AS2, que a sub-rede *x* pode ser alcançada via AS2. Então, AS1 poderia dizer a AS3 que *x* pode ser alcançada via AS1. Desse modo, se AS3 precisar rotear um pacote destinado a *x*, repassaria o pacote a AS1 que, por sua vez, repassaria o pacote para AS2. Como veremos quando discutirmos BGP, um AS tem bastante flexibilidade para decidir quais destinos anuncia a seus ASs vizinhos. Esta é uma decisão *política* que normalmente depende mais de questões econômicas do que técnicas.

Lembre-se de que dissemos na Seção 1.5 que a Internet consiste em uma hierarquia de ISPs interconectados. Então, qual é a relação entre ISPs e ASs? Você talvez pense que os roteadores em um ISP e os enlaces que os interconectam constituem um único AS. Embora esse seja freqüentemente o caso, muitos ISPs dividem sua rede em vários ASs. Por exemplo, alguns ISPs de nível 1 utilizam um AS para toda a sua rede; outros subdividem sua rede em dezenas de ASs interconectados.

Em resumo, os problemas de escala e de autoridade administrativa são resolvidos pela definição de sistemas autônomos. Dentro de um AS, todos os roteadores rodam o mesmo protocolo de roteamento intra-sistema autônomo. Entre eles, os ASs rodam o mesmo protocolo de roteamento inter-AS. O problema de escala é resolvido porque um roteador intra-AS precisa saber apenas dos outros roteadores dentro do AS. O problema de autoridade administrativa é resolvido, já que uma organização pode rodar o protocolo de roteamento intra-AS que quiser; todavia, cada par de ASs conectados precisa rodar o mesmo protocolo de roteamento inter-AS para trocar informações de alcançabilidade.

Na seção seguinte, examinaremos dois protocolos de roteamento intra-AS (RIP e OSPF) e o protocolo inter-AS (BGP), que são usados na Internet de hoje. Esses estudos de casos dão um bom arremate ao nosso estudo de roteamento hierárquico.

4.6 Roteamento na Internet

Agora que já estudamos endereçamento na Internet e o protocolo IP, vamos voltar nossa atenção aos protocolos de roteamento da Internet. A tarefa desses protocolos é determinar o caminho tomado por um datagrama entre a fonte e o destino. Veremos que os protocolos de roteamento da Internet incorporam muitos dos princípios que aprendemos anteriormente neste capítulo. As abordagens de estado de enlace e de

vetor de distâncias estudadas nas seções 4.5.1 e 4.5.2 e a idéia de um sistema autônomo considerada na Seção 4.5.3 são fundamentais para o modo como o roteamento é feito na Internet de hoje.

Lembre-se de que na Seção 4.5.3 vimos que um sistema autônomo (AS) é um conjunto de roteadores que estão sob o mesmo controle administrativo e técnico e que rodam, todos, o mesmo protocolo de roteamento entre eles. Cada AS, por sua vez, normalmente contém várias sub-redes (aqui, usamos o termo sub-rede no sentido preciso de endereçamento, como na Seção 4.4.2).

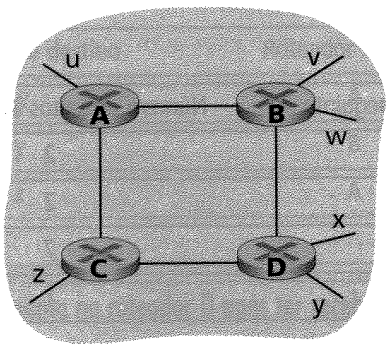
4.6.1 Roteamento intra-AS na Internet: RIP

Um protocolo de roteamento intra-AS é usado para determinar como é rodado o roteamento dentro de um sistema autônomo (AS). Protocolos de roteamento intra-AS são também conhecidos como **protocolos de roteadores internos** (IGP — *Interior Gateway Protocols*). Historicamente, dois protocolos de roteamento têm sido usados extensivamente para roteamento dentro de um sistema autônomo na Internet: o **RIP** (*Routing Information Protocol* — Protocolo de Informação de Roteamento) e o **OSPF** (*open shortest path first*). Um protocolo muito relacionado com o OSPF é o protocolo **IS-IS** [RFC 1142, Perlman, 1999]. Em primeiro lugar, discutiremos o RIP e, em seguida, consideraremos o OSPF.

O RIP foi um dos primeiros protocolos de roteamento intra-AS da Internet, e seu uso é ainda amplamente disseminado. Sua origem e seu nome podem ser traçados até a arquitetura XNS (Xerox Network Systems). A ampla disponibilização do RIP se deveu em grande parte à sua inclusão, em 1982, na versão do UNIX do Berkeley Software Distribution (BSD), que suportava TCP/IP. A versão 1 do RIP está definida no RFC 1058 e a versão 2, compatível com a versão 1, no RFC 1723.

O RIP é um protocolo de vetor de distâncias que funciona de um modo muito parecido com o protocolo DV idealizado que examinamos na Seção 4.5.2. A versão do RIP especificada no RFC 1058 usa contagem de saltos como métrica de custo, isto é, cada enlace tem um custo 1. Por simplicidade, no algoritmo DV da Seção 4.5.2 os custos foram definidos entre pares de roteadores. No RIP (e também no OSPF), na realidade, os custos são definidos desde um roteador de origem até uma sub-rede de destino. O RIP usa o termo *salto*, que é o número de sub-redes percorridas ao longo do caminho mais curto entre o roteador de origem e uma sub-rede de destino, incluindo a sub-rede de destino. A Figura 4.31 ilustra um AS com seis sub-redes folha. A tabela da figura indica o número de saltos desde o roteador de origem A até todas as sub-redes folha.

O custo máximo de um caminho é limitado a 15, limitando assim o uso do RIP a sistemas autônomos que têm menos de 15 saltos de diâmetro. Lembre-se de que em protocolos DV, roteadores vizinhos trocam vetores de distância entre si. O vetor de distâncias para qualquer roteador é a estimativa corrente das distâncias dos caminhos de menor custo entre aquele roteador e as sub-redes no AS. No RIP, tabelas de roteamento são trocadas entre vizinhos a cada 30 segundos aproximadamente, usando uma mensagem de **resposta RIP**. A mensagem de resposta enviada por um roteador ou um hospedeiro contém uma lista de até 25 sub-redes de destino dentro do AS, bem como as distâncias entre o remetente e cada uma dessas sub-redes. Mensagens de resposta também são conhecidas como **anúncios RIP**.



Destino	Saltos
u	1
v	2
w	2
x	3
y	3
z	2

Figura 4.31 Número de saltos do roteador de origem, A, até várias sub-redes

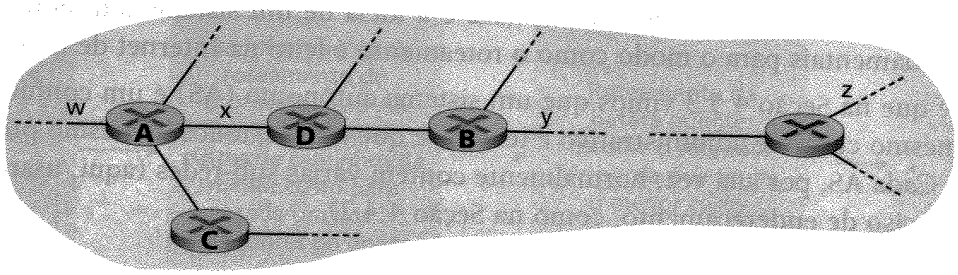


Figura 4.32 Uma parte de um sistema autônomo

Vamos examinar um exemplo simples de como funcionam os anúncios RIP. Considere a parte de um AS mostrada na Figura 4.32. Nessa figura, as linhas que conectam os roteadores representam sub-redes. Somente os roteadores (*A*, *B*, *C* e *D*) e as redes (*w*, *x*, *y*, *z*) selecionados são rotulados. As linhas tracejadas indicam que o AS continua; portanto, esse sistema autônomo tem muito mais roteadores e enlaces do que os mostrados na figura.

Cada roteador mantém uma tabela RIP denominada **tabela de roteamento.** A tabela de roteamento de um roteador inclui o vetor de distâncias e a tabela de repasse desse roteador. A Figura 4.33 mostra a tabela de roteamento do roteador *D*. Note que a tabela de roteamento tem três colunas. A primeira coluna é para a sub-rede de destino, a segunda coluna indica a identidade do roteador seguinte ao longo do caminho mais curto até a sub-rede de destino e a terceira coluna indica o número de saltos (isto é, o número de sub-redes que têm de ser atravessadas, incluindo a rede de destino) para chegar à sub-rede de destino ao longo do caminho mais curto. Para este exemplo, a tabela mostra que, para enviar um datagrama do roteador *D* até a sub-rede de destino *w*, o datagrama deve primeiramente ser enviado ao roteador vizinho *A*; a tabela também mostra que a sub-rede de destino *w* está a dois saltos de distância ao longo do caminho mais curto. De modo semelhante, a tabela indica que a sub-rede *z* está a sete saltos de distância via o roteador *B*. Em princípio, uma tabela de roteamento terá apenas uma linha para cada sub-rede no AS, embora a versão 2 do RIP permita a agregação de registros de sub-redes usando técnicas de agregação de rotas semelhantes àquelas que examinamos na Seção 4.4. A tabela na Figura 4.33 e as tabelas subseqüentes estão apenas parcialmente completas.

Suponha agora que 30 segundos mais tarde o roteador *D* receba do roteador *A* o anúncio mostrado na Figura 4.34. Note que esse anúncio nada mais é do que informações da tabela de roteamento do roteador *A*! Essa informação indica, em particular, que a sub-rede *z* está a apenas quatro saltos do roteador *A*. Ao receber esse anúncio, o roteador *D* o funde (Figura 4.34) com a tabela de roteamento antiga (Figura 4.33). Em particular, o roteador *D* fica sabendo que agora há um novo caminho através do roteador *A* até a sub-rede *z* que é mais curto do que o caminho pelo roteador *B*. Assim, o roteador *D* atualiza sua tabela de roteamento para levar em conta o mais curto dos caminhos mais curtos, como mostra a Figura 4.35. Você poderia perguntar como o caminho mais curto até a sub-rede *z* se tornou mais curto ainda? Possivelmente, o algoritmo de vetor de distâncias descentralizado ainda estava em processo de convergência (veja a Seção 4.5.2) ou, talvez, novos enlaces e/ou roteadores tenham sido adicionados ao AS, mudando assim os caminhos mais curtos dentro dele.

Sub-rede de destino	Roteador seguinte	Número de saltos até o destino
w	A	2
y	B	2
z	B	7
x	–	1
....	....	....

Figura 4.33 Tabela de roteamento no roteador D antes de receber anúncio do roteador A

Sub-rede de destino	Roteador seguinte	Número de saltos até o destino
z	C	4
w	–	1
x	–	1
....	....	....

Figura 4.34 Anúncio vindo do roteador A

Vamos agora considerar alguns dos aspectos da implementação do RIP. Lembre-se de que os roteadores RIP trocam anúncios a cada 30 segundos aproximadamente. Se um roteador não ouvir nada de seu vizinho ao menos uma vez a cada 180 segundos, esse vizinho será considerado impossível de ser alcançado dali em diante, isto é, o vizinho está inoperante ou o enlace de conexão caiu. Quando isso acontece, o RIP modifica a tabela de roteamento local e, em seguida, propaga essa informação enviando anúncios a seus roteadores vizinhos (os que ainda podem ser alcançados). Um roteador pode também requisitar informação sobre o custo de seu vizinho até um dado destino usando uma mensagem de requisição RIP. Roteadores enviam mensagens RIP de requisição e de resposta uns aos outros usando o número de porta 520. O segmento UDP é carregado entre roteadores dentro de um datagrama IP padrão. O fato de o RIP usar um protocolo de camada de transporte (UDP) sobre um protocolo de camada de rede (IP) para implementar funcionalidade de camada de rede (um algoritmo de roteamento) pode parecer bastante confuso (e é!). Um exame mais profundo sobre como o RIP é implementado esclarecerá esse assunto.

A Figura 4.36 ilustra esquematicamente como o RIP é comumente implementado em um sistema UNIX, como, por exemplo, uma estação de trabalho UNIX que está servindo como um roteador. Um processo denominado routed roda o RIP, isto é, mantém informações de roteamento e troca mensagens com processos routed que rodam em roteadores vizinhos. Como o RIP é implementado como um processo da camada de aplicação (se bem que um processo muito especial, que é capaz de manipular as tabelas de roteamento dentro do núcleo do UNIX), ele pode enviar e receber mensagens por uma porta padrão e usar um protocolo de transporte padrão. Assim, o RIP é um protocolo de camada de aplicação (veja o Capítulo 2) que roda sobre UDP.

4.6.2 Roteamento intra-AS na Internet: OSPF

Como o RIP, o roteamento OSPF é usado amplamente para roteamento intra-AS na Internet. O OSPF e seu primo, IS-IS, muito parecido com ele, são comumente disponibilizados em ISPs de níveis mais altos, ao passo que o RIP é disponibilizado em ISPs de níveis mais baixos e redes corporativas. O 'open' do OSPF significa que as especificações do protocolo de roteamento estão disponíveis ao público (ao contrário do protocolo EIGRP da Cisco, por exemplo). A versão mais recente do OSPF, versão 2, está definida no RFC 2178 — um documento público.

Sub-rede de destino	Roteador seguinte	Número de saltos até o destino
w	A	2
y	B	2
z	A	5
....	....	....

Figura 4.35 Tabela de roteamento no roteador D após ter recebido anúncio do roteador A

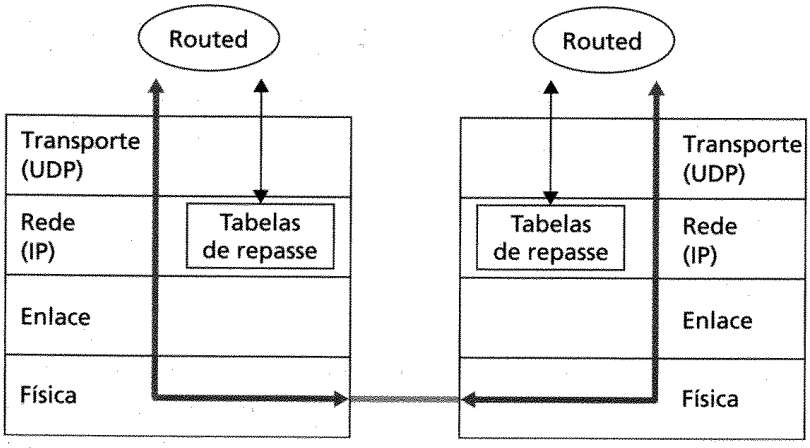


Figura 4.36 Implementação do RIP como um *daemon routed*

O OSPF foi concebido como sucessor do RIP e como tal tem uma série de características avançadas. Em seu âmago, contudo, ele é um protocolo de estado de enlace que usa *broadcasting* de informação de estado de enlace e um algoritmo de caminho de menor custo de Dijkstra. Com o OSPF, um roteador constrói um mapa topológico completo (isto é, um grafo) de todo o sistema autônomo. O roteador então roda localmente o algoritmo do caminho mais curto de Dijkstra para determinar uma árvore de caminho mais curto para todas as sub-redes, sendo ele próprio o nó raiz. Os custos de enlaces individuais são configurados pelo administrador da rede (veja Princípios na prática: determinando pesos OSPF). O administrador pode optar por estabelecer todos os custos de enlace em 1, conseguindo assim roteamento de salto mínimo, ou pode optar por designar para os enlaces pesos inversamente proporcionais à capacidade do enlace de modo a desincentivar o tráfego a usar enlaces de largura de banda baixa. O OSPF não impõe uma política para o modo como são determinados os pesos dos enlaces (essa tarefa é do administrador da rede); em vez disso, provê os mecanismos (protocolo) para determinar o caminho de roteamento de menor custo para um dado conjunto de pesos de enlaces.

Com OSPF, um roteador transmite informações de roteamento a *todos* os outros roteadores no sistema autônomo, e não apenas a seus roteadores vizinhos. Um roteador transmite informações de estado de enlace sempre que houver uma mudança no estado de um enlace (por exemplo, uma mudança de custo ou uma mudança de estado para cima/para baixo). Também transmite o estado de um enlace periodicamente (pelo menos a cada 30 minutos), mesmo que o estado não tenha mudado. O RFC 2328 observa que "essa atualização periódica de anúncios de enlace adiciona robustez ao algoritmo de estado de enlace". Anúncios OSPF são contidos em mensagens OSPF que são carregadas diretamente por IP, com um código protocolo de camada superior igual a 89 para OSPF. Assim, o próprio protocolo OSPF tem de implementar funcionalidades como transferência confiável de mensagem e transmissão *broadcast* de estado de enlace. O protocolo OSPF também verifica se os enlaces estão operacionais (via uma mensagem HELLO enviada a um vizinho ligado ao enlace) e permite que um roteador OSPF obtenha o banco de dados de um roteador vizinho referente ao estado do enlace no âmbito da rede.

Alguns dos avanços incorporados ao OSPF são:

- *Segurança.* Trocas entre roteadores OSPF (por exemplo, atualizações do estado de enlace) são autenticadas. A autenticação garante que apenas roteadores de confiança podem participar do protocolo OSPF dentro de um AS, evitando, assim, que intrusos mal-intencionados (ou estudantes de rede testando, por brincadeira, seu conhecimento recentemente adquirido) injetem informações incorretas em tabelas de roteamento. Por default, pacotes OSPF entre roteadores não são autenticados e poderiam ser forjados. Dois tipos de autenticação podem ser configurados — simples e MD5 (veja o Capítulo 8 para uma discussão sobre MD5 e autenticação em geral). Com autenticação simples, a mesma senha é configurada em cada roteador. Quando um roteador envia um paco-

te OSPF, inclui a senha em texto normal (não criptografado). É claro que a autenticação simples não é segura. A autenticação MD5 é baseada em chaves secretas compartilhadas que são configuradas em todos os roteadores. Cada roteador calcula um hash MD5 para cada pacote OSPF com base no conteúdo do pacote e na chave secreta configurada e, então, inclui o valor de hash resultante no pacote OSPF. O roteador receptor, usando a chave secreta preconfigurada, calculará um hash MD5 do pacote o comparará com o valor de hash que o pacote carrega, verificando assim a autenticidade do pacote. Números de seqüência também são utilizados com autenticação MD5 para proteção contra ataques por reenvio.

- *Caminhos múltiplos de igual custo.* Quando vários caminhos até o destino têm o mesmo custo, o OSPF permite que sejam usados vários caminhos (isto é, não é preciso escolher um caminho único para carregar todo o tráfego quando existem vários caminhos de igual custo).
- *Suporte integrado para roteamento* unicast *e* multicast. O OSPF Multicast (MOSPF) [RFC 1584] fornece extensões simples ao OSPF para prover roteamento *multicast* (um tópico que examinaremos com mais profundidade na Seção 4.8). O MOSPF usa o banco de dados de enlaces existente no OSPF e adiciona um novo tipo de anúncio de estado de enlace ao mecanismo OSPF existente de transmissão de estado de enlace.
- *Suporte para hierarquia dentro de um único domínio de roteamento.* Talvez o avanço mais significativo do OSPF seja a capacidade de estruturar hierarquicamente um sistema autônomo. Na Seção 4.5.3, vimos as muitas vantagens de estruturas de roteamento hierárquicas. Examinaremos a implementação do roteamento OSPF hierárquico no restante desta seção.

Um sistema autônomo OSPF pode ser configurado em áreas. Cada área roda seu próprio algoritmo de roteamento de estado de enlace OSPF, sendo que cada roteador em uma área transmite seu estado de enlace a todos os outros roteadores daquela área. Assim, os detalhes internos de uma área permanecem invisíveis para todos os roteadores externos a ela. O roteamento intra-área envolve apenas os roteadores que estão na mesma área.

Dentro de cada área, um ou mais **roteadores de borda de área** são responsáveis pelo roteamento de pacotes fora da área. Exatamente uma área OSPF no AS é configurada para ser a área de **backbone**. O papel primordial da área de backbone é rotear tráfego entre as outras áreas do AS. O backbone sempre contém todos os roteadores de borda de área que estão dentro do AS e pode conter também roteadores que não são de borda. O roteamento inter-área dentro do AS requer que o pacote seja primeiramente roteado até um roteador de borda de área (roteamento intra-área), em seguida roteado por meio do backbone até o roteador de borda de área que está na área de destino e, então, roteado até seu destino final.

Um diagrama de rede OSPF hierarquicamente estruturada é mostrado na Figura 4.37. Podemos identificar quatro tipos de roteadores OSPF nessa figura:

- *Roteadores internos.* Esses roteadores estão em áreas que não são de backbone e realizam apenas roteamento intra-AS.
- *Roteadores de borda de área.* Esses roteadores pertencem tanto a uma área quanto ao backbone.
- *Roteadores de backbone (roteadores que não são de borda).* Esses roteadores realizam roteamento dentro do backbone, mas não são roteadores de borda de área. Dentro de uma área que não é de backbone, roteadores internos ficam sabendo da existência de roteadores para outras áreas por informações (essencialmente um anúncio de estado de enlace, mas que anuncia o custo de uma rota para uma outra área, e não o custo de um enlace) transmitidas internamente à área por seus roteadores de backbone.
- *Roteadores de borda.* Um roteador de borda troca informações de roteamento com roteadores pertencentes a outros sistemas autônomos. Esse roteador poderia, por exemplo, usar BGP (*border gateway protocol*) para realizar roteamento inter-AS. É por meio desse roteador de borda que outros roteadores ficam a par dos caminhos para redes externas.

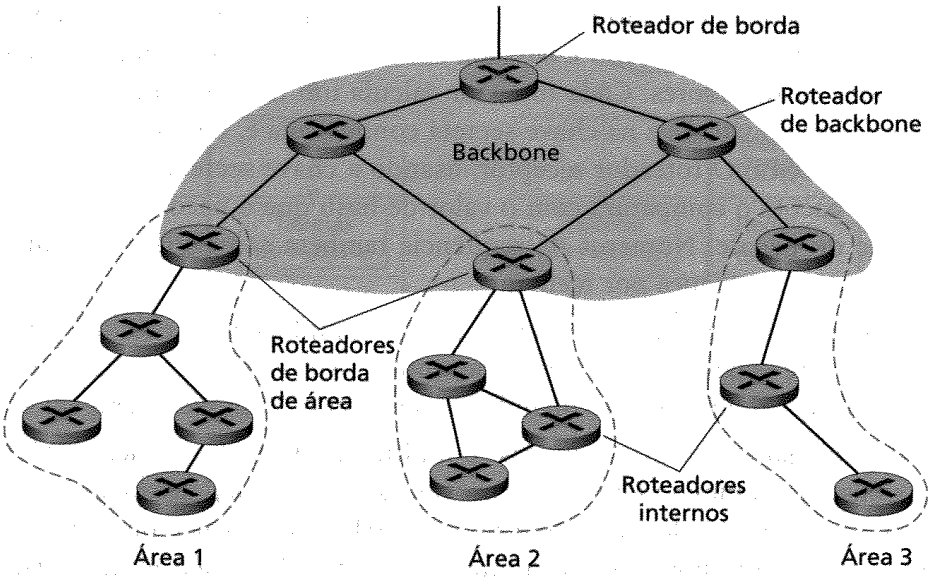


Figura 4.37 Sistema autônomo OSPF hierarquicamente estruturado com quatro áreas

O OSPF é um protocolo relativamente complexo e aqui, nosso tratamento foi necessariamente breve; [Huitema, 1998; Moy, 1998; RFC 2328] fornecem detalhes adicionais.

Princípios na prática

Determinação de pesos de enlaces OSPF

Nossa discussão de roteamento de estado de enlace admitiu implicitamente que os pesos dos enlaces são determinados, que um algoritmo de roteamento como o OSPF é rodado e que o tráfego flui de acordo com as tabelas de roteamento calculadas pelo algoritmo LS. Em termos de causa e efeito, os pesos dos enlaces são dados (isto é, vêm em primeiro lugar) e resultam (via o algoritmo de Dijkstra) em caminhos de roteamento que minimizam o custo geral. Desse ponto de vista, pesos de enlaces refletem o custo da utilização de um enlace (por exemplo, se os pesos dos enlaces forem inversamente proporcionais à capacidade, então a utilização de enlaces de alta capacidade teria pesos menores e, assim, seriam mais atraentes do ponto de vista de roteamento) e o algoritmo de Dijkstra serve para minimizar o custo geral.

Na prática, a relação causa e efeito entre pesos de enlaces e caminhos de roteamento pode ser invertida — operadores de rede configuram pesos de enlaces de modo a obter caminhos de roteamento que cumpram certas metas de engenharia de tráfego [Fortz, 2000; Fortz, 2002]. Por exemplo, suponha que um operador de rede tenha uma estimativa de fluxo do tráfego que entra na rede em cada ponto de ingresso na rede e destinado a cada ponto de saída da rede. Então, o operador poderia querer instituir um roteamento de tráfego do ponto de ingresso até o ponto de saída que minimizasse a utilização máxima em todos os enlaces da rede. Mas, com um algoritmo de roteamento como o OSPF, as principais ferramentas de que o operador dispõe para ajustar a sintonia do roteamento de fluxos através da rede são os pesos dos enlaces. Assim, para cumprir a meta de minimizar a utilização máxima do enlace, o operador tem de descobrir o conjunto de pesos de enlaces que alcance esse objetivo. Essa é um inversão da relação causa e efeito — o roteamento de fluxos desejado é conhecido e os pesos dos enlaces OSPF têm de ser encontrados de um modo tal que o algoritmo de roteamento OSPF resulte nesse roteamento de fluxos desejado.

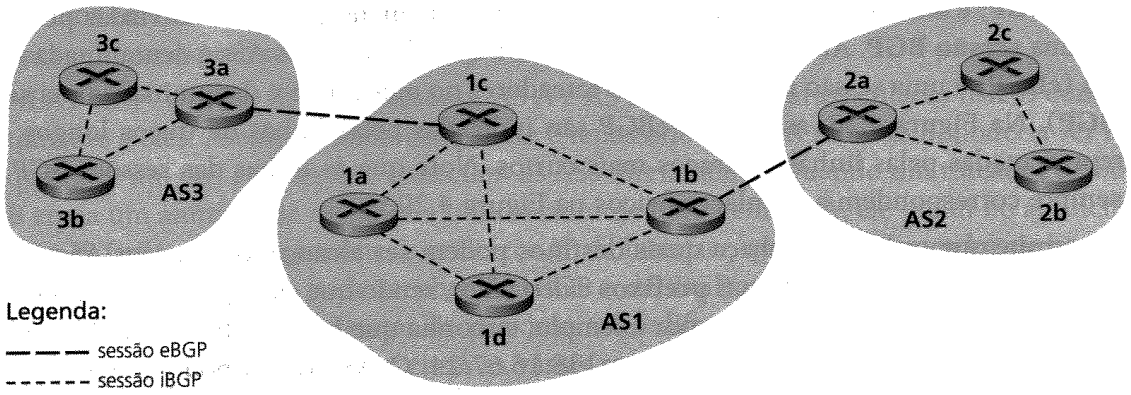


Figura 4.38 Sessões eBGP e iBGP

4.6.3 Roteamento externo a sistemas autônomos: BGP

Acabamos de aprender como ISPs utilizam RIP e OSPF para determinar caminhos ótimos para pares fonte-destino internos ao mesmo AS. Agora vamos examinar como são determinados caminhos para pares fonte-destino que abrangem vários ASs. A versão 4 do **Protocolo de Roteador de Borda (Border Gateway Protocol — BGP)**, especificada no RFC 1771 (veja também [RFC 1772; RFC 1773]), é o padrão, *de facto*, para roteamento entre sistemas autônomos na Internet de hoje. Esse protocolo é comumente denominado BGP4 ou simplesmente **BGP**. Na qualidade de um protocolo de roteamento inter-ASs (veja Seção 4.5.3), o BGP provê a cada AS meios de:

1. Obter de ASs vizinhos informações de atingibilidade de sub-redes.
2. Propagar a informação de atingibilidade a todos os roteadores internos ao AS.
3. Determinar rotas 'boas' para sub-redes com base na informação de atingibilidade e na política do AS.

Em particular, o BGP permite que cada sub-rede anuncie sua existência ao restante da Internet. Uma sub-rede grita "Eu existo e estou aqui" e o BGP garante que todos os ASs da Internet saibam da existência dessa sub-rede e como chegar até ela. Não fosse o BGP, cada sub-rede ficaria isolada — sozinha e desconhecida pelo restante da Internet.

O básico do BGP

O BGP é extremamente complexo; livros inteiros foram dedicados ao assunto. Além disso, mesmo após ler os livros e os RFCs, ainda assim você talvez ache difícil dominar o BGP totalmente sem ter trabalhado com ele na prática durante muitos meses (se não anos) como projetista ou administrador de um ISP de nível superior. Não obstante, como o BGP é um protocolo absolutamente crítico para a Internet — em essência, é o protocolo que agrega tudo —, precisamos ao menos entender os aspectos rudimentares do seu funcionamento. Começamos descrevendo como o BGP poderia funcionar no contexto do exemplo de rede simples que estudamos anteriormente na Seção 4.5.3; aconselhamos que você leia novamente esse material.

No BGP, pares de roteadores trocam informações de roteamento por conexões TCP semipermanentes usando a porta 179. As conexões TCP semipermanentes para a rede da Figura 4.29 são mostradas na Figura 4.38.

Normalmente há uma dessas conexões BGP TCP para cada enlace que conecta diretamente dois roteadores que estão em dois ASs diferentes; assim, na Figura 4.38 há uma conexão TCP entre os roteadores de borda 3a e 1c e uma outra conexão TCP entre os roteadores de borda 1b e 2a. Também há conexões BGP TCP semipermanentes entre roteadores dentro de um AS. Em particular, a Figura 4.38 apresenta uma configuração comum de uma conexão TCP para cada par de roteadores internos a um AS, criando uma malha de conexões TCP dentro de cada AS. Os dois roteadores nas extremidades de cada conexão TCP são deno-

minados **pares BGP**, e a conexão TCP, juntamente com todas as mensagens BGP enviadas pela conexão, é denominada uma **sessão BGP**. Além disso, uma sessão BGP que abranja dois ASs é denominada uma **sessão BGP externa (eBGP)** e uma sessão BGP entre dois roteadores no mesmo AS é denominada uma **sessão BGP interna (iBGP)**. Na Figura 4.38, as sessões eBGP são indicadas pelas linhas de traços longos; as sessões iBGP são representadas pelas linhas de traços mais curtos. Note que as linhas das sessões BGP na Figura 4.38 nem sempre correspondem aos enlaces físicos na Figura 4.29.

O BGP permite que cada AS conheça quais destinos podem ser alcançados via seus ASs vizinhos. Em BGP, os destinos não são hospedeiros, mas **prefixos** *ciderizados*, sendo que cada prefixo representa uma sub-rede ou um conjunto de sub-redes. Assim, por exemplo, suponha que haja quatro sub-redes conectadas ao AS2: 138.16.64/24, 138.16.65/24, 138.16.66/24 e 138.16.67/24. O AS2 então poderia agregar os prefixos dessas quatro sub-redes e utilizar o BGP para anunciar ao AS1 o prefixo único 138.16.64/22. Como outro exemplo, suponha que somente as primeiras três dessas quatro sub-redes estão em AS2 e que a quarta sub-rede, 138.16.67/24, está em AS3. Então, como descrito no quadro Princípios na Prática na Seção 4.4.2, os roteadores usam compatibilização de prefixo mais longo para repassar datagramas. Portanto, o AS3 poderia anunciar a AS1 o prefixo mais específico 138.16.67/24 e o AS2 ainda poderia anunciar ao AS1 o prefixo agregado 138.16.64/22. Podemos imaginar cada anúncio de prefixo como uma promessa. Quando um AS anuncia um prefixo a um outro AS, ele está prometendo que repassará qualquer datagrama destinado ao prefixo através de um caminho na direção desse prefixo.

Agora vamos examinar como o BGP distribuiria informações sobre a atingibilidade de prefixos pelas sessões BGP mostradas na Figura 4.38. Como seria de esperar, usando a sessão eBGP entre os roteadores de borda 3a e 1c, AS3 envia a AS1 a lista de prefixos que podem ser alcançados a partir de AS3; e AS1 envia a AS3 a lista de prefixos que podem ser alcançados de AS1. De modo semelhante, AS1 e AS2 trocam informações de atingibilidade de prefixos por meio de seus roteadores de borda 1b e 2a. E, como também era de esperar, quando um roteador de borda (em qualquer AS) recebe prefixos conhecidos pelo BGP, ele usa suas sessões iBGP para distribuir os prefixos aos outros roteadores no AS. Assim, não somente os roteadores pertencentes a AS1 que não são eBGP, mas também o roteador de borda 1b ficarão a par dos prefixos de AS3. Por conseguinte, o roteador de borda 1b (em AS1) pode reanunciar a AS2 os prefixos de AS3. Quando um roteador (seja ou não de borda) fica sabendo de um novo prefixo, cria um registro para esse prefixo em sua tabela de repasse, como descrito na Seção 4.5.3.

Atributos de caminho e roteadores BGP

Agora que já temos um conhecimento preliminar do BGP, vamos um pouco mais fundo (e, mesmo assim, estaremos varrendo para baixo do tapete alguns dos detalhes menos importantes!). No BGP, um sistema autônomo é identificado por seu **número de sistema autônomo** (ASN) globalmente exclusivo [RFC 1930]. (Tecnicamente, nem todo AS tem um ASN. Em particular, um *stub* AS que carregue somente tráfego do qual é uma fonte ou um destino normalmente não terá um ASN; ignoramos essa tecnicidade em nossa discussão para que os detalhes não nos impeçam de observar melhor o quadro geral.) Números de ASs, tal como endereços IP, são designados por entidades regionais ICANN de registro [ICANN, 2004].

Quando um roteador anuncia um prefixo para uma sessão BGP, inclui vários **atributos BGP** juntamente com o prefixo. Na terminologia do BGP, um prefixo, juntamente com seus atributos, é denominado uma **rota**. Assim, pares BGP anunciam rotas uns aos outros. Dois dos atributos mais importantes são AS-PATH e NEXT-HOP:

- *AS-PATH*. Este atributo contém os ASs pelos quais passou o anúncio para o prefixo. Quando um prefixo é passado para dentro de um AS, o AS adiciona seu ASN ao atributo AS-PATH. Por exemplo, considere a Figura 4.38 e suponha que o prefixo 138.16.64/24 seja anunciado primeiramente de AS2 para AS1; se AS1 então anunciar o prefixo a AS3, o AS-PATH seria AS2 AS1. Roteadores usam o atributo AS-PATH para detectar e evitar looping de anúncios; especificamente, se um roteador perceber que seu AS está contido na lista de caminhos, rejeitará o anúncio. Como discu-

tiremos em breve, roteadores também usam o atributo AS-PATH ao escolher entre vários caminhos para o mesmo prefixo.

- *NEXT-HOP*. Um par de ASs, digamos, AS A e AS B, pode ter vários enlaces físicos conectados diretamente a eles. Assim, quando um pacote é repassado de AS A para AS B, poderia ser enviado por qualquer um dos enlaces que conectam os ASs diretamente. Quando um roteador de borda em AS B envia um anúncio de roteador a um roteador de borda em AS A, ele inclui no anúncio seu endereço IP (mais especificamente, o endereço IP da interface que leva ao roteador de borda em AS A). Um roteador em AS A pode receber de eBGP e de iBGP várias rotas para o mesmo prefixo, cada uma passando por um roteador de próximo salto diferente. Ao configurar sua tabela de repasse, esse roteador teria de escolher entre as várias rotas.

O BGP também inclui atributos que permitem que roteadores designem a métrica de sua preferência às rotas, e um atributo que indica como o prefixo foi inserido no BGP na origem. Para uma discussão completa de atributos de rota, veja [Griffin, 2002; Stewart, 1999; Halabi, 2000; Feamster, 2004].

Quando um roteador de borda recebe um anúncio de roteador, ele utiliza sua **política de importação** para decidir se aceita ou filtra a rota e se estabelece certos atributos, tal como a métrica de preferência do roteador. A política de importação pode filtrar uma rota porque o AS pode não querer enviar tráfego por um dos ASs no AS-PATH. O roteador de borda também pode filtrar uma rota porque já conhece uma rota preferencial para o mesmo prefixo.

Seleção de rota do BGP

Como descrito anteriormente nesta seção, o BGP usa eBGP e iBGP para distribuir rotas a todos os roteadores dentro de ASs. Com essa distribuição, um roteador pode conhecer mais do que uma rota para qualquer prefixo determinado, caso em que deve selecionar uma das rotas possíveis. O dado usado por esse processo de seleção de rota é o conjunto de todas as rotas que o roteador conhece e aceitou. Se houver duas ou mais rotas para o mesmo prefixo, então o BGP invoca seqüencialmente as seguintes regras de eliminação até sobrar apenas uma rota:

- Rotas recebem, como um de seus atributos, um valor de preferência local. A preferência local de uma rota pode ter sido estabelecida pelo roteador ou pode ter sido aprendida por um outro roteador no mesmo AS. Essa é uma decisão política que fica a cargo do administrador da rede do AS. (Mais adiante discutiremos questões de política do BGP.) São selecionadas as rotas que têm os valores de preferência local mais altos.
- No caso das outras rotas remanescentes (todas com o mesmo valor de preferência local), é selecionada a rota que tenha o AS-PATH mais curto. Se essa fosse a única regra de seleção de rotas, então o BGP estaria usando um algoritmo DV para determinação de caminho no qual a métrica da distância utiliza o número de saltos de AS em vez do número de saltos de roteadores.
- Dentre as rotas remanescentes (todas com o mesmo valor de preferência local e com o mesmo comprimento de AS-PATH) é selecionada a rota que tenha o roteador NEXT-HOP mais próximo. Nesse caso, mais próximo significa o roteador que tenha o menor custo de caminho de menor custo determinado pelo algoritmo intra-AS. Esse processo é normalmente denominado roteamento da batata quente.
- Se ainda restar mais de uma rota, o roteador usa identificadores BGP para selecionar a rota; veja [Stewart, 1999].

As regras de eliminação são ainda mais complicadas do que descrevemos aqui. Para evitar pesadelos com o BGP, é melhor aprender suas regras de seleção em pequenas doses!

Política de roteamento

Vamos ilustrar alguns dos conceitos básicos do roteamento BGP com um exemplo simples. A Figura 4.39 mostra seis sistemas autônomos interconectados: A, B, C, W, X, e Y. É importante notar que A, B, C, W, X, e Y são ASs, e não roteadores. Vamos admitir que os sistemas autônomos W, X, e Y são redes stub e que A, B, e C são redes provedoras de backbone. Todo o tráfego que entrar em uma **rede stub** deve ser destinado a essa rede, e todo o tráfego que sair da rede stub deve ter sido originado naquela rede. W e Y são claramente redes stub. X é uma **rede stub com múltiplas interconexões**, visto que está conectado ao resto da rede por meio de dois provedores diferentes (um cenário que está se tornando cada vez mais comum na prática). Todavia, tal como W e Y, o próprio X deve ser a origem/destino de todo o tráfego que entra/sai de X. Mas, como esse comportamento da rede stub será implementado e imposto? Como X será impedido de repassar tráfego entre B e C? Isso pode ser conseguido facilmente controlando o modo como as rotas BGP são anunciadas. Em particular, X funcionará como uma rede stub se anunciar (a seus vizinhos B e C) que não há nenhum caminho para quaisquer outros destinos a não ser ele mesmo. Isto é, mesmo que X conheça um caminho, digamos, XCY, que chegue até a rede Y, ele *não* anunciará esse caminho a B. Uma vez que B não fica sabendo que X tem um caminho para Y, B nunca repassaria tráfego destinado a Y (ou a C) via X. Esse exemplo simples ilustra como uma política seletiva de anúncio de rota pode ser usada para implementar relacionamentos de roteamento cliente/provedor.

Em seguida, vamos focalizar uma rede provedora, digamos AS B. Suponha que B ficasse sabendo (por A), que A tem um caminho AW para W. Assim, B pode instalar a rota BAW em sua base de informações de roteamento. É claro que B também quer anunciar o caminho BAW a seu cliente, X, de modo que X saiba que pode rotear para W via B. Mas, B deveria anunciar o caminho BAW a C? Se o fizer, então C poderia rotear tráfego para W via CBAW. Se A, B e C forem todos provedores de backbone, então B poderia sentir-se no direito de achar que não deveria ter de suportar a carga (e o custo!) de carregar tráfego em trânsito entre A e B. B poderia sentir-se no direito de achar que é de A e C o trabalho (e o custo) de garantir que C possa rotear de/para clientes de A via uma conexão direta entre A e C. Atualmente não existe nenhum padrão oficial que determine como ISPs de backbone devem rotear entre si. Todavia, os ISPs comerciais adotam uma regra prática que diz que qualquer tráfego que esteja fluindo por uma rede de backbone de um ISP deve ter ou uma origem ou um destino (ou ambos) em uma rede que seja cliente daquele ISP; caso contrário, o tráfego estaria pegando uma carona gratuita na rede do ISP. Acordos individuais de parceria (*peering*) (para reger questões como as levantadas acima) normalmente são negociados entre pares de ISPs e, em geral, são confidenciais; [Huston, 1999a] provê uma discussão interessante sobre acordos de parceria. Se quiser uma descrição detalhada sobre como a política de roteamento reflete os relacionamentos comerciais entre ISPs, veja [Gao, 2001].

Como já observamos, o BGP é um padrão *de facto* para roteamento inter-AS na Internet pública. Para ver o conteúdo de várias tabelas de roteamento BGP (grandes!) extraídas de roteadores pertencentes a ISPs de nível 1, consulte http://www.routeviews.org. Tabelas de roteamento BGP geralmente contêm dezenas de milhares de prefixos e atributos correspondentes. Estatísticas sobre tamanho e características de tabelas de roteamento BGP são apresentadas em [Huston, 2001].

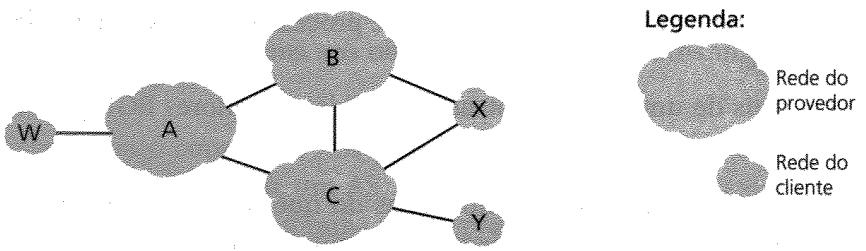


Figura 4.39 Um cenário BGP simples

Por que há diferentes protocolos de roteamento inter-AS e intra-AS?

Agora que já examinamos os detalhes de protocolos de roteamento inter-AS e intra-AS específicos utilizados pela Internet de hoje, vamos concluir considerando a questão talvez mais fundamental que, antes de mais nada, poderíamos levantar sobre esses protocolos (esperamos que você tenha estado preocupado com isso o tempo todo e que não tenha deixado de enxergar o quadro geral por causa dos detalhes!). Por que são usados diferentes protocolos de roteamento inter-AS e intra-AS?

A resposta a essa pergunta expõe o âmago da diferença entre os objetivos do roteamento interno a um AS e entre ASs:

- *Política.* Entre ASs, as questões políticas dominam. Pode até ser importante que o tráfego que se origina em um dado AS não possa passar através de um outro AS específico. De maneira semelhante, um dado AS pode muito bem querer controlar o tráfego em trânsito que ele carrega entre outros ASs. Vimos que o BGP carrega atributos de caminho e oferece distribuição controlada de informação de roteamento, de modo que essas decisões de roteamento baseadas em políticas possam ser tomadas. Dentro de um AS, tudo está nominalmente sob o mesmo controle administrativo. Assim, as questões de políticas de roteamento desempenham um papel bem menos importante na escolha de rotas dentro do AS.
- *Escalabilidade.* A escalabilidade de um algoritmo de roteamento e de suas estruturas de dados para manipular o roteamento para/entre grandes números de redes é uma questão fundamental para o roteamento inter-AS. Dentro de um AS, a escalabilidade é uma preocupação menor. Isso porque, se um único domínio administrativo ficar muito grande, é sempre possível dividi-lo em dois ASs e realizar roteamento inter-AS entre esses dois novos ASs. (Lembre-se de que o OSPF permite que essa hierarquia seja construída dividindo um AS em áreas.)
- *Desempenho.* Dado que o roteamento inter-AS é bastante orientado pelas políticas, a qualidade (por exemplo, o desempenho) dos roteadores usados é freqüentemente uma preocupação secundária (isto é, uma rota mais longa ou de custo mais alto que satisfaça certos critérios políticos pode muito bem prevalecer sobre uma rota que é mais curta, mas que não satisfaz esses critérios). Na verdade, vimos que entre ASs não há nem mesmo a idéia de custo associado às rotas (exceto a contagem de saltos). Dentro de um AS individual, contudo, essas preocupações com as políticas podem ser ignoradas, permitindo que o roteamento se concentre mais no nível de desempenho atingido em uma rota.

Com isso concluímos nossa breve introdução ao BGP. Entender esse protocolo é importante porque ele desempenha um papel central na Internet. Aconselhamos você a consultar as referências [Griffin, 2002; Stewart, 1999; Labovitz, 1997; Halabi, 2000; Huitema, 1998; Gao, 2001; Feamster, 2004] para aprender mais sobre BGP.

4.7 Roteamento broadcast e multicast

Até este ponto do capítulo, focalizamos protocolos de roteamento que suportam comunicação *unicast* (isto é, ponto-a-ponto) na qual um único nó de fonte envia um pacote a um único nó de destino. Nesta seção, voltaremos nossa atenção a protocolos de roteamento *broadcast* e *multicast*. No **roteamento broadcast** a camada de rede provê um serviço de entrega de um pacote enviado de um nó de fonte a todos os outros nós da rede; o **roteamento multicast** habilita um único nó de fonte a enviar uma cópia de um pacote a um subconjunto de nós das outras redes. Na Seção 4.7.1 consideraremos algoritmos de roteamento *broadcast* e sua incorporação em protocolos de roteamento. Examinaremos roteamento *multicast* na Seção 4.7.2.

4.7.1 Algoritmos de roteamento broadcast

Talvez o modo mais direto de conseguir comunicação *broadcast* é o nó remetente enviar uma cópia separada do pacote para cada destino, como mostra a Figura 4.40(a). Dados N nós de destino, o nó da fonte simplesmente faz N cópias do pacote, endereça cada cópia a um destino diferente e então transmite N cópias aos N destinos usando roteamento *unicast*. Essa abordagem ***unicast* de N caminhos** da transmissão *broadcast* é simples — não é preciso nenhum novo protocolo de roteamento de camada de rede, nem duplicação de pacotes, nem funcionalidade de repasse. Porém, essa abordagem tem várias desvantagens. A primeira desvantagem é sua ineficiência. Se o nó da fonte estiver conectado ao resto da rede por um único enlace, então N cópias separadas do (mesmo) pacote transitarão por esse único enlace. Evidentemente seria mais eficiente enviar somente uma única cópia de um pacote até esse primeiro salto e então fazer com que o nó na outra extremidade do primeiro salto produzisse e transmitisse quaisquer cópias adicionais necessárias. Isto é, seria mais eficiente que os próprios nós da rede (em vez de apenas o nó da fonte) criassem e duplicassem cópias de um pacote. Por exemplo, na Figura 4.40(b), somente uma única cópia de um pacote transita pelo enlace R1-R2. Aquele pacote então é duplicado em R2, e uma única cópia será enviada pelos enlaces R2-R3 e R2-R4.

As desvantagens adicionais do *unicast* de N caminhos talvez sejam mais sutis, mas nem por isso menos importantes. Uma premissa implícita desse tipo de roteamento é que o remetente conheça os destinatários do *broadcast* e seus endereços. Mas, como essa informação é obtida? Muito provavelmente seriam exigidos mecanismos de protocolo adicionais (tais como associação ao *broadcast* ou protocolo de registro de destino), o que adicionaria mais sobrecarga e, o que é importante, complexidade adicional a um protocolo que, inicialmente parecia bastante simples. Uma desvantagem final do *unicast* de N caminhos está relacionada aos propósitos para os quais o *broadcast* seria utilizado. Na Seção 4.5 aprendemos que protocolos de roteamento de estado de enlace usam *broadcast* para propagar informações de estado de enlace que são usadas para calcular rotas *unicast*. É claro que, em situações em que o *broadcast* é usado para criar e atualizar rotas *unicast*, seria imprudente (no mínimo!) confiar em infra-estrutura de roteamento *unicast* para implantar *broadcast*.

Dadas as diversas desvantagens do *broadcast* de N caminhos, são claramente de interesse abordagens nas quais os próprios nós da rede desempenham um papel ativo na duplicação de pacotes, no repasse de pacotes e no cálculo de rotas de *broadcast*. A seguir, examinaremos diversas dessas abordagens e, mais uma vez, adotaremos a notação de grafo apresentada na Seção 4.5. Modelaremos novamente a rede como um grafo $G = (N, E)$, onde N é um conjunto de nós e uma coleção E de arestas, sendo que cada aresta é um par de nós de N. Não seremos muito exigentes quanto à notação e adotaremos N para denotar ambos os conjuntos de nós, e a cardinalidade ($|N|$) ou o tamanho daquele conjunto, quando não houver possibilidade de confusão.

Inundação não controlada

A técnica mais óbvia para conseguir *broadcast* é uma abordagem de **inundação** na qual o nó da fonte envia uma cópia do pacote a todos os seus vizinhos. Quando um nó recebe um pacote *broadcast*, ele duplica o pacote e o repassa a todos os seus vizinhos (exceto ao vizinho do qual recebeu o pacote). É claro que, se o grafo for conectado, esse esquema eventualmente entregará uma cópia do pacote *broadcast* a todos os nós no grafo. Embora esse esquema seja simples e elegante, tem uma falha fatal (antes de continuar, tente imaginar qual seria essa falha): se o grafo tiver ciclos, então uma ou mais cópias de cada pacote de *broadcast* permanecerão em ciclo indefinidamente. Por exemplo, na Figura 4.40, R2 inundará R3, R3 inundará R4, R4 inundará R2 e R2 inundará (novamente!) R3 e assim por diante. Esse cenário simples resulta no ciclo sem fim de dois pacotes *broadcast*, um em sentido horário e um em sentido anti-horário. Mas pode haver uma falha fatal ainda mais catastrófica: quando um nó estiver conectado a mais de dois outros nós, criará e repassará várias cópias do pacote *broadcast*, cada uma das quais criará várias cópias de si mesmas (em outros nós com mais de dois vizinhos) e assim por diante. Essa **tempestade de *broadcast***, gerada pela infindável multiplicação de pacotes *broadcast*, eventualmente resultaria na criação de uma quantidade tão grande de pacotes *broadcast* que a rede ficaria inutilizada. (Consulte nos exercícios de fixação ao final do capítulo um problema que analisa a velocidade com que essa tempestade de *broadcast* cresce.)

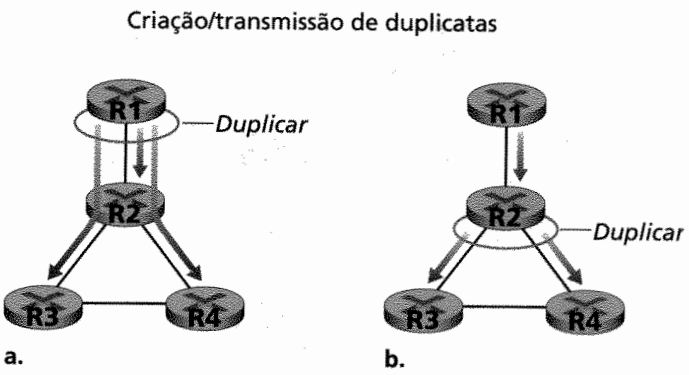


Figura 4.40 Duplicação de fonte versus duplicação dentro da rede

Inundação controlada

A chave para evitar uma tempestade de *broadcast* é um nó escolher sensatamente quando repassa um pacote (por exemplo, se já tiver recebido e repassado uma cópia anterior do pacote) e quando não repassa um pacote. Na prática, há vários modos de fazer isso.

Na **inundação controlada com número de seqüência,** um nó de fonte coloca seu endereço (ou outro identificador exclusivo), bem como um **número de seqüência de *broadcast*** em um pacote *broadcast* e então envia o pacote a todos os seus vizinhos. Cada nó mantém uma lista de endereços de fonte e números de seqüência para cada pacote *broadcast* que já recebeu, duplicou e repassou. Quando um nó recebe um pacote *broadcast*, primeiramente verifica se o pacote está nessa lista. Se estiver, o pacote é descartado; se não estiver, é duplicado e repassado para todos os vizinhos do nó (exceto para o nó de quem o pacote acabou de ser recebido). O protocolo Gnutella, discutido no Capítulo 2, usa inundação controlada com número de seqüência para fazer transmissão *broadcast* de consultas em sua rede de sobreposição. (Em Gnutella, a duplicação e transmissão de mensagens é realizada na camada de aplicação, e não na camada de rede.)

Uma segunda abordagem da inundação controlada é conhecida como **repasse pelo caminho inverso** (*reverse path forwarding* — RPF) [Dalal, 1978], às vezes também denominada *broadcast* pelo caminho inverso (*reverse path broadcast* — RPB). A idéia por trás do repasse pelo caminho inverso é simples, mas elegante. Quando um roteador recebe um pacote *broadcast* com um dado endereço de fonte, ele transmite o pacote para todos os seus enlaces de saída (exceto para aquele do qual o pacote foi recebido) somente se o pacote chegou pelo enlace que está em seu próprio caminho *unicast* mais curto de volta ao remetente. Caso contrário, o roteador simplesmente descarta o pacote que está entrando sem repassá-lo para nenhum de seus enlaces de saída. Esse pacote pode ser descartado porque o roteador sabe que receberá, ou já recebeu, uma cópia desse pacote no enlace que está em seu próprio caminho mais curto de volta ao remetente. (É provável que você esteja querendo se convencer de que isso, de fato, acontecerá, e que não ocorrerão nem looping nem tempestades de *broadcast*.) Note que o RPF não usa roteamento *unicast* para entregar um pacote a um destino, nem exige que um roteador conheça o caminho mais curto completo entre ele mesmo e a fonte. O RPF precisa conhecer apenas o próximo vizinho em seu caminho *unicast* mais curto até o remetente; usa a identidade desse vizinho apenas para determinar se repassa ou não repassa um pacote *broadcast* recebido.

A Figura 4.41 ilustra o RPF. Suponha que os enlaces com as linhas mais grossas representem os caminhos de menor custo desde os destinatários até a fonte (A). O nó *A* inicialmente faz a transmissão *broadcast* de um pacote da fonte *A* até os nós *C* e *B*. O nó *B* repassará o pacote da fonte *A* que recebeu de *A* (uma vez que *A* está em seu caminho de menor custo até *A*) para *C* e *D*. *B* ignorará (descartará sem repassar) quaisquer pacotes da fonte A que receba de quaisquer outros nós (por exemplo, dos roteadores *C* ou *D*). Vamos considerar agora o nó *C*, que receberá um pacote da fonte *A* diretamente de *A* e também de *B*. Uma vez que *B* não está no caminho mais curto de *C* de retorno a *A*, *C* ignorará quaisquer pacotes da fonte *A* que receba de *B*. Por outro lado, quando *C* receber um pacote da fonte *A* diretamente de *A*, ele o repassará aos nós *B*, *E* e *F*.

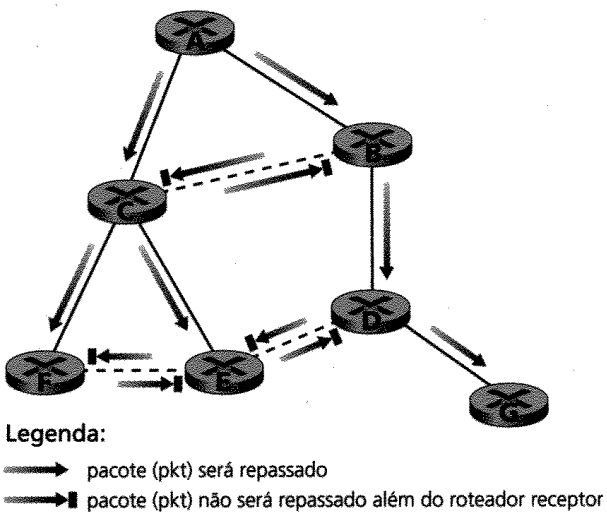
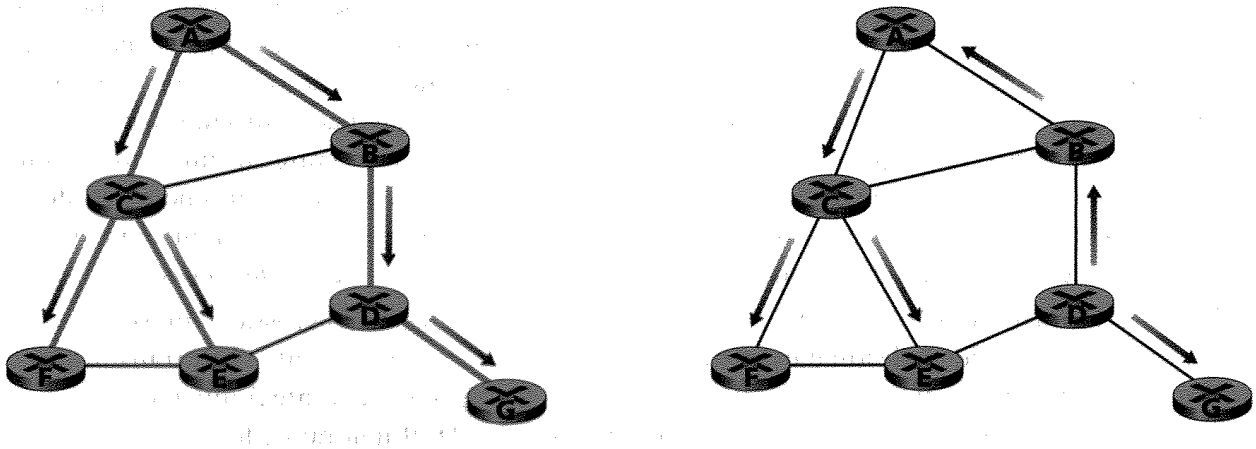


Figura 4.41 Repasse pelo caminho inverso

Broadcast por spanning tree

Conquanto a inundação controlada com números de seqüência e o RPF impeçam tempestades de *broadcast*, não evitam completamente a transmissão de pacotes *broadcast* redundantes. Por exemplo, na Figura 4.42, os nós *B*, *C*, *D* e *F* recebem ou um ou dois pacotes redundantes. Idealmente, cada nó deveria receber somente uma cópia do pacote *broadcast*.

Examinando a árvore formada pelos nós conectados por linhas grossas na Figura 4.42(a), podemos ver que, se os pacotes *broadcast* fossem repassados por enlaces dessa árvore, cada nó da rede receberia exatamente uma cópia do pacote *broadcast* — exatamente a solução que procurávamos! Esse é um exemplo de ***spanning tree*** — uma árvore que contém todos os nós, sem exceção, em um grafo. Mais formalmente, uma *spanning tree* de um grafo $G = (N,E)$ é um grafo $G' = (N,E')$ tal que E' é um subconjunto de E, G' é conectado, G' não contém nenhum ciclo e G' contém todos os nós originais em G. Se cada enlace tiver um custo associado e o custo de uma árvore for a soma dos custos dos enlaces, então uma á arvore cujo custo seja o mínimo entre todas as *spanning trees* do gráfico é denominada uma ***spanning tree* mínima** (o que não é nenhuma surpresa).



a. Broadcast iniciado em A

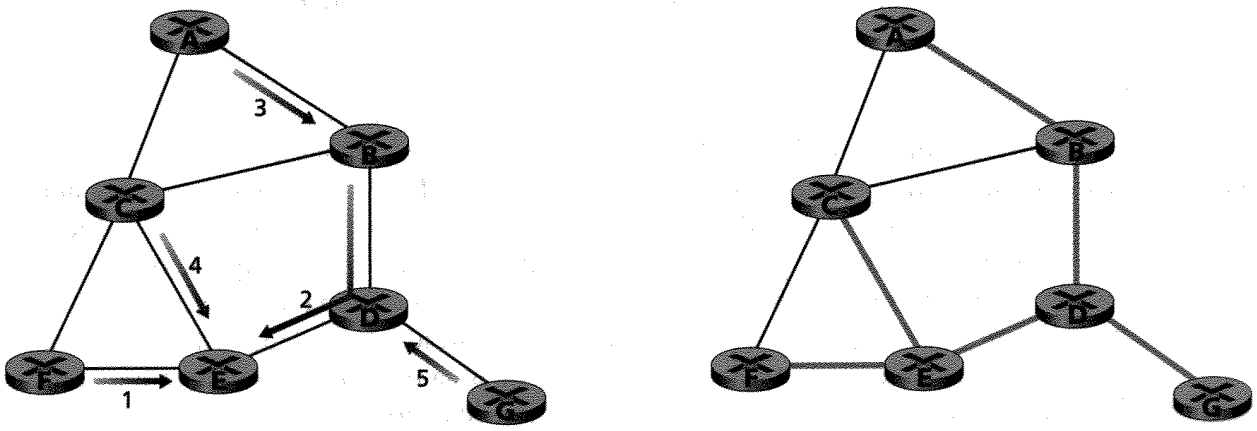
b. Broadcast iniciado em D

Figura 4.42 Broadcast por uma spanning tree

Assim, uma outra abordagem para o fornecimento de *broadcast* é os nós da rede construírem uma *spanning tree*, em primeiro lugar. Quando um nó de fonte quiser enviar um pacote *broadcast*, enviará o pacote por todos os enlaces incidentes que pertençam à *spanning tree*. Um nó que receba um pacote *broadcast* então o repassa a todos os seus vizinhos na *spanning tree* (exceto para o vizinho do qual recebeu o pacote). A *spanning tree* não apenas elimina pacotes *broadcast* redundantes mas, uma vez instalada, pode ser usada por qualquer nó para iniciar um *broadcast*, como mostram as Figuras 4.42(a) e 4.42(b). Note que um nó não precisa estar ciente da árvore inteira; ele precisa simplesmente saber qual de seus vizinhos em *G* são vizinhos que pertencem à *spanning tree*.

A principal complexidade associada com a abordagem de *spanning tree* é sua criação e manutenção. Foram desenvolvidos numerosos algoritmos distribuídos de *spanning tree* [Gallager, 1983; Gartner, 2003]. Aqui, consideramos apenas um algoritmo simples. Na **abordagem de nó central** da construção de uma *spanning tree*, é definido um nó central (também conhecido como **ponto de encontro** ou **núcleo**). Os nós então transmitem em *unicast* mensagens de adesão à árvore endereçadas ao nó central. Uma mensagem de adesão à árvore é repassada usando roteamento *unicast* em direção ao centro até que ela chegue a um roteador que já pertence à árvore spanning ou até que chegue ao centro. Em qualquer um dos casos, o caminho que a mensagem de adesão à árvore seguiu define o ramo da *spanning tree* entre o roteador de porta que enviou a mensagem de adesão à árvore e o centro. Esse novo caminho pode ser visto como um enxerto à *spanning tree* existente.

A Figura 4.43 ilustra a construção de uma *spanning tree* com centro. Suponha que o nó *E* seja selecionado como centro da árvore. Suponha que o nó *F* primeiramente se junte à árvore e repasse uma mensagem de adesão à árvore a *E*. O único enlace *EF* se torna a *spanning tree* inicial. O nó *B* então se junta à *spanning tree* enviando a *E* sua mensagem de adesão à árvore. Suponha que a rota do caminho *unicast* de *E* para *B* seja por *D*. Nesse caso, a mensagem de adesão à árvore resulta no enxerto do caminho *BDE* à *spanning tree*. Em seguida, o nó *A* se junta ao grupo crescente repassando sua mensagem de adesão à árvore em direção a *E*. Se o caminho *unicast* de *A* a *E* passar por *B*, então, uma vez que *B* já se juntou à *spanning tree*, a chegada da mensagem de adesão à árvore de *A* em *B* resultará no enxerto imediato do enlace *AB* à *spanning tree*. Em seguida, o nó *C* se junta à *spanning tree* repassando sua mensagem de adesão à árvore diretamente a *E*. Finalmente, como o roteamento *unicast* de *G* para *E* tem de passar pelo nó *D*, quando *G* enviar sua mensagem de adesão à árvore a *E*, o enlace *GD* será enxertado à *spanning tree* no nó *D*.



a. Construção da spanning tree passo a passo

b. Spanning tree construída

Figura 4.43 Construção de uma spanning tree com centro

Algoritmos de broadcast na prática

Protocolos *broadcast* são usados na prática nas camada de aplicação e de rede. Como discutimos na Seção 2.6, Gnutella [Gnutella, 2004] usa *broadcast* no nível de aplicação para fazer transmissão *broadcast* de consultas de conteúdo entre pares Gnutella. Nesse caso, um enlace entre dois processos pares distribuídos no nível de aplicação na rede Gnutella é, na verdade, uma conexão TCP. O Gnutella usa uma forma de inundação controlada com número de seqüência na qual um identificador de 16 bits e um descritor de carga útil de 16 bits (que identifica o tipo de mensagem Gnutella) são usados para detectar se uma consulta *broadcast* recebida já foi recebida, duplicada e repassada anteriormente. Também discutido na Seção 2.6, o Gnutella usa um campo de tempo de vida (TTL) para limitar o número de saltos pelos quais uma consulta será repassada. Quando um processo Gnutella recebe e duplica uma consulta, ele decrementa o campo de TTL antes de retransmiti-la. Assim, uma consulta Gnutella em *broadcast* alcançará apenas pares que estão dentro de um dado número (o valor inicial do TTL) de saltos de nível de aplicação desde o iniciador da consulta. Por isso, o mecanismo de consultas por inundação do Gnutella às vezes é denominado *inundação de escopo limitado.*

Uma forma de inundação controlada com número de seqüência também é usada para fazer a transmissão *broadcast* de anúncios de estado de enlace (*link-state advertisements* — LSAs) no algoritmo de roteamento OSPF [RFC 2328; Perlman, 1999] e no algoritmo de roteamento Sistema Intermediário a Sistema Intermediário (*Intermediate-System-to-Intermediate-System* — IS-IS) [RFC 1142; Perlman, 1999]. O OSPF usa um número de seqüência de 32 bits, bem como um campo de idade de 16 bits para identificar anúncios de estado de enlace (LSAs). Lembre-se de que um nó OSPF faz *broadcast* periódico de LSAs para os enlaces ligados a ele quando o custo de enlace até um vizinho muda ou quando um enlace ativa ou desativa. Números de seqüência LSA são usados para detectar LSAs duplicados, mas também cumprem uma segunda função importante no OSPF. Com inundação, é possível que um LSA gerado pela fonte no tempo t chegue *após* um LSA mais novo que foi gerado pela mesma fonte no tempo $t + \delta$. Os números de seqüência usados pelo nó da fonte permitem que um LSA mais velho seja distinguido de um LSA mais novo. O campo de idade cumpre uma finalidade semelhante à de um valor TTL. O valor inicial do campo de idade é estabelecido em zero e é incrementado a cada salto à medida que este é retransmitido, e também é incrementado enquanto fica na memória de um roteador esperando para ser enviado. Embora nossa descrição do algoritmo de inundação LSA tenha sido apenas breve, observamos que, na verdade, projetar protocolos de *broadcast* LSA pode ser um negócio muito delicado. [RFC 789; Perlman, 1999, Seção 12.2.3.3] descrevem um incidente no qual LSAs transmitidos incorretamente por dois roteadores defeituosos fizeram com que uma versão antiga do algoritmo de inundação LSA derrubasse a ARPAnet inteira!

4.7.2 Multicast

Vimos, na seção anterior, que, com serviço *broadcast*, pacotes são entregues a cada um e a todos os nós em uma rede. Nessa seção, voltamos nossa atenção para o serviço ***multicast***, no qual um pacote *multicast* é entregue a apenas um *subgrupo* de nós de rede. Uma série de aplicações emergentes de rede requer a entrega de pacotes de um ou mais remetentes a um grupo de destinatários. Entre essas aplicações estão a transferência de dados em grandes volumes (por exemplo, a transferência de uma atualização de software do desenvolvedor do software para os usuários que necessitam da atualização), a recepção de aplicações de taxa constante (por exemplo, transferência de áudio, vídeo e texto de uma palestra ao vivo para um conjunto de participantes distribuídos), as aplicações de dados compartilhados (por exemplo, uma videoconferência compartilhada por muitos participantes distribuídos), a alimentação de dados (por exemplo, cotação de ações), a atualização de arquivos de memória intermediária e os jogos interativos (por exemplo, ambientes virtuais interativos distribuídos ou jogos multiusuários como o Quake).

Na comunicação *multicast*, enfrentamos imediatamente dois problemas — como identificar os destinatários de um pacote *multicast* e como endereçar um pacote enviado a um desses destinatários. No caso da comunicação *unicast*, o endereço IP do destinatário (receptor) é levado em cada datagrama IP *unicast* e identifica o único destinatário; no caso do *multicast*, temos vários destinatários. Tem sentido que cada pacote *multicast* carregue os endereços IP de todos os vários destinatários? Embora essa abordagem possa ser fun-

cional com um número pequeno de destinatários, não tem boa escalabilidade para o caso de centenas de milhares de destinatários; a quantidade de informações de endereçamento no datagrama sobrepujaria a quantidade de dados que é realmente carregada no campo de carga útil do pacote. A identificação explícita dos destinatários pelo remetente também exige que o remetente conheça as identidades e os endereços de todos os destinatários. Veremos em breve que há casos em que essa exigência pode ser indesejável.

Por essas razões, na arquitetura da Internet (e em outras arquiteturas de rede como ATM [Black, 1995]), um pacote *multicast* é endereçado usando **endereço indireto**, isto é, um único identificador é utilizado para o grupo de destinatários e uma cópia do pacote que é endereçada ao grupo usando esse único identificador é entregue a todos os destinatários *multicast* associados ao grupo em questão. Na Internet, o identificador único que representa um grupo de destinatários é um endereço *multicast* classe D. O grupo de destinatários associados a um endereço classe D é denominado **grupo multicast**. A abstração do grupo *multicast* é ilustrada na Figura 4.44. Nessa figura, quatro hospedeiros (no tom mais claro) estão associados ao endereço de grupo *multicast* 226.17.30.197 e receberão todos os datagramas endereçados a esse endereço *multicast*. A dificuldade que ainda temos de enfrentar é o fato de que cada hospedeiro tem um endereço *unicast* exclusivo que é completamente independente do endereço do grupo *multicast* do qual ele está participando.

Embora seja simples, a abstração *multicast* provoca uma série de perguntas. Como um grupo começa e como termina? Como é escolhido o endereço do grupo? Como novos hospedeiros são incorporados ao grupo (seja como remetentes, seja como destinatários)? Qualquer um pode se juntar ao grupo (e enviar para esse grupo e receber dele) ou a participação no grupo é limitada e, se for limitada, quem a limita? Os membros do grupo ficam conhecendo as identidades dos outros membros como parte do protocolo de camada de rede? Como os nós da rede interagem para entregar um datagrama *multicast* a todos os membros do grupo? Para a Internet, as respostas a todas essas perguntas envolvem o Protocolo de Gerenciamento de Grupo da Internet (*Internet Group Management Protocol* — IGMP [RFC 2236]). Assim, vamos considerar o protocolo IGMP. Mais adiante, voltaremos a essas perguntas de caráter mais geral.

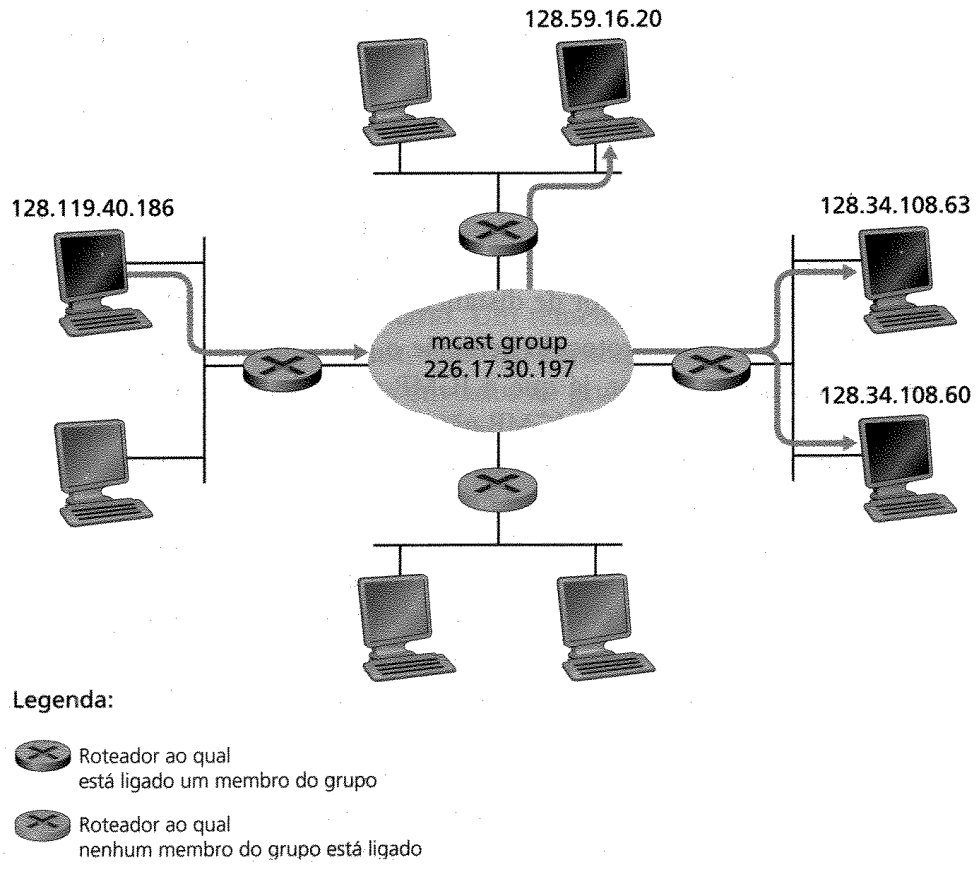


Figura 4.44 O grupo multicast: um datagrama endereçado ao grupo é entregue a todos os membros do grupo multicast

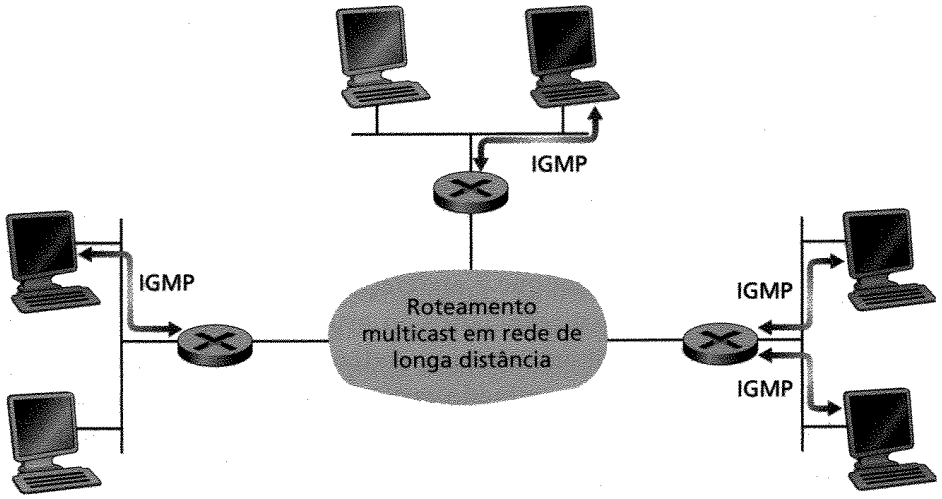


Figura 4.45 Os dois componentes de multicast de camada de rede: IGMP e protocolos de roteamento multicast

Internet Group Management Protocol – IGMP

O protocolo IGMP, versão 3 [RFC 3376] opera entre um hospedeiro e o roteador diretamente conectado a ele (em termos informais, pense no roteador diretamente conectado ao hospedeiro como o roteador de primeiro salto que um hospedeiro veria no caminho até qualquer hospedeiro externo à sua própria rede local, ou como o roteador de último salto em qualquer caminho até esse hospedeiro), como mostra a Figura 4.45. Essa figura mostra os três roteadores *multicast* do primeiro salto, cada um conectado ao hospedeiro ao qual está ligado por uma interface local de saída. Essa interface local está ligada a uma LAN nesse exemplo e, embora cada LAN tenha vários hospedeiros ligados a ela, no máximo apenas alguns deles comumente pertencerão a um dado grupo *multicast* a qualquer determinado instante.

O IGMP provê os meios para um hospedeiro informar ao roteador conectado a ele que uma aplicação que está funcionando no hospedeiro quer se juntar a um grupo *multicast* específico. Dado que o escopo de interação do IGMP é limitado a um hospedeiro e seu roteador conectado, um outro protocolo é claramente necessário para coordenar os roteadores *multicast* (incluindo os roteadores conectados) por toda a Internet, de modo que os datagramas *multicast* sejam roteados a seus destinos finais. Essa última funcionalidade é realizada por algoritmos de roteamento *multicast* de camada de rede, como o PIM (*protocol independent multicast* — *multicast* independente de protocolo), o DVMRP (*distance vector multicast routing protocol* — protocolo de roteamento *multicast* por vetor de distância) e o MOSPF (OSPF *multicast*).

Embora o nome IGMP sugira que o protocolo gerencie o grupo de hospedeiros que se juntaram a um grupo *multicast*, ele é um pouco enganador, já que o IGMP opera *localmente* entre um hospedeiro e seu roteador conectado. A despeito de seu nome, o IGMP não é um protocolo que opera entre todos os hospedeiros que se juntaram a um grupo *multicast*. Na realidade, não há nenhum protocolo de camada de rede de associação a grupo *multicast* que opere entre todos os hospedeiros da Internet pertencentes a um grupo. Não há nenhum protocolo de rede, por exemplo, que permita a um hospedeiro determinar as identidades de todos os outros hospedeiros em toda a rede que se juntaram ao grupo *multicast*. (Veja os exercícios ao final deste capítulo para compreender melhor as conseqüências da escolha desse tipo de projeto.)

O IGMP tem apenas três tipos de mensagens. O formato da mensagem IGMP está resumido na Figura 4.46. Como as mensagens ICMP, as mensagens IGMP são carregadas (encapsuladas) dentro de um datagrama IP, com um número de protocolo igual a 2. Uma mensagem de pesquisa geral de associação ao grupo (`membership_query`) é enviada por um roteador a todos os hospedeiros em uma interface conectada (por exemplo, para todos os hospedeiros em uma LAN) para determinar o conjunto de todos os grupos *multicast* aos quais se ligaram os hospedeiros naquela interface. Um roteador pode também determinar se hospedeiros em uma interface conectada se juntaram a um grupo *multicast* específico usando uma mensagem `membership_query` específica, como mostra a Figura 4.46.

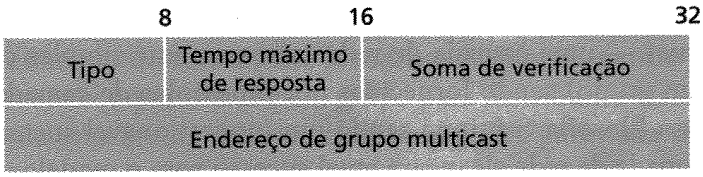


Figura 4.46 Formato da mensagem IGMP

Os hospedeiros respondem a uma mensagem `membership_query` com uma mensagem `membership_report`, como ilustra a Figura 4.47. As mensagens `membership_report` podem também ser geradas por um hospedeiro quando uma aplicação se junta pela primeira vez ao grupo *multicast* sem esperar por uma mensagem `membership_query` vinda do roteador.

O último tipo de mensagem IGMP é a mensagem `leave_group`. O interessante é que essa mensagem é opcional. Mas, se é opcional, como um roteador detecta se não há mais nenhum hospedeiro em uma interface conectada que ainda está ligado a um determinado grupo *multicast*? A resposta a essa pergunta está no uso da mensagem `membership_query` IGMP. O roteador *infere* que não há nenhum hospedeiro ligado a um dado grupo *multicast* quando nenhum hospedeiro responde a uma mensagem `membership_query` com o dado endereço de grupo. Isso é um exemplo do que às vezes é denominado **estado flexível** em um protocolo de Internet. Em um protocolo de estado flexível, o estado (no caso do IGMP, o fato de haver hospedeiros ligados a um dado grupo *multicast*) será encerrado por um evento de esgotamento de temporização (nesse caso, por uma mensagem `membership_query` periódica emitida pelo roteador) se não for explicitamente renovado (nesse caso, por uma mensagem `membership_report` vinda de um hospedeiro conectado). Argumenta-se que protocolos de estado flexível resultam em controle mais simples do que o dos protocolos de estado, que não somente exigem que o estado seja explicitamente iniciado e encerrado, mas também requerem mecanismos para se recuperar da situação em que uma entidade responsável pelo encerramento do estado encerrou prematuramente ou falhou [Sharma, 1997]. Uma discussão excelente sobre o estado flexível pode ser encontrada em [Raman, 1999; Ji, 2003]; veja também o quadro Princípios na Prática na Seção 7.9.

Agora que examinamos o protocolo para se juntar e para sair de grupos *multicast*, estamos em melhor posição para refletir sobre o atual modelo de serviço *multicast* da Internet, que é baseado no trabalho de Steve Deering [RFC 1112; Deering, 1990]. Nesse modelo de serviço *multicast*, qualquer hospedeiro pode se juntar a um grupo *multicast* na camada de rede. Um hospedeiro apenas emite uma mensagem IGMP `membership_report` a seu roteador conectado. Esse roteador, trabalhando em harmonia com outros roteadores da Internet, logo começará a entregar datagramas *multicast* ao hospedeiro. Assim, juntar-se a um grupo *multicast* é uma ação orientada para o destinatário. Um remetente não precisa se preocupar em adicionar explicitamente destinatários ao grupo *multicast*, mas, em contrapartida, não pode controlar quem se junta ao grupo e, portanto, quem recebe datagramas enviados ao grupo. De modo semelhante, em versões anteriores do IGMP, um destinatário não podia especificar um conjunto de fontes específicas das quais queria receber (ou evitar receber) pacotes *multicast*. O IGMPv3 fornece essa especificação.

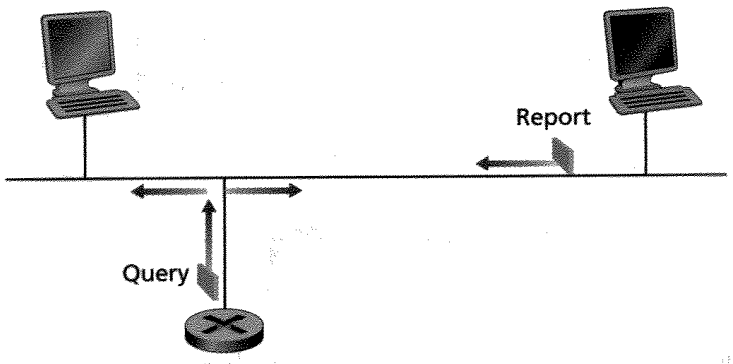


Figura 4.47 Pesquisa de associação e informação de associação IGMP

Em vários aspectos, o modelo de serviço *multicast* atual da Internet reflete a mesma filosofia do modelo de serviço *unicast* da Internet — uma camada de rede extremamente simples com funcionalidades adicionais (como associação a um grupo) sendo fornecidas por protocolos de camadas superiores dos hospedeiros nas bordas da rede. Essa filosofia continua sendo um inquestionável sucesso no caso do *unicast*. A filosofia minimalista para a camada de rede alcançará igual sucesso para o modelo de serviço *multicast*? Esta é uma pergunta que ainda não tem resposta. Vários modelos alternativos de serviços *multicast* foram propostos recentemente em [Holbrook, 1999; RFC 3569]. Uma discussão interessante sobre o atual modelo de serviço *multicast* da Internet e questões de oferecimento do serviço estão em [Diot, 2000].

Algoritmos de roteamento multicast

A Figura 4.48 ilustra o quadro para o **problema de roteamento multicast**. Consideramos um único grupo *multicast* e admitimos que qualquer roteador que tenha um hospedeiro conectado que se juntou a esse grupo possa tanto enviar quanto receber tráfego endereçado a esse grupo. Na Figura 4.48, os hospedeiros agregados a esse grupo *multicast* são os de tom mais claro; os roteadores imediatamente conectados a eles também estão nesse tom. Como mostra essa figura, dentro da população de roteadores *multicast*, somente um subconjunto desses roteadores (os que têm hospedeiros conectados que se juntaram ao grupo *multicast*) realmente precisa receber o tráfego *multicast*. Ou seja, somente os roteadores *A*, *B*, *E* e *F* precisam receber esse tráfego. Uma vez que nenhum dos hospedeiros conectados ao roteador *D* está ligado ao grupo *multicast* e o roteador *C* não tem hospedeiros conectados, nem *C* nem *D* precisam receber o tráfego do grupo *multicast*.

A meta do roteamento *multicast* é encontrar uma árvore de enlaces que conecte todos os roteadores que têm hospedeiros conectados pertencentes ao grupo *multicast*. Então, pacotes *multicast* serão roteados ao longo dessa árvore desde o remetente até todos os hospedeiros pertencentes à árvore *multicast*. Evidentemente, a árvore pode conter roteadores que não têm hospedeiros conectados pertencentes ao grupo *multicast* (por exemplo, na Figura 4.48, é impossível conectar os roteadores *A*, *B*, *E* e *F* em uma árvore sem envolver o roteador *C* e/ou *D*).

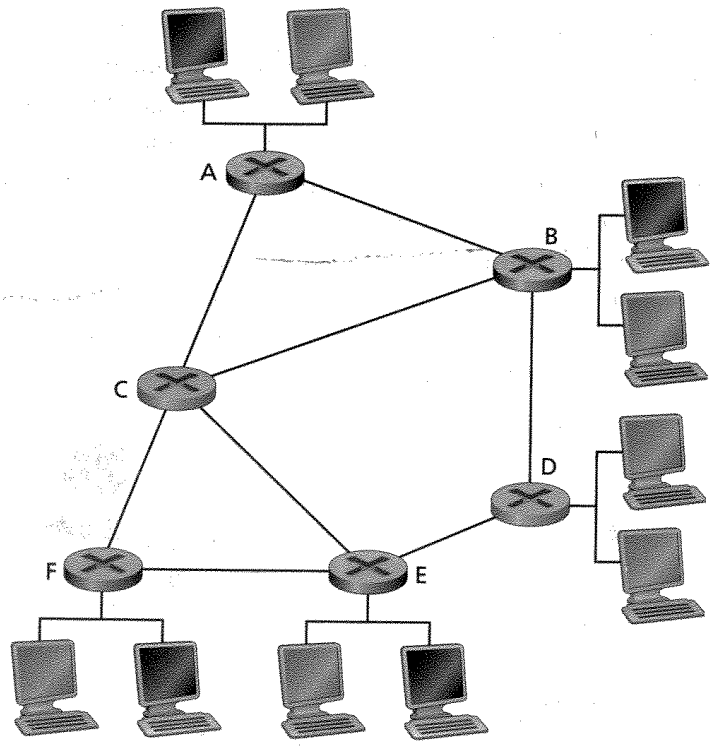


Figura 4.48 Hospedeiros multicast, seus roteadores conectados e outros roteadores

Na prática, duas abordagens foram adotadas para determinar a árvore de roteamento *multicast* e ambas já foram estudadas no contexto do roteamento *broadcast*. A diferença entre essas duas abordagens refere-se ao uso de uma única árvore compartilhada pelo grupo para distribuir o tráfego para todos os remetentes do grupo ou à construção de uma árvore de roteamento específica para cada remetente individual:

- *Roteamento multicast usando uma árvore compartilhada pelo grupo.* Como no caso do *broadcast* de *spanning tree*, o roteamento *multicast* por uma árvore compartilhada por um grupo é baseado na construção de uma árvore que inclui todos os roteadores de borda cujos hospedeiros a ele conectados pertencem ao grupo *multicast*. Na prática, é usada uma abordagem centralizada para construir a árvore de roteamento *multicast* — roteadores de aresta cujos hospedeiros a eles ligados pertencem ao grupo *multicast* enviam (via *unicast*) mensagens de adesão endereçadas ao nó central. Como no caso do *broadcast*, uma mensagem de adesão é transmitida usando roteamento *unicast* em direção ao centro até que a mensagem chegue a um roteador que já pertence à árvore *multicast* ou chegue até o centro. Todos os roteadores ao longo do caminho percorrido pela mensagem de adesão então transmitirão os pacotes *multicast* recebidos ao roteador de aresta que iniciou a adesão *multicast*. Uma questão crítica para o roteamento *multicast* baseado por árvore de centro é o processo utilizado para selecionar o centro. Algoritmos de seleção de centro são discutidos em [Wall, 1980; Thaler, 1997; Estrin, 1997].
- *Roteamento multicast usando uma árvore baseada na fonte.* Enquanto o roteamento *multicast* por árvore compartilhada pelo grupo constrói uma única arvore de roteamento compartilhada para rotear pacotes de *todos* os hospedeiros, a segunda abordagem constrói uma árvore de roteamento *multicast* para *cada fonte* no grupo *multicast*. Na prática, é usado um algoritmo RPF (com nó de fonte x) para construir uma árvore de transmissão *multicast* para fazer *multicast* de datagramas que se originam na fonte x. Para ser usado em *multicast*, o algoritmo RPF que estudamos precisa sofrer uma certa adaptação. Para ver por que, considere o roteador D na Figura 4.49. Com *broadcast* RPF, o algoritmo repassaria pacotes ao roteador G, mesmo que esse roteador não tivesse nenhum hospedeiro conectado a ele pertencente ao grupo *multicast*. Embora isso não seja tão ruim para este caso, em que D tem somente um roteador adiante dele, G, imagine o que aconteceria se houvesse milhares de roteadores adiante de D! Cada um desses milhares de roteadores receberia pacotes *multicast* indesejados. (Esse cenário não está tão longe da realidade como parece. A Mbone inicial (Casner, 1992; Macedonia, 1994), a primeira rede *multicast* global sofria precisamente desse problema no início.) A solução para o problema do recebimento de pacotes *multicast* indesejados sob RPF é conhecida como **poda**. Um roteador *multicast* que receba pacotes *multicast* e não tenha nenhum hospedeiro conectado a ele pertencente àquele grupo enviará uma mensagem de poda ao roteador que está antes dele. Se um roteador receber mensagens de poda de cada um dos roteadores que estão adiante dele, então pode repassar uma mensagem de poda aos roteadores que estão antes dele.

Roteamento multicast na Internet

O primeiro protocolo de roteamento *multicast* usado na Internet e o algoritmo de roteamento *multicast* mais amplamente suportado é o **Protocolo de Roteamento Multicast por Vetor de Distâncias** (*Distance-Vector Multicast Routing Protocol* — DVMRP) [RFC 1075]. O DVMRP implementa árvores específicas de fonte com repasse de caminho inverso e poda. O protocolo usa um algoritmo de vetor de distâncias que permite que cada roteador calcule o enlace de saída (próximo salto) que está no caminho mais curto de retorno a cada fonte possível. Essa informação é, então, usada no algoritmo RPF, como discutido antes. Uma cópia pública do software DVMRP está disponível em [mrouted, 1996].

Além do processamento da informação sobre o último salto, o DVMRP calcula uma lista de roteadores dependentes abaixo dele com a finalidade de poda. Quando um roteador receber uma mensagem de poda de todos os roteadores dependentes abaixo dele para um dado grupo, ele propagará uma mensagem de poda

para o roteador acima dele do qual ele recebe seu tráfego *multicast* para aquele grupo. Uma mensagem de poda DVMRP contém um tempo de vida de poda (com valor de default de duas horas) que indica quanto tempo um ramo podado permanecerá como tal antes de ser automaticamente restaurado. Mensagens de enxerto DVMRP são enviadas por um roteador a seu vizinho acima dele para obrigar a reintegração à árvore de *multicast* de um ramo previamente podado.

O segundo protocolo de roteamento *multicast* utilizado amplamente na Internet é o **Protocolo de Roteamento PIM** (*multicast* independente de protocolo) [Deering, 1996; RFC 2362; Estrin, 1998b], que visualiza explicitamente dois cenários diferentes de distribuição *multicast*. Em **modo denso**, a localização dos membros do grupo *multicast* é de alta densidade, isto é, muitos dos roteadores na área, ou a maioria deles, precisam ser envolvidos no roteamento de datagramas *multicast*. No **modo esparso**, o número de roteadores com membros do grupo conectados é pequeno em relação ao número total de roteadores; os membros do grupo estão amplamente dispersos. O PIM favorece essa dicotomia denso *versus* esparso, oferecendo dois modos explícitos de operação: modo denso e modo esparso. O PIM de Modo Denso é uma técnica de repasse de caminho inverso do tipo inundar e podar semelhante em espírito ao DVMRP. O PIM de Modo Esparso é uma abordagem baseada em centro semelhante à do protocolo de roteamento *multicast* de árvore com núcleo (CBT) [RFC 2201; RFC 2198]. Uma característica original do PIM é que ele nos dá a capacidade de comutar de uma árvore compartilhada por grupo para uma árvore específica de fonte após a adesão ao ponto de encontro. Uma árvore específica de fonte pode ser preferida devido à concentração menor de tráfego que ocorre quando são usadas várias dessas árvores (veja os exercícios). No PIM de Modo Esparso, o roteador que recebe um datagrama para enviar a partir de um de seus hospedeiros conectados o envia por *unicast* para o ponto de encontro. O ponto de encontro (*rendezvous point* — RP) então envia o datagrama por *multicast* por meio da árvore compartilhada pelo grupo. O RP avisa ao remetente que ele deve parar de enviar para o ponto de encontro sempre que não houver roteadores ligados àquela árvore (isto é, não houver nenhum à escuta.)

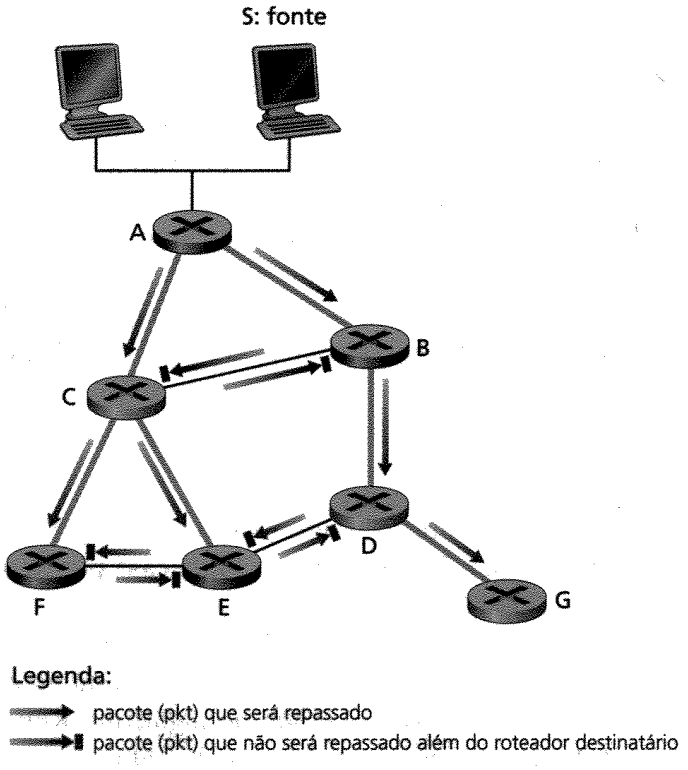


Figura 4.49 Repasse pelo caminho inverso, o caso do multicast

Em nossa discussão, admitimos que todos os roteadores rodavam o mesmo protocolo de roteamento *multicast*. Como vimos com o *unicast*, esse será normalmente o caso dentro de um único sistema autônomo (AS). Contudo, ASs diferentes podem optar por rodar protocolos de roteamento *multicast* diferentes. Foram definidas regras de interoperabilidade para os principais protocolos de roteamento *multicast* da Internet [RFC 2175]. (As regras são particularmente confusas devido às várias abordagens diferentes adotadas para rotear *multicast* por protocolos de modos esparso e denso.) O que ainda está faltando, no entanto, é um protocolo de roteamento *multicast* inter-AS para rotear datagramas *multicast* entre ASs diferentes. Até agora, o DVMRP tem servido como o protocolo de roteamento *multicast* inter-AS *de facto*.

Finalmente, mencionamos que, até agora, o *multicast* IP ainda não alcançou grande sucesso. Muitas empresas distribuidoras de conteúdo e fornecedoras de vídeo em tempo real usam sobreposição, em vez do *multicast* IP, para criar redes *multicast* de distribuição de conteúdo de camada de aplicação. Como acontece com o IPv6, o *multicast* IP vem lutando — e continua a lutar — para conquistar vitórias significativas na Internet.

4.8 Resumo

Neste capítulo, iniciamos nossa viagem rumo ao núcleo da rede. Aprendemos que a camada de rede envolve todos os roteadores da rede. Por causa disso, os protocolos de camada de rede estão entre os mais desafiadores da pilha de protocolos.

Aprendemos que um roteador pode precisar processar milhões de fluxos de pacotes entre diferentes pares fonte-destino ao mesmo tempo. Para que um roteador possa processar um número tão grande de fluxos, os projetistas de redes aprenderam, com o passar dos anos, que as tarefas do roteador devem ser o mais simples possível. Muitas medidas podem ser tomadas para facilitar a tarefa do roteador, incluindo a utilização de uma camada de rede de datagramas, em vez de uma camada de rede de circuitos virtuais, a utilização de um cabeçalho aprimorado de tamanho fixo (como no IPv6), a eliminação da fragmentação (também feita no IPv6) e o fornecimento de um único serviço de melhor esforço. Talvez a sutileza mais importante aqui seja não monitorar fluxos individuais, mas, em vez disso, tomar decisões de roteamento tendo exclusivamente como base endereços de destino hierarquicamente estruturados nos pacotes. É interessante notar que o serviço dos correios vem usando essa mesma abordagem há muitos anos.

Neste capítulo, examinamos também os princípios subjacentes de algoritmos de roteamento. Aprendemos como algoritmos de roteamento fazem a abstração da rede de computadores em um grafo com nós e enlaces. Com essa abstração, podemos pesquisar a rica teoria do roteamento de caminho mais curto em grafos, que foi desenvolvida nos últimos 40 anos nas comunidades de pesquisa operacional e de algoritmos. Vimos que há duas abordagens amplas: uma centralizada (global), em que cada nó obtém o mapeamento completo da rede e aplica independentemente o algoritmo de roteamento de caminho mais curto, e uma descentralizada, em que nós individuais têm apenas um quadro parcial da rede inteira e, mesmo assim, trabalham em conjunto para entregar pacotes ao longo das rotas mais curtas. Também estudamos como a hierarquia é utilizada para lidar com o problema da escala, dividindo redes de grande porte em domínios administrativos independentes denominados sistemas autônomos (ASs). Cada AS roteia independentemente seus datagramas pelo sistema, exatamente como cada país distribui sua correspondência postal pelo seu território. Aprendemos como abordagens centralizadas, descentralizadas e hierárquicas estão incorporadas nos principais protocolos de roteamento da Internet: RIP, OSPF e BGP. Concluímos nosso estudo de algoritmos de roteamento considerando roteamento *broadcast* e *multicast*.

Agora que concluímos nosso estudo da camada de rede, nossa jornada segue rumo a um nível mais baixo da pilha de protocolos, isto é, à camada de enlace. Como a camada de rede, a camada de enlace é também parte do núcleo da rede. Mas veremos no próximo capítulo que a camada de enlace tem a tarefa muito mais localizada de movimentar pacotes entre nós no mesmo enlace ou LAN. Embora essa tarefa possa à primeira vista parecer trivial quando comparada às tarefas da camada de rede, veremos que a camada de enlace envolve uma série de questões importantes e fascinantes que podem nos manter ocupados por muito tempo.

Exercícios de fixação

Capítulo 4 Questões de revisão

Seções 4.1 a 4.2

1. Vamos rever um pouco da terminologia usada neste livro. Lembre-se de que o nome de um pacote de camada de transporte é *segmento* e que o nome de um pacote de camada de enlace é *quadro*. Qual é o nome de um pacote de camada de rede? Lembre-se de que roteadores e comutadores de camada de enlace são denominados *comutadores de pacotes*. Qual é a diferença fundamental entre um roteador e um comutador de camada de enlace? Lembre-se de que usamos o termo *roteadores* tanto para redes de datagramas quanto para redes de CVs.
2. Quais são as duas funções mais importantes de camada de rede em uma rede de datagramas? Quais são as três funções mais importantes de camada de rede em uma rede com circuitos virtuais?
3. Qual é a diferença entre rotear e repassar (transmitir)?
4. Os roteadores de redes de datagramas e de redes de circuitos virtuais usam tabelas de repasse? Caso usem, descreva as tabelas de repasse para ambas as classes de redes.
5. Descreva alguns serviços hipotéticos que a camada de rede poderia oferecer a um pacote individual. Faça o mesmo para um fluxo de pacotes. Alguns dos serviços hipotéticos que você descreveu são fornecidos pela camada de rede da Internet? Alguns deles são fornecidos pelo modelo de serviço ATM CBR? Alguns são fornecidos pelo modelo de serviço ATM ABR?
6. Cite algumas aplicações que poderiam se beneficiar do modelo de serviço ATM CBR.

Seção 4.3

7. Discuta por que cada porta de entrada em um roteador de alta velocidade armazena uma cópia-sombra da tabela de repasse.
8. Três tipos de elementos de comutação são discutidos na Seção 4.3. Cite e descreva brevemente cada tipo.
9. Descreva como pode ocorrer perda de pacotes em portas de entrada. Descreva como a perda de pacotes pode ser eliminada em portas de entrada (sem usar buffers infinitos).
10. Descreva como pode ocorrer perda de pacotes em portas de saída.
11. O que é bloqueio HOL? Ele ocorre em portas de saída ou em portas de entrada?

Seção 4.4

12. Roteadores têm endereços IP? Em caso positivo, quantos endereços eles têm?
13. Qual é o equivalente binário de 32 bits para o endereço IP 223.1.3.27?
14. Visite um hospedeiro que usa DHCP para obter seu endereço IP, máscara de rede, roteador de default e endereço IP de seu servidor DNS local. Faça uma lista desses valores.
15. Suponha que haja três roteadores entre os hospedeiros da fonte e do destino. Ignorando a fragmentação, um datagrama IP enviado do hospedeiro da fonte até o hospedeiro do destino transitará por quantas interfaces? Quantas tabelas de repasse serão indexadas para deslocar o datagrama desde a fonte até o destino?
16. Suponha que uma aplicação gere blocos de 40 bytes de dados a cada 20 milissegundos e que cada bloco seja encapsulado em um segmento TCP e, em seguida, em um datagrama IP. Que porcentagem de cada datagrama será sobrecarga e que porcentagem será dados de aplicação?
17. Suponha que o Hospedeiro A envie ao Hospedeiro B um segmento TCP encapsulado em um datagrama IP. Quando o Hospedeiro B recebe o datagrama, como a camada de rede no Hospedeiro B sabe que deve passar o segmento (isto é, a carga útil do datagrama) para TCP e não para UDP ou qualquer outra coisa?

18. Suponha que você compre um roteador sem fio e o conecte a seu modem a cabo. Suponha também que seu ISP designe dinamicamente um endereço IP a seu dispositivo conectado (isto é, seu roteador sem fio). Suponha ainda que você tenha cinco PCs em casa e que usa 802.11 para conectá-los sem fio ao seu roteador também sem fio. Como são designados endereços IP aos cinco PCs? O roteador sem fio usa NAT?

19. Compare os campos de cabeçalho do IPv4 e do IPv6 e aponte suas diferenças. Eles têm algum campo em comum?

20. Afirma-se que, quando o IPv6 implementa túneis através de roteadores IPv4, o IPv6 trata os túneis IPv4 como protocolos de camada de enlace. Você concorda com essa afirmação? Explique sua resposta.

Seção 4.5

21. Compare e aponte as diferenças entre os algoritmos de estado de enlace e de vetor de distâncias.

22. Discuta como uma organização hierárquica da Internet possibilitou estender sua escala para milhões de usuários.

23. É necessário que todo sistema autônomo use o mesmo algoritmo de roteamento intra-AS? Justifique sua resposta.

Seção 4.6

24. Considere a Figura 4.31. Começando com a tabela original em *D*, suponha que *D* receba de *A* o seguinte anúncio:

Sub-rede de destino	Roteador seguinte	Número de saltos até o destino
z	C	10
w	–	1
x	–	1
....	....	....

A tabela em *D* mudará? Em caso positivo, como mudará?

25. Compare os anúncios utilizados por RIP e OSPF e aponte suas diferenças.

26. Complete: anúncios RIP normalmente anunciam o número de saltos até vários destinos. Atualizações BGP, por outro lado, anunciam _____ aos diversos destinos.

27. Por que são usados protocolos inter-AS e intra-AS diferentes na Internet?

28. Por que considerações políticas não são tão importantes para protocolos intra-AS como o OSPF e o RIP, quanto para um protocolo de roteamento inter-AS como BGP?

29. Defina e aponte as diferenças entre os seguintes termos: *sub-rede*, *prefixo* e *rota BGP*.

30. Como o BGP usa o atributo NEXT-HOP? Como ele usa o atributo AS-PATH?

31. Descreva como um administrador de rede de um ISP de nível superior pode implementar política ao configurar o BGP.

Seção 4.7

32. Cite uma importante diferença entre a implementação da abstração *broadcast* por múltiplos *unicasts* e a de um grupo *broadcast* suportado por uma única rede (roteador).

33. Para cada uma das três abordagens gerais que estudamos para a comunicação *broadcast* (inundação não controlada, inundação controlada e *broadcast* de spanning tree), as seguintes declarações são verdadeiras ou falsas? Você pode admitir que não há perda de pacotes devido ao transbordamento do buffer e que todos os pacotes são entregues em um enlace na ordem em que foram enviados.

a. Um nó pode receber várias cópias do mesmo pacote.

b. Um nó pode repassar várias cópias de um pacote pelo mesmo enlace de saída.

34. Quando um hospedeiro se junta a um grupo *multicast*, ele deve mudar seu endereço IP para o endereço do grupo *multicast* ao qual está se juntando?
35. Quais são os papéis desempenhados pelo protocolo IGMP e por um protocolo de roteamento *multicast* de longa distância?
36. Qual é a diferença entre uma árvore compartilhada por um grupo e uma árvore de fonte no contexto do roteamento *multicast*?

Problemas

1. Considere alguns dos prós e dos contras de redes de circuitos virtuais e redes de datagramas.
   a. Suponha que, na camada de rede, os roteadores estivessem sujeitos a condições estressantes que talvez os tenham feito falhar com bastante freqüência. Que ações deveriam ser executadas no alto nível para sanar essa falha do roteador? Essas ações favorecem uma arquitetura de circuitos virtuais ou de datagramas?
   b. Suponha que, para fornecer uma garantia do nível de desempenho (por exemplo, atraso) que seria observado ao longo do caminho fonte-destino, a rede exija que um remetente declare sua taxa de pico de tráfego. Se a taxa de pico de tráfego declarada e as taxas de pico declaradas existentes forem tais que não há nenhum modo de levar o tráfego da fonte ao destino que satisfaça as exigências quanto aos atrasos, o acesso à rede é negado a essa fonte. Essa abordagem seria mais facilmente realizada dentro de uma arquitetura de CVs ou de datagramas?
2. Considere uma rede de circuitos virtuais. Suponha que o número do CV é um campo de 16 bits.
   a. Qual é o número máximo de circuitos virtuais que pode ser carregado por um enlace?
   b. Suponha que um nó central determine caminhos e números de CVs no estabelecimento da conexão. Suponha que o mesmo número de CV seja usado em cada enlace ao longo do caminho do CV. Descreva como o nó central poderia determinar o número do CV no estabelecimento da conexão. É possível haver um número de CVs ativos menor do que o máximo determinado na parte (a) e, mesmo assim, não haver nenhum número de CV livre em comum?
   c. Suponha que sejam permitidos números diferentes de CVs em cada enlace ao longo do caminho de um CV. Durante o estabelecimento da conexão, após a determinação de um caminho fim-a-fim, descreva como os enlaces podem escolher seus números de CVs e configurar suas tabelas de repasse de uma maneira descentralizada, sem depender de um nó central.
3. Uma tabela de repasse bem básica em uma rede de CVs tem quatro colunas. O que significam os valores em cada uma dessas colunas? Uma tabela de repasse bem básica em uma rede de datagramas tem duas colunas. O que significam os valores em cada uma dessas colunas?
4. Considere uma rede de CVs cujo campo de número de CV tenha 2 bits. Suponha que a rede queira estabelecer um circuito virtual por quatro enlaces: enlace A, enlace B, enlace C e enlace D. Suponha que, nesse momento, cada um desses enlaces esteja carregando dois outros circuitos virtuais e que os números desses outros CVs são os seguintes:

Enlace A	Enlace B	Enlace C	Enlace D
00	01	10	11
01	10	11	00

Ao responder as perguntas a seguir, tenha em mente que cada um dos CVs existentes pode estar atravessando apenas um dos quatro enlaces.

a. Se cada CV tiver de usar o mesmo número de CV em todos os enlaces ao longo de seu caminho, qual número de CV poderia ser designado ao novo CV?

b. Se fosse permitido que cada CV tivesse um número de CV diferente nos diferentes enlaces ao longo de seu caminho (de modo que a tabela de repasse tenha de realizar tradução de número de CV), quantas combinações diferentes de quatro números de CVs (um para cada um dos quatro enlaces) poderiam ser usadas?

5. No texto usamos o termo *serviço orientado para conexão* para descrever a camada de transporte e *serviço de conexão* para a camada de rede. Por que essa sutileza na terminologia?

6. Na Seção 4.3 observamos que não pode haver nenhuma fila de entrada se o elemento de comutação for n vezes mais rápido do que as taxas das linhas de entrada, admitindo que n linhas de entrada tenham todas a mesma velocidade de linha. Explique (com palavras) por que isso tem de ser assim.

7. Considere uma rede de datagramas que usa endereços de hospedeiros de 32 bits. Suponha que um roteador tenha quatro enlaces, numerados de 0 a 3, e que os pacotes têm de ser repassados para as interfaces de enlaces como segue:

Faixa do endereço de destino	**Interface de enlace**
11100000 00000000 00000000 00000000 até 11100000 11111111 11111111 11111111	0
11100001 00000000 00000000 00000000 até 11100001 00000000 11111111 11111111	1
11100001 00000001 00000000 00000000 até 11100001 11111111 11111111 11111111	2
senão	3

a. Elabore uma tabela de repasse que tenha quatro registros, use compatibilização com o prefixo mais longo e repasse pacotes para as interfaces de enlace corretas.

b. Descreva como sua tabela de repasse determina a interface de enlace apropriada para datagramas com os seguintes endereços:

11001000 10010001 01010001 01010101
11100001 00000000 11000011 00111100
11100001 10000000 00010001 01110111

8. Considere uma rede de datagramas que utiliza endereços de 8 bits. Suponha que um roteador utilize compatibilização com o prefixo mais longo e tenha a seguinte tabela de repasse:

Prefixo a ser comparado	Interface
00	0
01	1
10	2
11	3

Para cada uma das quatro interfaces, forneça a faixa associada de endereços de hospedeiros de destino e o número de endereços na faixa.

9. Considere uma rede de datagramas que usa endereços de hospedeiros de 8 bits. Suponha que um roteador use compatibilização com o prefixo mais longo e tenha a seguinte tabela de repasse:

Prefixo a ser comparado	Interface
1	0
11	1
111	2
senão	3

Para cada uma das quatro interfaces, forneça a faixa associada de endereços de hospedeiros de destino e o número de endereços na faixa.

10. Considere um roteador que interconecta três sub-redes: Sub-rede 1, Sub-rede 2 e Sub-rede 3. Suponha que todas as interfaces de cada uma dessas três sub-redes tenha de ter o prefixo 223.1.17/24. Suponha também que a Sub-rede 1 tenha de suportar até 125 interfaces, e que cada uma das Sub-redes 2 e 3 tenha de suportar até 60 interfaces. Dê três endereços de rede (da forma a.b.c.d/x) que satisfaçam essas limitações.

11. Na Seção 4.2.2 é dado um exemplo de tabela de repasse (usando compatibilização de prefixo mais longo). Reescreva a tabela usando a notação a.b.c.d/x em vez da notação de cadeia binária.

12. No Problema 7, solicitamos que você elaborasse uma tabela de repasse (usando compatilização de prefixo mais longo). Reescreva a tabela usando a notação a.b.c.d/x em vez da notação de cadeia binária.

13. Considere uma sub-rede com prefixo 101.101.101.64/26. Dê um exemplo de um endereço IP (na forma xxx.xxx.xxx.xxx) que pode ser designado para essa rede. Suponha que um ISP possua o bloco de endereços na forma 101.101.128/17. Suponha que ele queira criar quatro sub-redes a partir desse bloco, e que cada bloco tenha o mesmo número de endereços IP. Quais são os prefixos (na forma a.b.c.d/x) para essas quatro sub-redes?

14. Considere a topologia mostrada na Figura 4.17. Começando pela parte superior e prosseguindo em sentido horário, denomine as três sub-redes como Redes A, B, e C. Denomine as sub-redes sem hospedeiros como Redes D, E, e F.

a. Designe endereços de rede a cada uma dessas seis sub-redes, com as seguintes restrições: todos os endereços deverão ser alocados a partir de 214.97.254/23; a Sub-rede A deve ter endereços suficientes para suportar 250 interfaces; a Sub-rede B deve ter endereços suficientes para suportar 120 interfaces e a Sub-rede C deve ter endereços suficientes para suportar 120 interfaces. É claro que cada uma das sub-redes D, E e F devem poder suportar duas interfaces. Para cada sub-rede, a designação deve tomar a forma a.b.c.d/x ou a.b.c.d/x – e.f.g.h/y.

b. Usando a resposta que você deu para a parte (a), elabore as tabelas de repasse (usando compatibilização de prefixo mais longo) para cada um dos três roteadores.

15. Considere enviar um datagrama de 3.000 bytes por um enlace que tem uma MTU de 500 bytes. Suponha que o datagrama original esteja marcado com o número de identificação 422. Quantos fragmentos são gerados? Quais são suas características?

16. Suponha que, entre o Hospedeiro de origem A e o Hospedeiro destinatário B, os datagramas estejam limitados a 1.500 bytes (incluindo cabeçalho). Admitindo um cabeçalho IP de 20 bytes, quantos datagramas seriam necessários para enviar um arquivo MP3 de 4 milhões de bytes?

17. Considere a configuração de rede da Figura 4.20. Suponha que o ISP designe ao roteador o endereço 126.13.89.67 e que o endereço de rede da rede residencial seja 192.168/16.

a. Designe endereços a todas as interfaces na rede residencial.

b. Suponha que haja duas conexões TCP em curso em cada hospedeiro, todas para a porta 80 no hospedeiro 128.119.40.86. Forneça os seis registros correspondentes na tabela de tradução NAT.

18. Neste problema estudaremos o impacto das NATs sobre aplicações P2P. Suponha que um parceiro com nome de usuário Arnold descubra, por meio de consulta, que um parceiro com nome de hospedeiro Bernard tem um arquivo que ele, Arnold, quer descarregar. Suponha também que Bernard esteja por trás de uma NAT, mas Arnold, não. Seja 138.76.29.7 o endereço da NAT do lado da WAN e seja 10.0.0.1 o endereço IP interno para Bernard. Admita que a NAT não está especificamente configurada para a aplicação P2P.

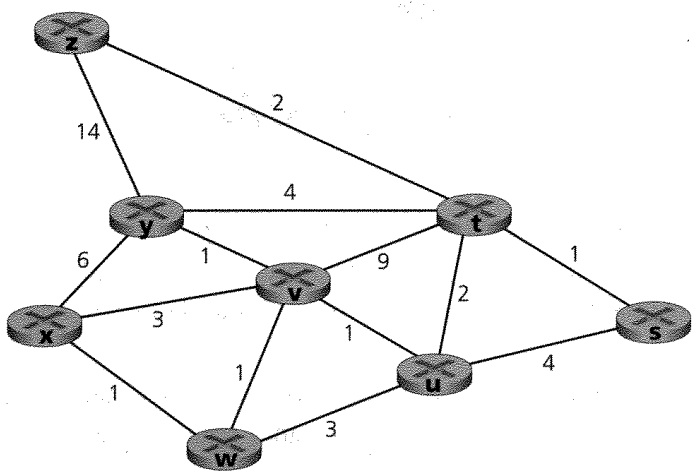
a. Discuta por que o Arnold não pode iniciar uma conexão TCP com o Bernard mesmo que Arnold conheça o endereço da NAT do lado da WAN, 138.76.29.7.

b. Agora suponha que Bernard tenha estabelecido uma conexão TCP em curso com um outro parceiro, Cindy, que não está atrás de uma NAT. Suponha também que Arnold soubesse, por Cindy, que Bernard tem o arquivo desejado e que ele, Arnold, pode estabelecer (ou já estabeleceu) uma conexão TCP com Cindy. Descreva como Arnold pode usar essas duas conexões TCP (uma de Bernard para Cindy e a outra de Arnold para Cindy) para instruir Bernard a iniciar uma conexão TCP direta com Arnold (isto é, que não esteja passando por Cindy). Essa técnica costuma ser denominada *reversão de conexão*. Note que, mesmo que Bernard esteja por trás de uma NAT, Arnold pode usar essa conexão TCP direta para requisitar o arquivo e Bernard pode usar a conexão para entregar o arquivo.

19. Prosseguindo com o problema anterior, suponha agora que ambos, Arnold e Bernard, estejam por trás de NATs. Tente imaginar uma técnica que permitirá a Arnold estabelecer uma conexão TCP com Bernard sem configuração de NAT específica para aplicação. Se tiver dificuldades para elaborar essa técnica, discuta por quê.

20. Considerando a Figura 4.25, enumere os caminhos de *u* a *z* que não tenham nenhum laço.

21. Considere a seguinte rede. Com os custos de enlace indicados, use o algoritmo do caminho mais curto de Dijkstra para calcular o caminho mais curto de *x* até todos os nós da rede. Mostre como o algoritmo funciona calculando uma tabela semelhante à Tabela 4.3.



22. Considere a rede mostrada no Problema 21. Usando o algoritmo de Dijkstra e mostrando seu trabalho usando uma tabela semelhante à Tabela 4.3, faça o seguinte:

a. Calcule o caminho mais curto de *s* até todos os nós da rede.

b. Calcule o caminho mais curto de *t* até todos os nós da rede.

c. Calcule o caminho mais curto de *u* até todos os nós da rede.

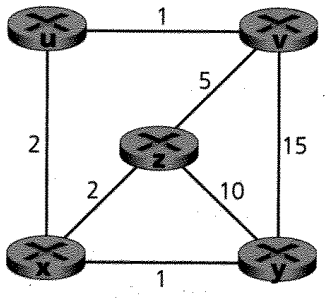
d. Calcule o caminho mais curto de *v* até todos os nós da rede.

e. Calcule o caminho mais curto de w até todos os nós da rede.

f. Calcule o caminho mais curto de y até todos os nós da rede.

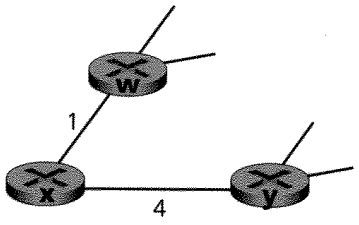
g. Calcule o caminho mais curto de z até todos os nós da rede.

23. Considere a rede mostrada a seguir e admita que cada nó inicialmente conheça os custos até cada um de seus vizinhos. Considere o algoritmo de vetor de distâncias e mostre os registros na tabela de distâncias para o nó z.



24. Considere uma topologia geral (isto é, não a rede específica mostrada anteriormente) e uma versão síncrona do algoritmo de vetor de distâncias. Suponha que, a cada iteração, um nó troque seus vetores de distâncias com seus vizinhos e receba os vetores de distâncias deles. Supondo que o algoritmo comece com cada nó conhecendo apenas os custos até seus vizinhos imediatos, qual é o número máximo de iterações requeridas até que o algoritmo distribuído convirja? Justifique sua resposta.

25. Considere o fragmento de rede mostrado a seguir. x tem apenas dois vizinhos ligados a ele: w e y. w tem um caminho de custo mínimo até o destino u (não mostrado) de 5 e y tem um caminho de custo mínimo u de 6. Os caminhos completos de w e de y até u (e entre w e y) não são mostrados. Todos os valores dos custos de enlace na rede são números inteiros estritamente positivos.



a. Dê os vetores de distâncias de x para os destinos w, y e u.

b. Dê uma mudança de custo de enlace para $c(x,w)$ ou para $c(x,y)$ tal que x informará a seus vizinhos um novo caminho de custo mínimo até u como resultado da execução do algoritmo de vetor de distâncias.

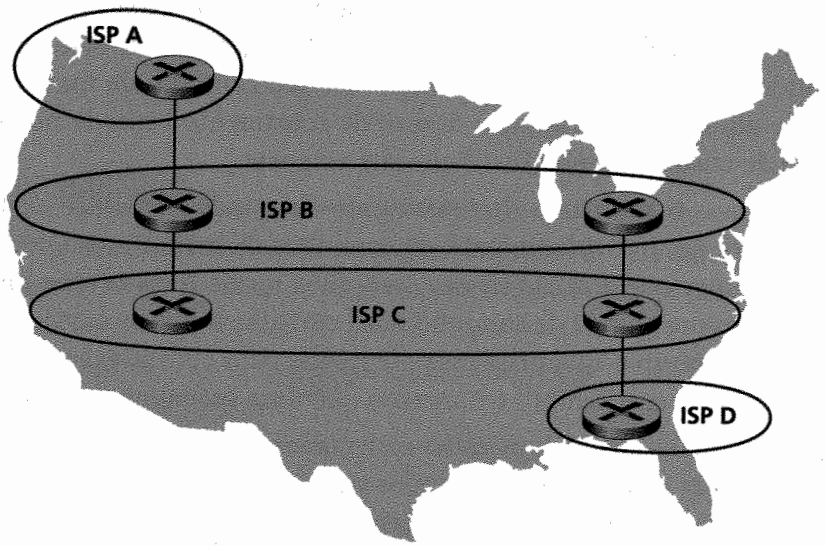
c. Dê uma mudança de custo de enlace para $c(x,w)$ ou para $c(x,y)$ tal que x não informará a seus vizinhos um novo caminho de custo mínimo até u como resultado da execução do algoritmo de vetor de distâncias.

26. Considere a topologia de três nós mostrada na Figura 4.27. Em vez de ter os custos de enlace da Figura 4.27, os custos de enlace são: $c(x,y) = 5$, $c(y,z) = 6$, $c(z,x) = 2$. Calcule as tabelas de distâncias após a etapa de inicialização e após cada iteração de uma versão síncrona do algoritmo de vetor de distâncias (como fizemos em nossa discussão anterior da Figura 4.27).

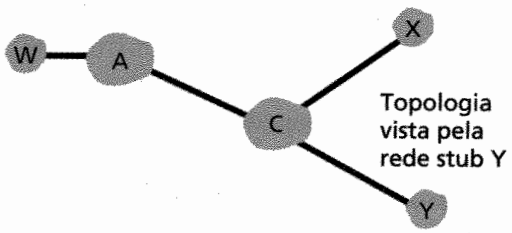
27. Descreva como laços no caminho podem ser detectados com BGP.

28. Considere a seguinte rede. O ISP B provê serviço nacional de backbone ao ISP regional A. O ISP C provê serviço nacional de backbone ao ISP regional D. Usando BGP, B e C formam pares entre si em dois luga-

res. Considere tráfego na direção de A a D. B preferiria passar esse tráfego para C na Costa Oeste (de modo que C teria de absorver o custo de carregar o tráfego através do país), enquanto C preferiria receber o tráfego via seu ponto de formação de par com B na Costa Leste (de modo que B carregaria o tráfego através do país). Qual mecanismo BGP C poderia usar de modo que B entregasse o tráfego de A a D em seu ponto de formação de par na Costa Leste? Para responder a essa pergunta, você precisará estudar muito bem a especificação do BGP.



29. Na Figura 4.39, considere a informação de caminho que chega às sub-redes stub W, X e Y. Com base na informação disponível em W e X, quais são as respectivas visões da topologia da rede? Justifique sua resposta. A topologia vista de Y é mostrada a seguir:



30. Considere a rede de oito nós (com nós rotulados de *s* a *z*) do Problema 21. Mostre a árvore de custo mínimo com raiz em *s* que inclua (como hospedeiros finais) os nós *u*, *v*, *w* e *y*. Justifique informalmente por que sua árvore é uma árvore de custo mínimo.

31. Considere as duas abordagens básicas identificadas para fazer *broadcast*: emulação de *unicast* e *broadcast* de camada de rede (isto é, assistido por roteador) e suponha que seja usado *broadcast* de *spanning tree* para fazer *broadcast* de camada de rede. Considere um único remetente e 32 destinatários. Suponha que o remetente esteja conectado aos destinatários por uma árvore binária de roteadores. Qual é o custo para enviar um pacote *broadcast* nos casos da emulação de *unicast* e de *broadcast* de camada de rede para essa topologia? Aqui, cada vez que um pacote (ou cópia de um pacote) é enviado por um único enlace, ele incorre em uma unidade de custo. Qual topologia para interconectar o remetente, os destinatários e os roteadores fará com que os custos da emulação de *unicast* e do *broadcast* verdadeiro de camada de rede fiquem o mais longe possível um do outro? Você pode escolher quantos roteadores quiser.

32. Considere a operação do algoritmo de repasse de caminho inverso (RPF) na Figura 4.41. Usando a mesma topologia, descubra um conjunto de caminhos de todos os nós até o nó de fonte *A* (e indique esses caminhos em um grafo usando linhas grossas como as da Figura 4.4.1 de modo que, se esses cami-

nhos forem os caminhos de menor custo, então o nó *B* receberia uma cópia das mensagens de *broadcast* de *A* dos nós *A*, *C* e *D* sob RPF.

33. Considere a topologia ilustrada na Figura 4.41. Suponha que todos os enlaces tenham custo unitário e que o nó *E* é a fonte de *broadcast*. Usando setas como as mostradas na Figura 4.41, indique enlaces pelos quais pacotes serão repassados usando RPF e enlaces pelos quais pacotes não serão repassados, dado que o nó *E* é a fonte.

34. Considere a topologia mostrada na Figura 4.43 e suponha que cada enlace tenha preço unitário. Suponha que o nó *C* seja escolhido como o centro de um algoritmo de roteamento *multicast* baseado em centro. Admitindo que cada roteador conectado use seu caminho de menor custo até o nó *C* para enviar mensagens de adesão a *C*, desenhe a árvore de roteamento baseada no centro resultante. Essa árvore é uma árvore de custo mínimo? Justifique sua resposta.

35. Na Seção 4.5.1 estudamos o algoritmo de roteamento de estado de enlace de Dijkstra para calcular os caminhos *unicast* que são, individualmente, os caminhos de menor custo da fonte até todos os destinos. Poderíamos imaginar que a união desses caminhos forme uma **árvore *unicast* de caminho de menor custo** (ou uma árvore *unicast* de caminho mais curto, se todos os custos de enlaces fossem idênticos). Construindo um exemplo que nega essa afirmação (um contra-exemplo), mostre que a árvore de caminho de menor custo *nem* sempre é o mesmo que uma *spanning tree* mínima.

36. Considere uma rede na qual todos os nós estão conectados a três outros nós. Em uma única etapa de tempo, um nó pode receber de seus vizinhos todos os pacotes transmitidos por *broadcast*, duplicar os pacotes e enviá-los a todos os seus vizinhos (exceto ao nó que enviou um dado pacote). Na próxima etapa, nós vizinhos podem receber, duplicar e repassar esses pacotes e assim por diante. Suponha que seja utilizada a inundação não controlada para prover *broadcast* a essa rede. Na etapa de tempo *t*, quantas cópias do pacote *broadcast* serão transmitidas, admitindo que, durante a etapa 1, um único pacote *broadcast* é transmitido pelo nó da fonte a seus três vizinhos?

37. Vimos na Secão 4.7 que não há nenhum protocolo de camada de rede que possa ser usado para identificar os hospedeiros que participam de um grupo *multicast*. Isso posto, como aplicações *multicast* podem aprender as identidades dos hospedeiros que estão participando de um grupo *multicast*?

38. Projete (dê uma descrição fictícia em pseudocódigo) um protocolo de nível de aplicação que mantenha os endereços de hospedeiros para todos os hospedeiros participantes de um grupo *multicast*. Identifique especificamente o serviço de rede (*unicast* ou *multicast*) que é usado por seu protocolo e indique se seu protocolo está enviando mensagens dentro da banda ou fora da banda (com relação ao fluxo de dados de aplicação entre os participantes do grupo *multicast*) e por quê.

39. Qual é o tamanho do espaço de endereço *multicast*? Suponha agora que dois grupos *multicast* escolham aleatoriamente um endereço *multicast*. Qual é a probabilidade de que escolham o mesmo endereço? Suponha agora que mil grupos *multicast* estejam em operação ao mesmo tempo e escolham seus endereços de grupo *multicast* aleatoriamente. Qual é a probabilidade de que uns interfiram nos outros?

Questões dissertativas

1. Descubra três empresas que estão vendendo roteadores de alta velocidade. Compare os produtos.

2. Use o serviço whois da American Registry for Internet Numbers (http://www.arin.net/whois) para determinar os blocos de endereços IP para três universidades. Os serviços whois podem ser usados para determinar com certeza a localização de um endereço IP específico?

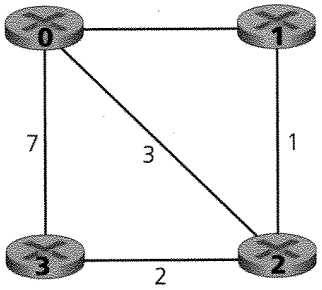
3. É possível escrever o programa ping cliente (usando mensagens ICMP) em Java? Justifique sua resposta.

4. Na Seção 4.4, dissemos que a disponibilização do IPv6 tem sido lenta. Por quê? O que é necessário para acelerar sua disponibilização?

5. Discuta alguns dos problemas que as NATs criam para a segurança Ipsec (veja [Phifer, 2001]).
6. Suponha que os ASs X e Z não estejam diretamente conectados, mas, em vez disso, estejam conectados pelo AS Y. Suponha também que X tenha um acordo de interligação com Y e que Y tenha um acordo de interligação com Z. Finalmente, suponha que Z queira carregar todo o tráfego de Y, mas não queira carregar o tráfego de X. O BGP permite que Z implemente essa política?
7. Na Seção 4.7, identificamos uma série de aplicações *multicast*. Quais dessas aplicações se ajustam bem ao modelo minimalista de serviço *multicast* da Internet? Por quê? Quais aplicações não se ajustam particularmente bem a esse modelo de serviço?

Tarefas de programação

Nesta tarefa de programação, você escreverá um conjunto 'distribuído' de procedimentos que implementam um roteamento de vetor de distâncias assíncrono distribuído para a rede mostrada a seguir.



Você deve escrever as seguintes rotinas que "rodarão" assincronamente dentro do ambiente emulado fornecido para essa tarefa. Para o nó 0, você escreverá as rotinas:

- *rtinit0()*. Essa rotina será chamada uma vez no início da emulação. *rtinit0()* não tem argumentos. Você deve inicializar sua tabela de distâncias no nó 0 para refletir os custos diretos de 1, 3 e 7 até os nós 1, 2 e 3, respectivamente. Na figura anterior, todos os enlaces são bidirecionais e os custos em ambas as direções são idênticos. Após inicializar a tabela de distâncias e quaisquer outras estruturas de dados necessárias às rotinas de seu nó 0, este deve então enviar a seus vizinhos diretamente ligados (nesse caso, 1, 2 e 3) o custo de seus caminhos de custo mínimo para todos os outros nós da rede. Essa informação do custo mínimo é enviada aos nós vizinhos em um pacote de atualização de roteamento denominado rotina *tolayer2()*, como descrita na tarefa completa. O formato do pacote de atualização de roteamento também está descrito na tarefa completa.
- *rtupdate0(struct rtpkt *rcvdpkt)*. Essa rotina será chamada quando o nó 0 receber um pacote de roteamento que foi enviado a ele por um de seus vizinhos diretamente conectados. O parâmetro **rcvdpkt* é um indicador para o pacote que foi recebido. *rtupdate0()* é o coração do algoritmo de vetor de distâncias. Os valores que ele recebe em um pacote de atualização de roteamento de algum outro nó *i* contêm os custos correntes do caminho mais curto de *i* para todos os outros nós da rede. *rtupdate0()* usa esses valores recebidos para atualizar sua própria tabela de distâncias (como especificada pelo algoritmo de vetor de distâncias). Se seu próprio custo mínimo até um outro nó mudar como resultado da atualização, o nó 0 informará essa mudança no custo mínimo a seus vizinhos diretamente conectados enviando a eles um pacote de roteamento. Lembre-se de que no algoritmo de vetor de distâncias apenas os nós conectados diretamente trocarão pacotes de roteamento. Assim, os nós 1 e 2 vão se comunicar, mas os nós 1 e 3, não.

Rotinas semelhantes são definidas para os nós 1, 2 e 3. Assim, você escreverá oito procedimentos ao todo: *rtinit0()*, *rtinit1()*, *rtinit2()*, *rtinit3()*, *rtupdate0()*, *rtupdate1()*, *rtupdate2()*, *rtupdate3()*. Juntas, essas rotinas implementarão um cálculo assíncrono, distribuído, das tabelas de distâncias para a topologia e os custos mostrados na figura apresentada anteriormente.

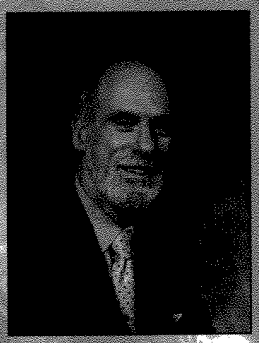
No site `http://www.aw.com/kurose_br` *você poderá encontrar todos os detalhes da tarefa de programação, bem como o código em C de que precisará para criar o ambiente hardware/software simulado. Também está disponível uma versão da tarefa em Java.*

Ethereal Lab

No Companion Website deste livro, `www.aw.com/kurose_br`, você encontrará duas tarefas de laboratório Ethereal, em inglês. A primeira examina a operação do protocolo IP e, em particular, o formato do datagrama IP. A segunda explora a utilização do protocolo ICMP nos comandos ping e traceroute.

Entrevista

Vinton G. Cerf



Vinton G. Serf é vice-presidente sênior de Arquitetura e Tecnologia da WorldCom. É muito conhecido pela co-autoria dos protocolos TCP/IP e da arquitetura da Internet. Como vice-presidente da MCI Digital Information Services de 1982 a 1986, liderou a engenharia do MCI Mail, o primeiro serviço comercial de e-mail a ser conectado à Internet. Durante sua gestão de 1976 a 1982 na Advanced Research Projects Agency do Departamento de Defesa dos Estados Unidos (DARPA), desempenhou um papel fundamental na liderança do desenvolvimento de pacotes de dados da Internet e relacionados com a Internet e de tecnologias de segurança. Vinton é bacharel em Matemática pela Stanford University e Doutor em ciência da computação pela UCLA.

O que o fez se decidir pela especialização em rede?

Eu trabalhava como programador na UCLA no final da década de 1960 com o patrocínio da Advanced Research Projects Agency do Departamento de Defesa dos Estados Unidos (na época conhecida como ARPA e hoje, como DARPA). Meu trabalho era desenvolvido no laboratório do professor Leonard Kleinrock no Network Measurement Center da recém-criada ARPANET. O primeiro nó da ARPANET foi instalado na UCLA no dia 1º de setembro de 1969. Eu era responsável pela programação de um computador utilizado para coletar informações de desempenho da ARPANET e passar essas informações para comparação com modelos matemáticos e previsões de desempenho da rede.

Eu e vários outros estudantes de pós-graduação éramos responsáveis pelo trabalho com o que era conhecido como protocolos de nível de hospedeiro da ARPANET — os procedimentos e formatos que permitiriam a interação dos muitos tipos diferentes de computadores na rede. Era uma exploração fascinante de um novo mundo (para mim) de computação distribuída e comunicação.

Quando começou a projetar o IP, você imaginava que esse protocolo tornar-se-ia tão predominante quanto é hoje?

Quando Bob Kahn e eu começamos a trabalhar nisso, em 1973, acho que estávamos muito mais preocupados com a questão central: como fazer com que redes de pacotes heterogêneas interagissem umas com as outras, admitindo que não poderíamos modificá-las. Esperávamos descobrir um modo que permitisse que um conjunto arbitrário de redes de comutação de pacotes fosse interligado de maneira transparente, de modo que os computadores

componentes das redes pudessem se comunicar fim-a-fim sem precisar de nenhuma tradução entre eles. Acho que sabíamos que estávamos lidando com uma tecnologia poderosa e expansível, mas duvido que tivéssemos uma idéia muito clara do que seria o mundo com centenas de milhões de computadores todos interligados com a Internet.

Em sua opinião, qual é o futuro das redes e da Internet? Quais são os grandes obstáculos/desafios que estão no caminho do seu desenvolvimento?

Acredito que a Internet, em particular, e as redes, em geral, continuarão a proliferar. Hoje já existem evidências convincentes de que haverá bilhões de dispositivos habilitados para a Internet, entre eles equipamentos como telefones celulares, refrigeradores, PDAs, servidores residenciais, televisões, bem como a costumeira coleção de laptops, servidores e assim por diante. Entre os grandes desafios estão o suporte para a mobilidade, a duração das baterias, a capacidade dos enlaces de acesso à rede e a escalabilidade ilimitada do núcleo ótico da rede. A tarefa com a qual estou profundamente envolvido no Jet Propulsion Laboratory é o projeto de uma extensão interplanetária da Internet. Precisaremos descobrir um atalho para passar do IPv4 [endereços de 32 bits] para o IPv6 [128 bits]. A lista é comprida!

Quais pessoas o inspiraram profissionalmente?

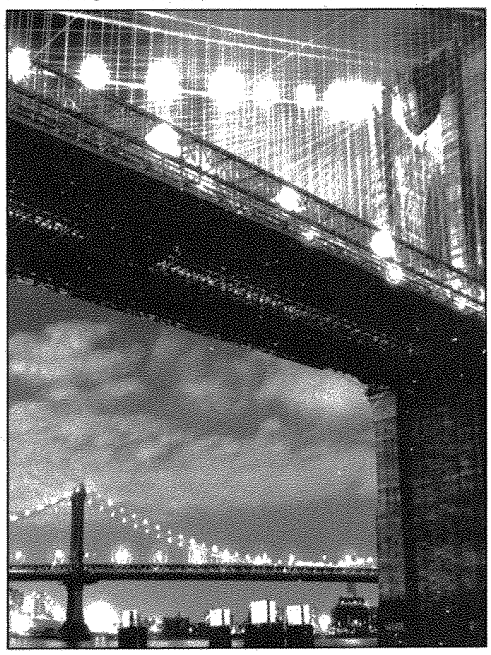
Meu colega Bob Kahn; o orientador de minha tese, Gerald Estrin; meu melhor amigo, Steve Crocker (nós nos conhecemos na escola secundária e ele me apresentou aos computadores em 1960!); e os milhares de engenheiros que continuam a desenvolver a Internet.

Você pode dar algum conselho aos estudantes que estão ingressando no campo das redes/Internet?

Não limitem seu pensamento aos sistemas existentes — imaginem o que poderia ser possível e então comecem a trabalhar para descobrir um meio de sair do estado atual das coisas e chegar lá. Ousem sonhar; eu e alguns colegas do Jet Propulsion Laboratory estamos trabalhando no projeto de uma extensão interplanetária da Internet terrestre. A implementação dessa rede pode levar décadas, uma missão por vez, mas, usando uma paráfrase: "O homem deve tentar alcançar o que está fora do seu alcance; senão, para que existiriam os céus?"

Capítulo 5

A Camada de Enlace e Redes Locais



No capítulo anterior, aprendemos que a camada de rede fornece um serviço de comunicação entre dois hospedeiros. Como mostra a Figura 5.1, esse caminho de comunicação consiste em uma série de enlaces de comunicação que tem início no hospedeiro de origem, passa por uma série de roteadores e termina no hospedeiro de destino. À medida que continuamos a descer a pilha de protocolos, da camada de rede até a camada de enlace, é natural que imaginemos como pacotes são enviados pelos enlaces individuais dentro do caminho de comunicação fim-a-fim. Como os datagramas de camada de rede são encapsulados nos quadros de camada de enlace para transmissão por um único enlace? Protocolos de camada de enlace podem prover transferência confiável de dados de roteador a roteador? Protocolos de camada de enlace diferentes podem ser usados nos diferentes enlaces ao longo do caminho de comunicação? Neste capítulo responderemos a essas e a outras perguntas.

Ao discutir a camada de enlace, descobriremos que há dois tipos de canais de camada de enlace completamente diferentes. O primeiro tipo são os canais de *broadcast*, que são comuns em redes locais (LANs), LANs sem fio, redes por satélite e redes de acesso híbridas de cabo coaxial e de fibra (HFC). No caso do canal de *broadcast*, muitos hospedeiros estão conectados ao mesmo canal de comunicação e é preciso um protocolo de acesso ao meio para coordenar transmissões e evitar colisões. O segundo tipo de canal de camada de enlace é o enlace de comunicação ponto-a-ponto, tal como o existente entre dois roteadores ou entre um modem discado residencial e um roteador ISP. Coordenar o acesso a um enlace ponto-a-ponto é trivial, mas há ainda questões importantes referentes a enquadramento, transferência confiável de dados, detecção de erros e controle de fluxo.

Neste capítulo estudaremos diversas tecnologias importantes de camada de enlace. Examinaremos minuciosamente a Ethernet, de longe a tecnologia predominante para LANs com fio. Estudaremos também o protocolo ponto-a-ponto (*point-to-point protocol* — PPP), o preferido para modens discados em hospedeiros residenciais.

Embora certamente sejam tópicos de camada de enlace, as redes Wi-Fi e, mais geralmente, as LANs sem fio não são estudadas neste capítulo. E não porque a Wi-Fi não seja um tópico importante — muito pelo contrário, pois a revolução das Wi-Fi está mudando drasticamente o modo como as pessoas acessam e utilizam a Internet. Esse tipo de rede é estudado detalhadamente no Capítulo 6, dedicado às redes de computadores sem fio e à mobilidade.

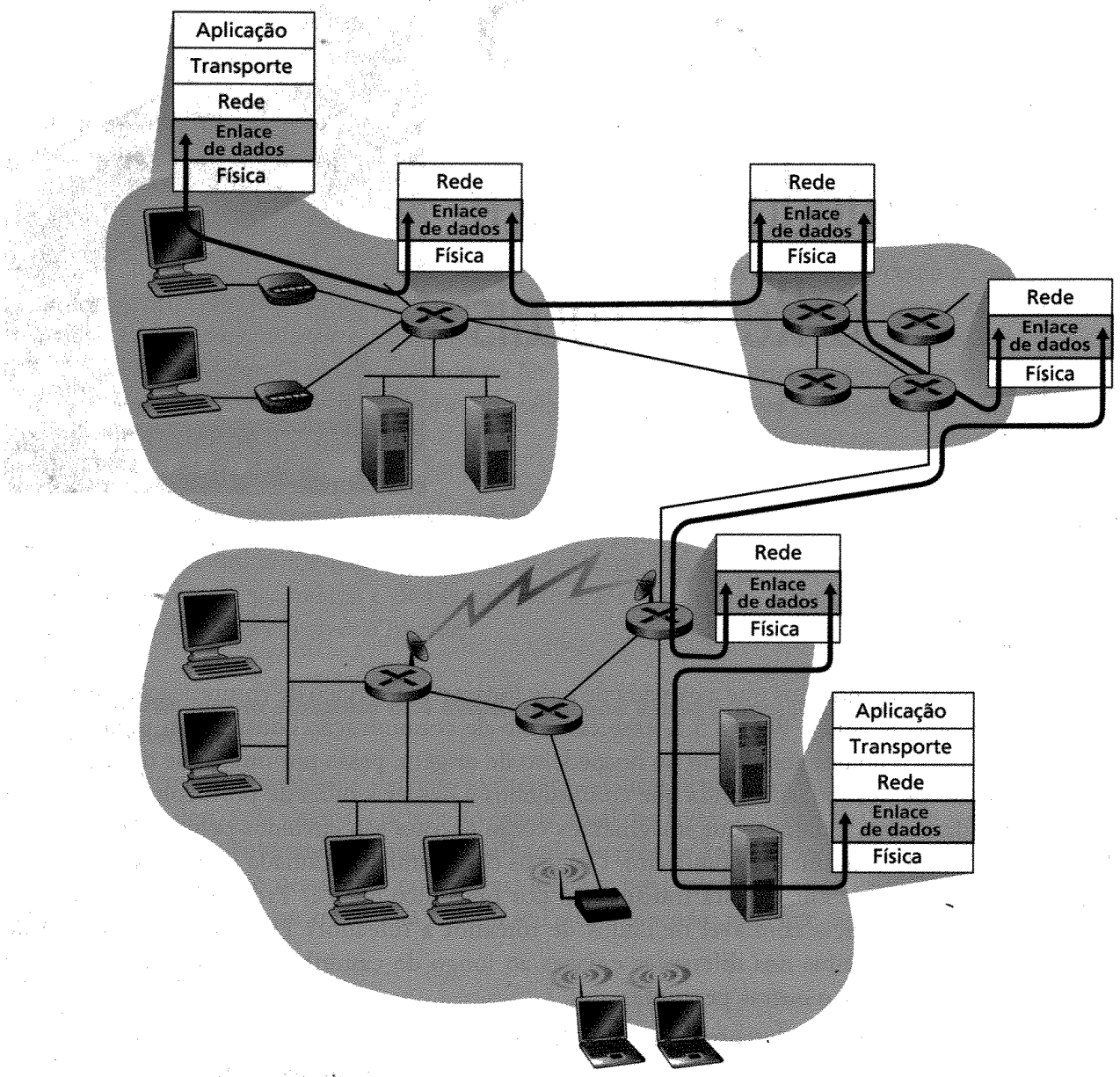


Figura 5.1 A camada de enlace

5.1 Camada de enlace: introdução e serviços

Vamos começar com um pouco de terminologia útil. Achamos que, neste capítulo, é conveniente que nos refiramos aos hospedeiros e roteadores apenas como **nós**, já que, como veremos em breve, não estaremos particularmente preocupados em saber se um nó é um roteador ou um hospedeiro. Também nos referiremos aos canais de comunicação que conectam nós adjacentes ao longo dos caminhos de comunicação como **enlaces**. Para levar um datagrama de um hospedeiro de origem até um hospedeiro de destino, o datagrama tem de ser transportado sobre cada um dos enlaces individuais existentes no caminho fim-a-fim. Considerando um dado enlace, um nó transmissor encapsula o datagrama em um quadro de camada de enlace e transmite o quadro para dentro do enlace, e um nó receptor recebe o quadro e extrai o datagrama.

5.1.1 Os serviços fornecidos pela camada de enlace

Um protocolo de camada de enlace é usado para transportar um datagrama por um enlace individual. O **protocolo de camada de enlace** define o formato dos pacotes trocados entre os nós nas extremidades do

enlace, bem como as ações realizadas por esses nós ao enviar e receber os pacotes. Lembre-se de que no Capítulo 1 dissemos que as unidades de dados trocadas pelo protocolo de camada de enlace são denominadas **quadros** e que cada quadro de camada de enlace caracteristicamente encapsula um datagrama de camada de rede. Como veremos adiante, entre as ações realizadas por um protocolo de camada de enlace ao enviar e receber quadros, estão detecção de erros, retransmissão, controle de fluxo e acesso aleatório. Como exemplos de protocolos de camada de enlace temos os protocolos Ethernet, 802.11 para LANs sem fio (também conhecidas como Wi-Fi), token ring e PPP. Em muitos contextos, o ATM também pode ser considerado um protocolo de camada de enlace. Analisaremos muitos desses protocolos detalhadamente na segunda metade deste capítulo.

Enquanto a camada de rede tem como tarefa movimentar segmentos da camada de transporte fim-a-fim, desde o hospedeiro de origem até o hospedeiro de destino, um protocolo de camada de enlace é encarregado de movimentar datagramas de camada de rede nó a nó por um *único enlace* no caminho. Uma característica importante da camada de enlace é que um datagrama pode ser manipulado por diferentes protocolos de enlace nos diferentes enlaces no caminho. Por exemplo, um datagrama pode ser manipulado pelo protocolo Ethernet no primeiro enlace, pelo PPP no último enlace e por um protocolo WAN de camada de enlace nos enlaces intermediários. É importante notar que os serviços fornecidos pelos protocolos de camada de enlace podem ser diferentes. Por exemplo, um protocolo de camada de enlace pode prover ou não entrega confiável. Assim, a camada de rede deve ser capaz de realizar sua tarefa fim-a-fim em face de um conjunto heterogêneo de serviços individuais de camada de enlace.

Para uma boa compreensão da camada de enlace e de como ela se relaciona com a camada de rede, vamos considerar uma analogia com um sistema de transporte. Considere um agente de viagens que está fazendo o planejamento de uma viagem para um turista desde Princeton, em Nova Jersey, até Lausanne, na Suíça. Suponha que o agente de viagens decida que é mais conveniente para o turista pegar uma limusine de Princeton até o Aeroporto JFK, em seguida um avião do Aeroporto JFK até o Aeroporto de Genebra e, finalmente, um trem do Aeroporto de Genebra até a estação ferroviária de Lausanne. Assim que o agente fizer as três reservas, é responsabilidade da empresa de limusines levar o turista de Princeton ao Aeroporto JFK, é responsabilidade da companhia aérea levar o turista do Aeroporto JFK a Genebra, e é responsabilidade do trem suíço levar o turista de Genebra a Lausanne. Cada um dos três segmentos da viagem é 'direto' entre duas localidades 'adjacentes'. Note que os três segmentos de transporte são administrados por empresas diferentes e usam meios de transporte completamente diferentes (limusine, avião e trem). Embora os meios de transporte sejam diferentes, cada um deles fornece o serviço básico de levar passageiros de uma localidade a outra localidade adjacente. Nessa comparação com o transporte, o turista é análogo a um datagrama, cada segmento de transporte é análogo a um enlace de comunicação, o meio de transporte é análogo a um protocolo de camada de enlace e o agente de viagens é análogo a um protocolo de roteamento.

Embora o serviço básico de qualquer camada de enlace seja mover um datagrama de um nó até um nó adjacente por um único enlace de comunicação, os detalhes do serviço podem variar de um protocolo de camada de enlace para outro. Entre os possíveis serviços que podem ser oferecidos por um protocolo de camada de enlace, estão:

- *Enquadramento de dados.* Quase todos os protocolos de camada de enlace encapsulam cada datagrama de camada de rede dentro de um quadro de camada de enlace antes de transmiti-lo pelo enlace. Um quadro consiste em um campo de dados no qual o datagrama da camada de rede é inserido e em uma série de campos de cabeçalho. (Um quadro pode incluir também campos de trailer [campos no final do quadro], contudo, vamos nos referir também aos campos de trailer como campos de cabeçalho.) A estrutura do quadro é especificada pelo protocolo de camada de enlace. Veremos diversos formatos de quadros diferentes quando examinarmos os protocolos de camada de enlace específicos na segunda metade deste capítulo.
- *Acesso ao enlace.* Um protocolo de controle de acesso ao meio (*medium access control protocol* — MAC) especifica as regras segundo as quais um quadro é transmitido pelo enlace. Para enlaces

ponto-a-ponto que têm um único remetente em uma extremidade do enlace e um único receptor na outra extremidade, o protocolo MAC é simples (ou inexistente) — o remetente pode enviar um quadro sempre que o enlace estiver ocioso. O caso mais interessante é quando vários nós compartilham um único enlace de *broadcast* — o denominado problema de acesso múltiplo. Aqui, o protocolo MAC serve para coordenar as transmissões de quadros dos muitos nós; estudaremos protocolos MAC detalhadamente na Seção 5.3.

- *Entrega confiável.* Quando um protocolo de camada de enlace fornece serviço confiável de entrega, ele garante que vai transportar cada datagrama da camada de rede pelo enlace sem erro. Lembre-se de que certos protocolos de camada de transporte (como o TCP) também fornecem um serviço confiável de entrega. Semelhante ao que acontece com um serviço confiável de entrega de camada de transporte, consegue-se um serviço confiável de entrega de camada de enlace com reconhecimentos e retransmissões (veja a Seção 3.4). Um serviço confiável de entrega de camada de enlace é muito usado por enlaces que costumam ter altas taxas de erros, como é o caso de um enlace sem fio, com a finalidade de corrigir um erro localmente, no enlace no qual o erro ocorre, em vez de forçar uma retransmissão fim-a-fim dos dados por um protocolo de camada de transporte ou de aplicação. Contudo, a entrega confiável de camada de enlace pode ser considerada uma sobrecarga desnecessária para enlaces de baixa taxa de erros, incluindo enlaces de fibra, enlaces coaxiais e muitos enlaces de pares de fios trançados de cobre. Por essa razão, muitos protocolos de camada de enlace com fio não fornecem um serviço de entrega confiável.
- *Controle de fluxo.* Os nós de cada lado de um enlace têm uma capacidade limitada de armazenar quadros. Este é um problema potencial, pois um nó receptor pode receber quadros a uma velocidade maior do que sua capacidade de processá-los. Sem controle de fluxo, o buffer do receptor pode transbordar e quadros podem ser perdidos. Semelhante à camada de transporte, um protocolo de camada de enlace pode fornecer controle de fluxo para evitar que o nó remetente de um lado de um enlace congestione o nó receptor do outro lado do enlace.
- *Detecção de erros.* Um nó receptor pode decidir incorretamente que um bit de um quadro é 0 quando foi transmitido como 1 e vice-versa. Esses erros de bits são introduzidos por atenuação de sinal e ruído eletromagnético. Como não há necessidade de repassar um datagrama que tem um erro, muitos protocolos de camada de enlace oferecem um mecanismo para detectar a presença de um ou mais erros. Isso é feito obrigando o nó transmissor a enviar bits de detecção de erros no quadro e obrigando o nó receptor a realizar uma verificação de erros. A detecção de erros é um serviço muito comum entre protocolos de camada de enlace. Lembre-se de que dissemos nos capítulos 3 e 4 que as camadas de transporte e de rede da Internet também fornecem um serviço limitado de detecção de erros. A detecção de erros na camada de enlace geralmente é mais sofisticada e é implementada em hardware.
- *Correção de erros.* A correção de erros é semelhante à detecção de erros, exceto que um receptor não somente detecta se foram introduzidos erros no quadro, mas também determina exatamente em que lugar do quadro os erros ocorreram (e, então, os corrige). Alguns protocolos (como o ATM) fornecem correção de erros na camada de enlace apenas para o cabeçalho do pacote, e não para o pacote inteiro. Examinaremos a detecção e a correção de erros na Seção 5.2.
- *Half-duplex e full-duplex.* Com transmissão *full-duplex*, os nós em ambas as extremidades de um enlace podem transmitir pacotes ao mesmo tempo. Com transmissão *half-duplex* um nó não pode transmitir e receber pacotes ao mesmo tempo.

Como dissemos antes, muitos serviços fornecidos pela camada de enlace apresentam grandes paralelos com os serviços fornecidos na camada de transporte. Por exemplo, tanto a camada de enlace como a de transporte podem fornecer entrega confiável. Embora os mecanismos usados para prover entrega confiável nas duas camadas sejam semelhantes (veja a Seção 3.4), os dois serviços de entrega confiável não são iguais. Um protocolo de transporte fornece entrega confiável entre dois processos que operam fim-a-fim na rede;

um protocolo de enlace confiável fornece serviço de entrega confiável entre dois nós ligados por um único enlace. De modo semelhante, tanto os protocolos de camada de enlace como os de camada de transporte podem fornecer controle de fluxo e detecção de erros; novamente, o controle de fluxo em um protocolo de camada de transporte é fornecido fim-a-fim, enquanto em um protocolo de camada de enlace é fornecido entre nós adjacentes.

5.1.2 Comunicação por adaptadores

Para um dado enlace de comunicação, o protocolo de camada de enlace é, na maioria das vezes, implementado em um **adaptador**. Um adaptador é uma placa (ou um cartão PCMCIA) que contém RAM, chips DSP, uma interface de barramento de hospedeiro e uma interface de enlace. Adaptadores são também comumente conhecidos como **cartões (placas) de interface de rede** ou **NIC**s (*network interface cards*). Como mostra a Figura 5.2, a camada de rede do nó transmissor (isto é, um hospedeiro ou um roteador) passa um datagrama da camada de rede ao adaptador que processa o lado remetente do enlace de comunicação. O adaptador encapsula o datagrama em um quadro e, então, transmite o quadro para dentro do enlace de comunicação. Do outro lado, o adaptador receptor recebe o quadro inteiro, extrai o datagrama de camada de rede e o passa para a camada de rede. Se o protocolo de camada de enlace fornecer detecção de erros, então é o adaptador remetente que configurará os bits de detecção de erros e é o adaptador receptor que realizará a detecção de erros. Se o protocolo de camada de enlace fornecer entrega confiável, então os mecanismos para entrega confiável (por exemplo, números de seqüência, temporizadores e reconhecimentos) serão inteiramente implementados nos adaptadores. Se o protocolo de camada de enlace utilizar acesso aleatório (veja a Seção 5.3), então o protocolo de acesso aleatório será inteiramente implementado nos adaptadores.

Um adaptador é uma unidade semi-autônoma. Por exemplo, um adaptador pode receber um quadro, determinar se ele está errado e descartá-lo sem notificar outros componentes (por exemplo, uma unidade central de processamento) no nó em que está abrigado (o qual denominaremos nó *pai* do adaptador). Ao receber um quadro, um adaptador interromperá seu nó pai somente se quiser passar um datagrama de camada de rede para cima na pilha de protocolos. De modo semelhante, quando um nó passa um datagrama a um adaptador para baixo na pilha de protocolos, o nó está delegando totalmente ao adaptador a tarefa de transmitir o datagrama por aquele enlace. Um adaptador não é uma unidade completamente autônoma, é semi-autônoma. Embora tenhamos mostrado o adaptador como uma caixa separada na Figura 5.3, ele é normalmente abrigado na mesma caixa física do resto do nó, compartilha energia elétrica e barramentos com o resto do nó e está, afinal, sob o controle do nó.

Como mostra a Figura 5.3, os componentes principais de um adaptador são a interface de barramento e a interface de enlace. A interface de barramento é responsável pela comunicação com o nó pai do adaptador. Ela transfere dados e informações de controle entre o adaptador e o nó pai. A interface de enlace é responsável pela implementação do protocolo de camada de enlace. Além da montagem e desmontagem de datagramas, ela pode prover detecção de erros, acesso aleatório e outras funções de camada de enlace. Ela inclui também o conjunto de circuitos de transmissão e de recepção. Nas tecnologias populares de camada

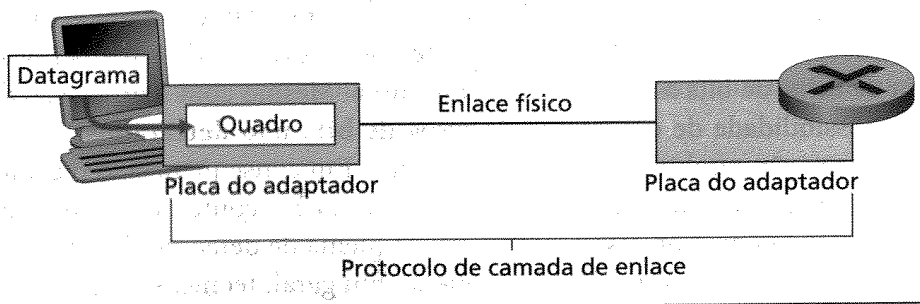


Figura 5.2 O protocolo de camada de enlace para um enlace de comunicação é implementado nos adaptadores nas duas extremidades do enlace

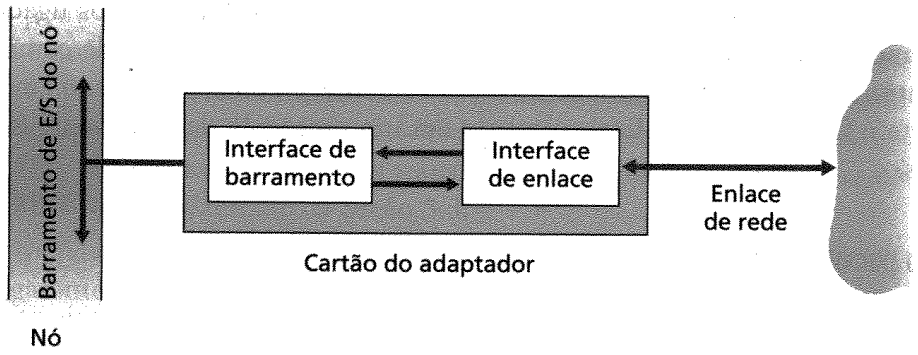


Figura 5.3 O adaptador é uma unidade semi-autônoma

de enlace como Ethernet e Wi-Fi, a interface de enlace é implementada por conjuntos de chips que podem ser comprados no mercado de componentes. Por essa razão, os adaptadores Ethernet e Wi-Fi são baratos. Se quiser saber mais sobre arquitetura de adaptadores Ethernet de 10, 100 e 1.000 Mbps e de adaptadores ATM de 155 Mbps, visite a página da 3Com [3Com, 2004].

5.2 Técnicas de detecção e correção de erros

Na seção anterior, observamos que **detecção e correção de erros no nível de bits** — detecção e correção da corrupção de bits em um quadro de camada de enlace enviado de um nó para outro nó vizinho fisicamente ligado a ele — são dois serviços freqüentemente fornecidos pela camada de enlace. Vimos no Capítulo 3 que serviços de detecção e correção de erros também são freqüentemente oferecidos na camada de transporte. Nesta seção, examinaremos algumas das técnicas mais simples que podem ser usadas para detectar e, em alguns casos, corrigir esses erros de bits. Como esse assunto — em especial, a teoria e a implementação dessas técnicas — é tratado detalhadamente por muitos autores (como [Schwartz, 1980] ou [Bertsekas, 1991]), nossa abordagem será necessariamente breve. Nossa meta é desenvolver uma visão intuitiva das capacidades que as técnicas de detecção e correção de erros fornecem e ver como algumas técnicas simples funcionam e são usadas na prática na camada de enlace.

A Figura 5.4 ilustra o cenário de nosso estudo. No nó remetente, para que os dados, *D*, fiquem protegidos contra erros de bits, eles são aumentados com bits de detecção e de correção (*error detection-and-correction bits* — EDC). Em geral, os dados que devem ser protegidos incluem não somente o datagrama passado para baixo a partir da camada de rede para transmissão pelo enlace, mas também informações de endereçamento de camada de enlace, números de seqüência e outros campos do cabeçalho do quadro de enlace. Tanto *D* como *EDC* são enviados ao nó receptor em um quadro de enlace. No nó receptor, são recebidas seqüências *D'* e *EDC'*. Note que *D'* e *EDC'* podem ser diferentes dos *D* e *EDC* originais, como resultado de alterações nos bits em trânsito.

O desafio do receptor é determinar se *D'* é ou não igual ao *D* original, uma vez que recebeu apenas *D'* e *EDC'*. A exata sintaxe da decisão do receptor na Figura 5.4 (perguntamos se um erro foi detectado, e não se um erro foi cometido!) é importante. Técnicas de detecção e correção de erros permitem que o receptor detecte a ocorrência de erros de bits às vezes, mas não sempre. Mesmo com a utilização de bits de detecção de erros, ainda há a possibilidade de ocorrência de **erros de bits não detectados**, isto é, o receptor pode não perceber que a informação recebida contém erros de bits. Em conseqüência, o receptor poderá entregar um datagrama corrompido à camada de rede ou não perceber que o conteúdo de um campo no cabeçalho do quadro foi corrompido. Assim, é preciso escolher um esquema de detecção de erros para o qual a probabilidade de que essas ocorrências aconteçam seja pequena. Em geral, técnicas mais sofisticadas de detecção e correção de erros (isto é, as que têm uma probabilidade menor de permitir erros de bits não detectados) ficam sujeitas a uma sobrecarga maior — é preciso mais processamento para computar e transmitir um número maior de bits de detecção e correção de erros.

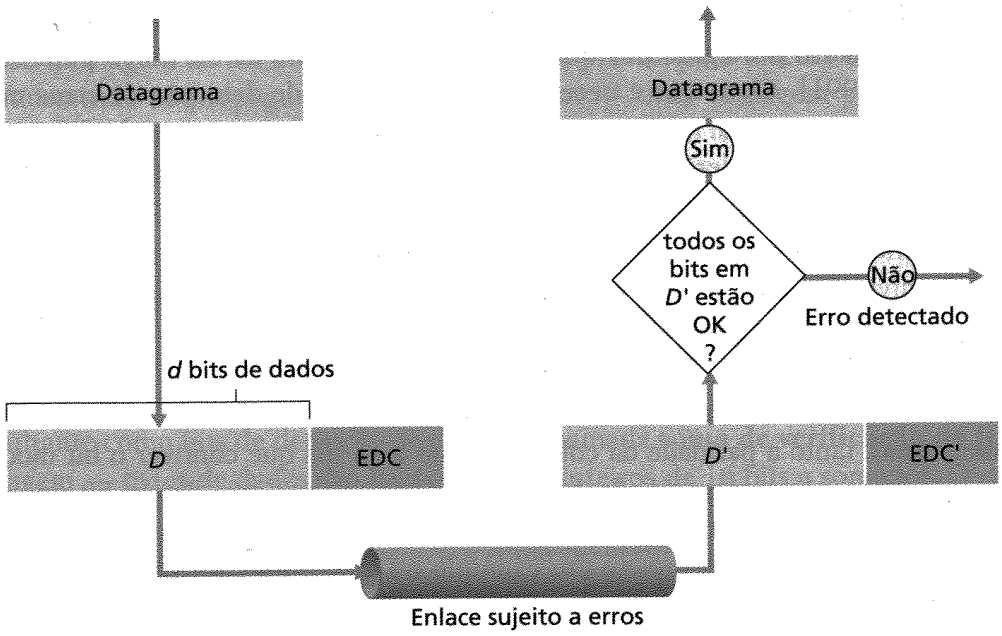


Figura 5.4 Cenário de detecção e correção de erros

Vamos examinar agora três técnicas de detecção de erros nos dados transmitidos — verificações de paridade (para ilustrar as idéias básicas da detecção e correção de erros), métodos de soma de verificação (que são mais empregados na camada de transporte) e verificações de redundância cíclica (CRCs) (que são normalmente empregadas na camada de enlace, nos adaptadores).

5.2.1 Verificações de paridade

Talvez a maneira mais simples de detectar erros seja utilizar um único **bit de paridade**. Suponha que a informação a ser enviada, *D* na Figura 5.4, tenha *d* bits. Em um esquema de paridade par, o remetente simplesmente inclui um bit adicional e escolhe o valor desse bit de modo que o número total de '1' nos bits *d* + 1 (a informação original mais um bit de paridade) seja par. Em esquemas de paridade ímpar, o valor do bit de paridade é escolhido de modo que haja um número ímpar de '1'. A Figura 5.5 ilustra um esquema de paridade par com o bit de paridade armazenado em um campo separado.

A operação do receptor também é simples com um único bit de paridade. O receptor precisa apenas contar quantos '1' há nos *d* + 1 bits recebidos. Se, utilizando um esquema de paridade par, for encontrado um número ímpar de bits de valor 1, o receptor saberá que ocorreu pelo menos um erro de bit. Mais precisamente, ele saberá que ocorreu algum número *ímpar* de erros de bit.

Mas o que acontecerá se ocorrer um número par de erros de bit? É bom que você se convença de que isso resultaria em um erro não detectado. Se a probabilidade de erro de bits for pequena e se for razoável admitir que os erros ocorrem independentemente entre um bit e o bit seguinte, a probabilidade de haver vários erros de bits em um pacote seria bastante pequena. Nesse caso, um único bit de paridade poderia ser suficiente. Contudo, medições demonstraram que, em vez de acontecerem independentemente, os erros fre-

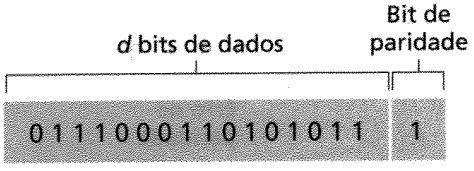


Figura 5.5 Paridade par com um bit de paridade

qüentemente se aglomeram em 'rajadas'. Sob a condição de rajada de erros, a probabilidade de haver erros não detectados em um quadro protegido por um esquema de paridade de bit único pode se aproximar de 50 por cento [Spragins, 1991]. Claro que é necessário um esquema de detecção de erros mais robusto (e, felizmente, é o que se usa na prática!). Mas, antes de examinarmos os esquemas de detecção de erros que são usados na prática, vamos considerar uma generalização simples da paridade de bit único que nos dará uma percepção das técnicas de correção de erros.

A Figura 5.6 mostra uma generalização bidimensional do esquema de paridade de bit único. Nessa figura, os d bits de D são divididos em i filas e j colunas. Um valor de paridade é calculado para cada fila e para cada coluna. Os $i + j + 1$ bits de paridade resultantes compreendem os bits de detecção de erros do quadro da camada de enlace.

Suponha agora que ocorra um erro de bit único nos d bits originais de informação. Com esse esquema de **paridade bidimensional**, tanto a paridade da coluna quanto a da fila que contiver o bit modificado estarão com erro. O receptor então não somente pode detectar que ocorreu um erro de um bit único, mas também usar os índices da fila e da coluna com erros de paridade para realmente identificar o bit que foi corrompido e *corrigir* aquele erro! A Figura 5.6 mostra um exemplo no qual o bit com valor 1 na posição (2,2) está corrompido e mudou para um 0 — um erro que não somente é detectável, como também corrigível no receptor. Embora nossa discussão tenha focalizado os d bits originais de informação, um erro único nos próprios bits de paridade também é detectável e corrigível. A paridade bidimensional também pode detectar (mas não corrigir!) qualquer combinação de dois erros em um pacote. Outras propriedades do esquema de paridade bidimensional são discutidas nos exercícios ao final deste capítulo.

A capacidade do receptor para detectar e corrigir erros é conhecida como **correção de erros de repasse** (*forward error correction* — FEC). Essas técnicas são comumente usadas na armazenagem de áudio e em equipamentos de reprodução, como CDs de áudio. Em um ambiente de rede, as técnicas FEC podem ser usadas isoladamente ou em conjunto com as técnicas ARQ, que examinamos no Capítulo 3. As técnicas FEC são valiosas porque podem reduzir o número requerido de retransmissões do remetente. Talvez o mais

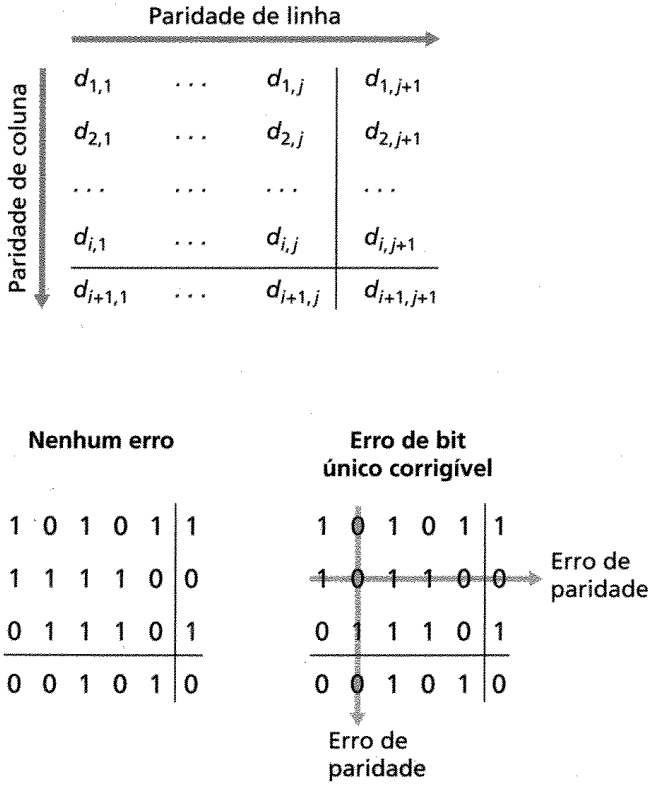


Figura 5.6 Paridade par bidimensional

importante seja que elas permitem imediata correção de erros no receptor. Isso evita ter de esperar pelo atraso de propagação da viagem de ida e volta que o remetente precisa para receber um pacote NAK e para que um pacote retransmitido se propague de volta ao receptor — uma vantagem potencialmente importante para aplicações de rede em tempo real [Rubenstein, 1998]. Entre os trabalhos recentes que examinaram a utilização da FEC em protocolos de controle de erros estão os de [Biersack, 1992; Nonnenmacher, 1998; Byers, 1998; Shacham, 1990].

5.2.2 Métodos de soma de verificação

Em técnicas de soma de verificação, os d bits de dados na Figura 5.4 são tratados como uma seqüência de números inteiros de k bits. Um método simples de soma de verificação é somar esses inteiros de k bits e usar o total resultante como bits de detecção de erros. A chamada **soma de verificação da Internet** é baseada nessa abordagem — bytes de dados são tratados como inteiros de 16 bits e somados. O complemento de 1 dessa soma então forma a soma de verificação da Internet que é carregada no cabeçalho do segmento. Como discutido na Seção 3.3, o receptor verifica a soma de verificação calculando os complementos de 1 da soma dos dados recebidos (inclusive a soma de verificação) e verificando se o resultado contém somente bits 1. Se qualquer um dos bits for 0, isso indicará um erro. O RFC 1071 discute detalhadamente o algoritmo da soma de verificação da Internet e sua implementação. Nos protocolos TCP/IP, a soma de verificação da Internet é calculada em todos os campos (incluindo os campos de cabeçalho e de dados). Em outros protocolos, o XTP, por exemplo [Strayer, 1992], uma soma de verificação é calculada para o cabeçalho e outra soma de verificação é calculada para o pacote inteiro.

Métodos de soma de verificação exigem relativamente pouca sobrecarga de pacote. Por exemplo, as somas de verificação em TCP e UDP utilizam apenas 16 bits. Contudo, oferecem proteção relativamente baixa contra erros em comparação com a verificação de redundância cíclica, discutida mais adiante, que é utilizada com freqüência na camada de enlace. Uma pergunta que surge naturalmente neste ponto é: por que a soma de verificação é utilizada na camada de transporte e a verificação de redundância cíclica é utilizada na camada de enlace? Lembre-se de que a camada de transporte é normalmente implementada em software como parte do sistema operacional de um hospedeiro. Como a detecção de erros na camada de transporte é implementada em software, é importante que o esquema de detecção de erros seja simples e rápido como a soma de verificação. Por outro lado, a detecção de erro na camada de enlace é implementada em hardware dedicado, em adaptadores que podem rodar velozmente as mais complexas operações de CRC.

A principal razão por que a soma de verificação é utilizada na camada de transporte e a CRC, mais forte, é utilizada na camada de enlace é que é fácil implementar a soma de verificação em software.

McAuley [McAuley, 1994] descreve códigos aperfeiçoados de soma de verificação ponderada que são adequados para implementação em software de alta velocidade e Feldmeier [Feldmeier, 1995] apresenta técnicas de implementação rápida em software não somente para códigos de soma de verificação ponderada, mas também para CRC (veja a seguir) e outros códigos.

5.2.3 Verificação de redundância cíclica (CRC)

Uma técnica de detecção de erros usada amplamente nas redes de computadores de hoje é baseada em **códigos de verificação de redundância cíclica** (*cyclic redundancy check* — CRC). Códigos de CRC também são conhecidos como **códigos polinomiais**, já que é possível considerar a cadeia de bits a ser enviada como um polinômio cujos coeficientes são os valores 0 e 1 na cadeia de bits, sendo as operações na cadeia de bits interpretadas como aritmética polinomial.

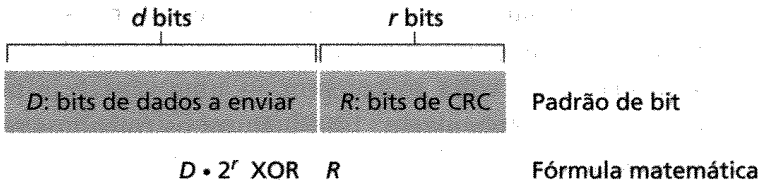


Figura 5.7 Códigos de CRC

Códigos de CRC funcionam como segue. Considere a parcela de d bits de dados, D, que o nó remetente quer enviar para o nó receptor. O remetente e o receptor devem, primeiramente, concordar com um padrão de $r + 1$ bits, conhecido como um **gerador**, que denominaremos G. Vamos exigir que o bit mais significativo (o da extrema esquerda) de G seja um 1. A idéia fundamental por trás dos códigos de CRC é mostrada na Figura 5.7. Para uma dada parcela de dados, D, o remetente escolherá r bits adicionais, R, e os anexará a D de modo que o padrão de $d + r$ bits resultante (interpretado como um número binário) seja divisível exatamente por G, usando aritmética de módulo 2. Assim, o processo de verificação de erros com CRC é simples: o receptor divide os $d + r$ bits recebidos por G. Se o resto for diferente de zero, o receptor saberá que ocorreu um erro; caso contrário, os dados são aceitos como corretos.

Todos os cálculos de CRC são feitos por aritmética de módulo 2 sem 'vai 1' nas adições nem 'empresta 1' nas subtrações. Isso significa que a adição e a subtração são idênticas e ambas são equivalentes à operação *ou exclusivo* (XOR) dos bits dos operandos. Por exemplo,

```
1011 XOR 0101 = 1110
1001 XOR 1101 = 0100
```

Também de modo semelhante temos:

```
1011 - 0101 = 1110
1001 - 1101 = 0100
```

A multiplicação e a divisão são as mesmas da aritmética binária, exceto que, em qualquer adição ou subtração exigida, não se somam nem se emprestam casas. Como na aritmética binária comum, a multiplicação por 2^k desloca um padrão de bits para a esquerda por k casas. Assim, dados D e R, a quantidade $(D \cdot 2^r)$ XOR R dá como resultado o padrão de bit $d + r$ mostrado na Figura 5.7. Usaremos essa representação algébrica do padrão de bit $d + r$ da Figura 5.7 em nossa discussão a seguir.

Vamos agora voltar à questão crucial de como o remetente calcula R. Lembre-se de que queremos descobrir um R tal que exista um n que

$$(D \cdot 2^r) \text{ XOR } R = nG,$$

isto é, devemos escolher um R tal que G divida $(D \cdot 2^r)$ XOR sem resto. Se usarmos a operação *ou exclusivo*, isto é, adicionarmos binariamente, sem vai 1, em R em ambos os lados da equação anterior, teremos:

$$D \cdot 2^r = nG \text{ XOR } R.$$

Essa equação nos diz que, se dividirmos $D \cdot 2^r$ por G, o valor do resto será exatamente R. Em outras palavras, podemos calcular R como

$$R = \text{resto}\ \frac{D \cdot 2^r}{G}$$

A Figura 5.8 ilustra esses cálculos para o caso de $D = 101110$, $d = 6$ e $G = 1001$, $r = 3$. Os 9 bits transmitidos nesse caso são 101110 011. Você deve fazer esses cálculos e também verificar se, na verdade, $D \cdot 2^r = 101011 \cdot G$ XOR R.

Padrões internacionais foram definidos para geradores G de 8, 12, 16 e 32 bits. Uma CRC de 8 bits é usada para proteger o cabeçalho de 5 bytes em células ATM. O padrão CRC 32 de 32 bits, que foi adotado em uma série de protocolos do IEEE da camada de enlace, usa um gerador igual a:

$$G_{\text{CRC-32}} = 10000010011000001000111010110111.$$

Cada um dos padrões de CRC pode detectar erros de rajada de menos do que $r + 1$ bits. (Isso significa que todos os erros de bits consecutivos de r bits ou menos serão detectados.) Além disso, sob hipóteses apropriadas, uma rajada de comprimento maior do que $r + 1$ bits é detectada com probabilidade de $1 - 0{,}5^r$. Cada um dos padrões de CRC também pode detectar qualquer número ímpar de erros de bits. A teoria por trás dos códigos de CRC e de códigos até mais poderosos ultrapassa o escopo deste livro. O livro de [Schwartz, 1980] oferece uma excelente introdução a esse tópico.

5.3 Protocolos de acesso múltiplo

Na introdução deste capítulo, observamos que há dois tipos de enlaces de redes: enlaces ponto-a-ponto e enlaces *broadcast*. Um **enlace ponto-a-ponto** consiste em um único remetente em uma extremidade do enlace e um único receptor na outra extremidade do enlace. Muitos protocolos de camada de enlace foram projetados para enlaces ponto-a-ponto; o PPP (protocolo ponto-a-ponto) é um desses protocolos que examinaremos mais adiante neste capítulo. O segundo tipo de enlace, o **enlace *broadcast***, pode ter vários nós remetentes e receptores, todos conectados ao mesmo canal de transmissão único e compartilhado. O termo *broadcast* é usado aqui porque, quando qualquer um dos nós transmite um quadro, o canal propaga o quadro e cada um dos nós recebe uma cópia. A Ethernet e as LANs sem fio são exemplos de tecnologias de *broadcast* de camada de enlace. Nesta seção, vamos nos afastar um pouco dos protocolos específicos de camada de enlace e examinar, primeiramente, um problema de importância fundamental para a camada de enlace de dados: como coordenar o acesso de vários nós remetentes e receptores a um canal *broadcast* com-

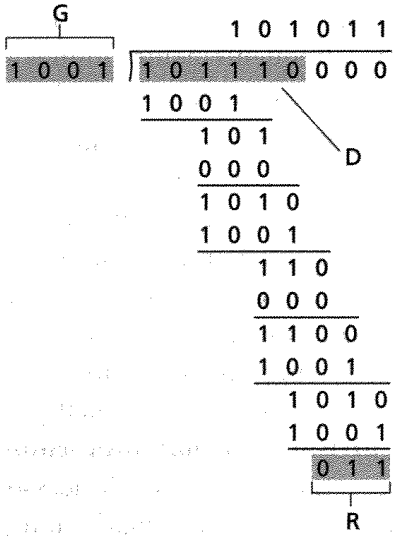


Figura 5.8 Um exemplo de cálculo de CRC

partilhado — o **problema do acesso múltiplo**. Canais *broadcast* são muito usados em LANs, redes que estão geograficamente concentradas em um único edifício (ou empresa, ou campus universitário). Assim, no final da seção, examinaremos também como os canais de acesso múltiplo são usados em LANs.

Todos estamos familiarizados com a noção de *broadcast* — a televisão usa essa tecnologia desde que foi inventada. Mas a televisão tradicional é um *broadcast* unidirecional (isto é, um nó fixo que transmite para muitos nós receptores), ao passo que os nós em um canal *broadcast* de uma rede de computadores tanto podem enviar quanto receber. Talvez uma analogia humana mais apropriada para um canal *broadcast* seja um coquetel, no qual muitas pessoas se encontram em uma sala grande para falar e ouvir (e o ar fornece o meio de transmissão). Uma outra boa analogia é algo com que muitos leitores estão familiarizados — uma sala de aula, onde professor(es) e estudante(s) compartilham, de maneira semelhante, o mesmo e único meio de transmissão *broadcast*. Um problema fundamental em ambos os cenários é determinar quem fala (isto é, quem transmite pelo canal) e quando fala. Como seres humanos, desenvolvemos uma série elaborada de protocolos para compartilhar o canal *broadcast*:

"Dê a todos uma oportunidade de falar."

"Não fale até que alguém fale com você."

"Não monopolize a conversa."

"Levante a mão se tiver uma pergunta a fazer."

"Não interrompa uma pessoa quando ela estiver falando."

"Não durma quando alguém estiver falando."

Redes de computadores têm protocolos semelhantes — denominados **protocolos de acesso múltiplo** —, pelos quais os nós regulam sua transmissão pelos canais *broadcast* compartilhados. Como mostra a Figura 5.9, os protocolos de acesso múltiplo são necessários em uma ampla variedade de cenários de rede, que inclui redes locais com fio e sem fio e redes por satélite. Embora tecnicamente cada nó acesse o canal *broadcast* por meio de seu adaptador, nesta seção vamos nos referir ao nó como o dispositivo de envio e de recepção. Na prática, centenas ou até milhares de nós podem se comunicar diretamente por um canal *broadcast*.

Como todos os nós têm a capacidade de transmitir quadros, mais do que dois nós podem transmitir quadros ao mesmo tempo. Quando isso acontece, todos os nós recebem vários quadros ao mesmo tempo, isto é, os quadros transmitidos **colidem** em todos os receptores. Em geral, quando há uma colisão, nenhum dos nós receptores consegue perceber algum sentido nos quadros que foram transmitidos; de certo modo, os sinais dos quadros que colidem ficam inextricavelmente embaralhados. Assim, todos os quadros envolvidos na colisão são perdidos e o canal *broadcast* é desperdiçado durante o intervalo de colisão. É claro que, se muitos nós querem transmitir quadros freqüentemente, muitas transmissões resultarão em colisões e grande parte da largura de banda do canal *broadcast* será desperdiçada.

Para assegurar que o canal *broadcast* realize trabalho útil quando há vários nós ativos, é preciso coordenar, de algum modo, as transmissões desses nós ativos. Essa tarefa de coordenação é de responsabilidade do protocolo de acesso múltiplo. Durante os últimos 30 anos, milhares de artigos e centenas de teses foram escritos sobre protocolos de acesso múltiplo; um levantamento abrangente desse volume de trabalho pode ser encontrado em [Rom, 1990]. Além disso, a pesquisa ativa sobre protocolos de acesso múltiplo continua devido ao surgimento contínuo de novos tipos de enlaces, particularmente novos enlaces sem fio. Durante anos, dezenas de protocolos de acesso múltiplo foram implementados em uma variedade de tecnologias de camada de enlace. Não obstante, podemos classificar praticamente qualquer protocolo de acesso múltiplo em uma das seguintes categorias: **protocolos de divisão de canal**, **protocolos de acesso aleatório** e **protocolos de revezamento**. Examinaremos essas categorias de protocolos de acesso múltiplo nas três subseções seguintes. Vamos concluir essa visão geral mencionando que, idealmente, um protocolo de acesso múltiplo para um canal *broadcast* com velocidade de R bits por segundo tem as seguintes características desejáveis:

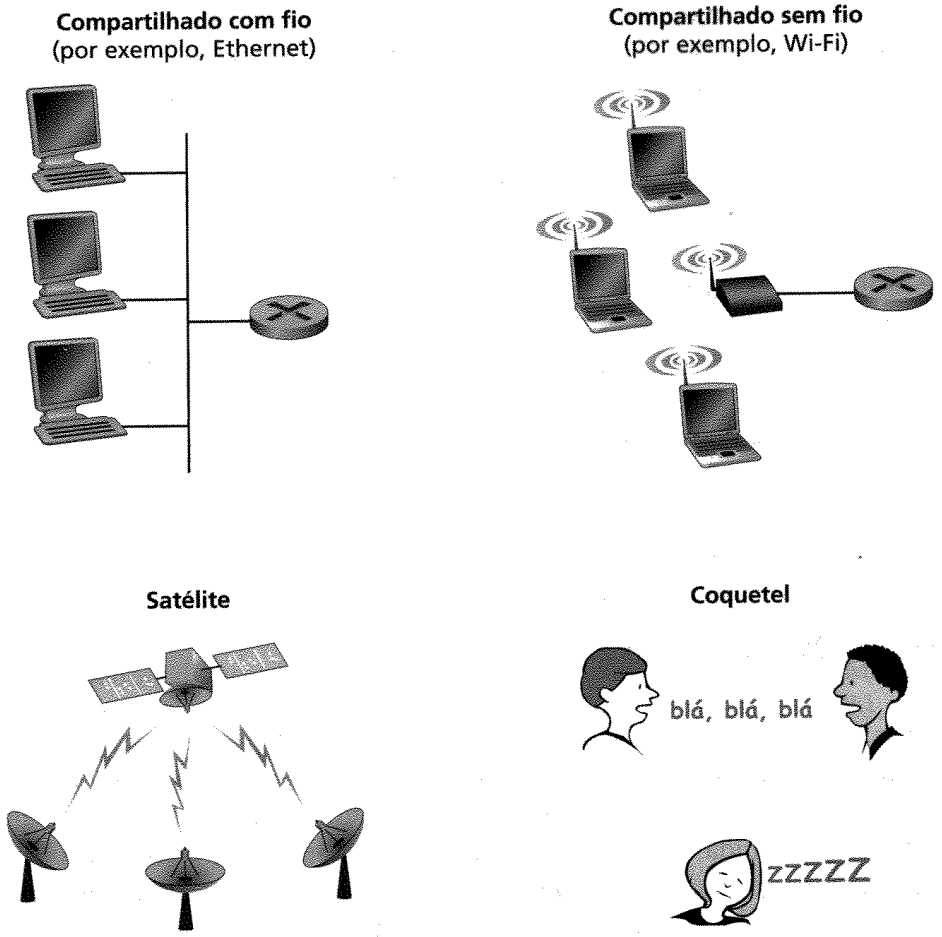


Figura 5.9 Vários canais de acesso múltiplo

1. Quando apenas um nó tem dados para enviar, esse nó tem uma vazão de *R* bps.
2. Quando *M* nós têm dados para enviar, cada um desses nós tem uma vazão de *R/M* bps. Isso não significa necessariamente que cada um dos *M* nós sempre terá uma velocidade instantânea de *R/M*, mas que cada nó deverá ter uma velocidade média de transmissão de *R/M* durante algum intervalo de tempo adequadamente definido.
3. O protocolo é descentralizado, isto é, não há nós mestres que possam falhar e derrubar o sistema inteiro.
4. O protocolo é simples para que sua implementação seja barata.

5.3.1 Protocolos de divisão de canal

Lembre-se de que na Seção 1.3 dissemos que a multiplexação por divisão de tempo (TDM) e a multiplexação por divisão de freqüência (FDM) são duas técnicas que podem ser usadas para dividir a largura de banda de um canal *broadcast* entre todos os nós que compartilham esse canal. Como exemplo, suponha que o canal suporte *N* nós e que a velocidade de transmissão do canal seja *R* bps. O protocolo TDM divide o tempo em **quadros temporais** e depois divide cada quadro temporal em *N* **compartimentos**. (O quadro temporal TDM não deve ser confundido com a unidade de dados de camada de enlace trocada entre adaptadores remetentes e receptores, que também é denominada um quadro. Para reduzir a confusão, nesta seção vamos nos referir à unidade de dados trocada na camada de enlace como um pacote.) Cada compartimento é, então, atribuído a um dos *N* nós. Sempre que um nó tiver um pacote para enviar, ele transmite os bits do pacote durante o compartimento atribuído a ele no quadro temporal rotativo TDM. Normalmente, os

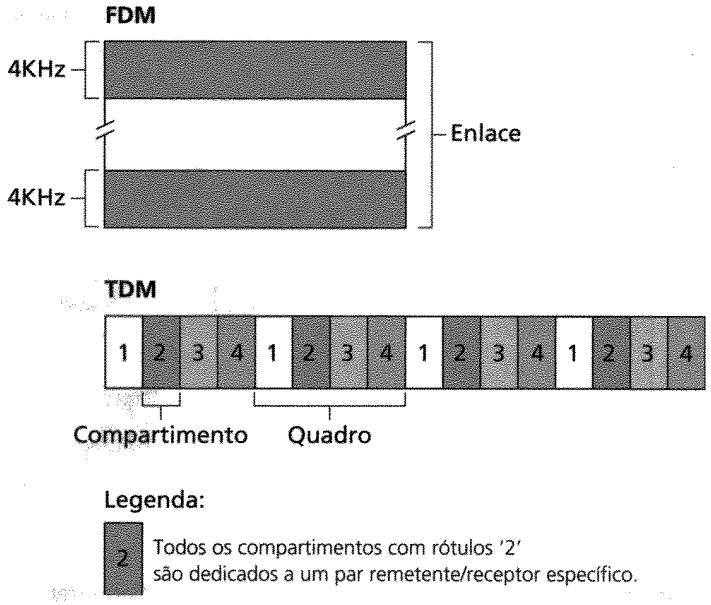


Figura 5.10 Um exemplo de TDM e FDM de quatro nós

tamanhos dos quadros são escolhidos de modo que um único quadro possa ser transmitido durante um compartimento. A Figura 5.10 mostra um exemplo de TDM simples de quatro nós. Voltando à analogia do coquetel, um coquetel regulamentado por TDM permitiria que um dos convidados falasse durante um período de tempo fixo; em seguida, permitiria que um outro convidado falasse pelo mesmo período de tempo e assim por diante. Quando todos tivessem tido sua chance de falar, o modelo seria repetido.

O protocolo TDM é atraente, pois elimina colisões e é perfeitamente justo: cada nó ganha uma velocidade de transmissão dedicada de *R/N* bps durante cada quadro temporal. Contudo, ele tem duas desvantagens importantes. A primeira desvantagem é que um nó fica limitado a uma velocidade média de *R/N* bps, mesmo quando ele é o único nó com pacotes para enviar. A segunda desvantagem é que o nó deve sempre esperar sua vez na seqüência de transmissão — de novo, mesmo quando ele é o único nó com um quadro a enviar. Imagine um convidado que é o único que tem algo a dizer (e imagine uma situação ainda mais rara, em que todos na festa querem ouvir o que aquela pessoa tem a dizer). É óbvio que o protocolo TDM seria uma má escolha para um protocolo de acesso múltiplo para essa festa em particular.

Enquanto o protocolo TDM compartilha o canal *broadcast* no tempo, o protocolo FDM divide o canal de *R* bps em freqüências diferentes (cada uma com uma largura de banda de *R/N*) e reserva cada freqüência a um dos *N* nós, criando, desse modo, *N* canais menores de *R/N* bps a partir de um único canal maior de *R* bps. O protocolo FDM compartilha as vantagens do protocolo TDM — evita colisões e divide a largura de banda com justiça entre os *N* nós. Porém, também compartilha uma desvantagem principal com o protocolo TDM — um nó é limitado a uma largura de banda *R/N*, mesmo quando é o único nó que tem pacotes a enviar.

Um terceiro protocolo de divisão de canal é o protocolo de **acesso múltiplo por divisão de código** (*code division multiple access* — CDMA). Enquanto os protocolos TDM e FDM atribuem aos nós intervalos de tempo e freqüências, respectivamente, o protocolo CDMA atribui um *código* diferente a cada nó. Então, cada nó usa seu código exclusivo para codificar os bits de dados que envia. Se os códigos forem escolhidos cuidadosamente, as redes CDMA terão a maravilhosa propriedade de permitir que nós diferentes transmitam simultaneamente e, ainda assim, consigam que seus receptores respectivos recebam corretamente os bits codificados pelo remetente (admitindo-se que o receptor conheça o código do remetente), a despeito das interferências causadas pelas transmissões dos outros nós. O CDMA vem sendo usado em sistemas militares há algum tempo (devido às suas propriedades antiinterferências) e agora está começando a se difundir para uso civil, em particular para canais de acesso múltiplo sem fio. Como a utilização do CDMA está

muito ligada a canais sem fio, adiaremos a discussão de seus detalhes técnicos para o Capítulo 6. Por enquanto, basta saber que códigos CDMA, assim como compartimentos de tempo em TDM e freqüências em FDM, podem ser alocados a usuários de canais de múltiplo acesso.

5.3.2 Protocolos de acesso aleatório

A segunda classe geral de protocolos de acesso múltiplo são os protocolos de acesso aleatório. Com um protocolo de acesso aleatório, um nó transmissor sempre transmite à taxa total do canal, isto é, R bps. Quando há uma colisão, cada nó envolvido nela retransmite repetidamente seu quadro (isto é, pacote) até que este passe sem colisão. Mas, quando um nó sofre uma colisão, ele nem sempre retransmite o quadro imediatamente. *Em vez disso, ele espera um tempo aleatório antes de retransmitir o quadro.* Cada nó envolvido em uma colisão escolhe atrasos aleatórios independentes. Como após uma colisão os tempos de atraso são escolhidos independentemente, é possível que um dos nós escolha um atraso suficientemente mais curto do que os atrasos dos outros nós em colisão e, portanto, consiga passar seu quadro discretamente para dentro do canal, sem colisão.

Há dezenas, se não centenas, de protocolos de acesso aleatório descritos na literatura [Rom, 1990; Bertsekas, 1991]. Nesta seção, descreveremos alguns dos mais usados — os protocolos ALOHA [Abramson, 1970; 1985] e os protocolos de acesso múltiplo com detecção de portadora (CSMA) [Kleinrock, 1975b]. Mais adiante, na Seção 5.5, examinaremos os detalhes da Ethernet [Metcalfe, 1976], uma variante popular e muito disseminada do protocolo CSMA.

Slotted Aloha

Vamos começar nosso estudo de protocolos de acesso aleatório com um dos mais simples desses protocolos, o slotted ALOHA. Em nossa descrição do slotted ALOHA, admitiremos o seguinte:

- Todos os quadros consistem em exatamente L bits.
- O tempo é dividido em intervalos de tamanho L/R segundos (isto é, um intervalo é igual ao tempo de transmissão de um quadro).
- Os nós começam a transmitir quadros somente no início dos intervalos.
- Os nós são sincronizados de modo que cada nó sabe onde os intervalos começam.
- Se dois ou mais nós colidirem em um intervalo, então todos os nós detectarão o evento de colisão antes do término do intervalo.

Seja p uma probabilidade, isto é, um número entre 0 e 1. O funcionamento do slotted ALOHA em cada nó é simples:

- Quando o nó tem um novo quadro para enviar, espera até o início do próximo intervalo e transmite o quadro inteiro no intervalo.
- Se não houver uma colisão, o nó transmitirá seu quadro com sucesso e, assim, não precisará considerar a retransmissão do quadro. (O nó pode preparar um novo quadro para transmitir, caso tenha algum.)
- Se houver uma colisão, o nó a detectará antes do final do intervalo. Ele retransmitirá seu quadro em cada intervalo subseqüente com probabilidade p até que o quadro seja transmitido sem colisão.

Por retransmissão com probabilidade p, queremos dizer que o nó efetivamente joga uma moeda viciada; coroa corresponde a 'retransmitir', o que ocorre com probabilidade p, enquanto cara corresponde a "pule o intervalo e jogue a moeda novamente no próximo intervalo", o que ocorre com probabilidade $(1 - p)$. Todos os nós envolvidos na colisão jogam suas moedas independentemente.

Aparentemente o slotted ALOHA teria muitas vantagens. Diferentemente da partição de canal, esse protocolo permite que um único nó transmita continuamente à taxa total do canal, R, quando ele for o único nó ativo. (Diz-se que um nó é ativo quanto tem quadros a enviar.) O slotted ALOHA também é altamente descentralizado, porque cada nó detecta colisões e decide independentemente quando retransmitir. (No entanto, requer que os intervalos sejam sincronizados nos nós; em breve discutiremos uma versão sem intervalos do protocolo ALOHA [unslotted ALOHA], bem como protocolos CSMA — nenhum deles requer essa sincronização e, portanto, são totalmente descentralizados.) O slotted ALOHA é também um protocolo extremamente simples.

O slotted ALOHA funciona bem quando há somente um nó ativo, mas qual é sua eficiência quando há vários nós ativos? Nesse caso, há duas preocupações possíveis. A primeira, como mostra a Figura 5.11, é que, quando há vários nós ativos, uma certa fração dos intervalos terá colisões e, portanto, será 'desperdiçada'. A segunda preocupação é que uma outra fração dos intervalos estará *vazia* porque todos os nós ativos evitarão transmitir como resultado da política probabilística de transmissão. Os únicos intervalos 'não desperdiçados' serão aqueles em que exatamente um nó transmite. Um intervalo em que exatamente um nó transmite é denominado um **intervalo bem-sucedido**. A **eficiência** de um protocolo de acesso múltiplo com intervalos é definida como a fração (calculada durante um longo tempo) de intervalos bem-sucedidos no caso em que há um grande número de nós ativos, cada qual tendo sempre um grande número de quadros a enviar. Note que, se não fosse usado nenhum tipo de controle de acesso e cada nó retransmitisse imediatamente após cada colisão, a eficiência seria zero. O slotted ALOHA claramente aumenta a eficiência para além de zero, mas em quanto?

Vamos agora esboçar a derivação da eficiência máxima do slotted ALOHA. Para manter a simplicidade dessa derivação, vamos modificar um pouco o protocolo e admitir que cada nó tenta transmitir um quadro em cada intervalo com probabilidade p. (Isto é, admitimos que cada nó sempre tenha um quadro para enviar e que cada nó transmita com probabilidade p tanto para um quadro novo como para um quadro que já sofreu uma colisão.) Suponha que haja N nós. Então, a probabilidade de que um dado intervalo seja um intervalo bem-sucedido é a probabilidade de que um dos nós transmita e os restantes $N - 1$ nós não transmitam. A probabilidade de que um dado nó transmita é p; a probabilidade de que os nós restantes não transmitam é $(1 - p)^{N-1}$. Por conseguinte, a probabilidade de que um dado nó tenha sucesso é $p(1 - p)^{N-1}$. Como há N nós, a probabilidade de um nó arbitrário ter sucesso é $Np(1 - p)^{N-1}$.

Assim, quando há N nós ativos, a eficiência do slotted ALOHA é $Np(1 - p)^{N-1}$. Para obtermos a eficiência *máxima* para N nós ativos, temos de encontrar um p^* que maximize essa expressão. (Veja os exer-

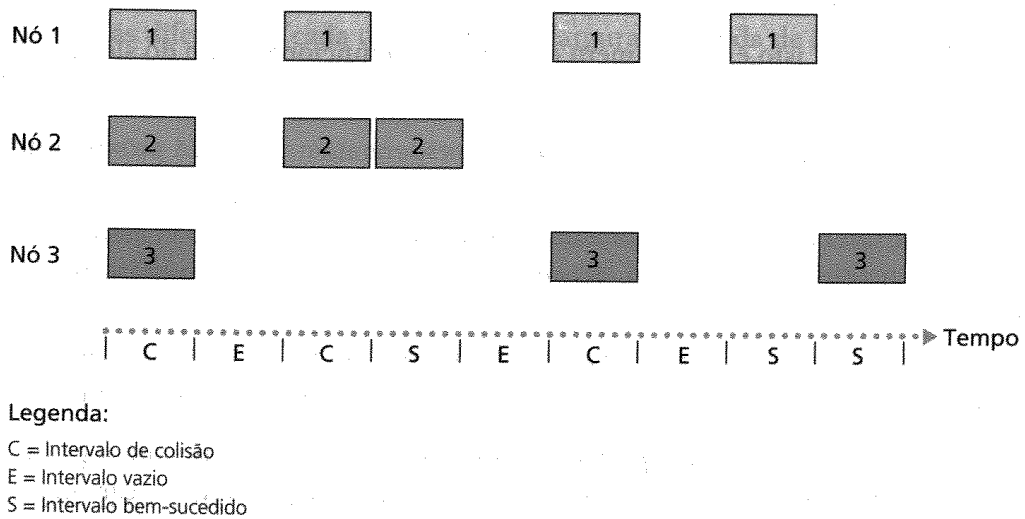


Figura 5.11 Nós 1, 2 e 3 colidem no primeiro intervalo. O nó 2 finalmente é bem-sucedido no quarto intervalo, o nó 1 no oitavo intervalo e o nó 3 no nono intervalo

cícios ao final deste capítulo para um esboço geral dessa derivação.) E, para obtermos a eficiência máxima para um grande número de nós ativos, consideramos o limite de $Np^*(1 - p^*)^{N-1}$ quando N tende ao infinito. (Novamente, veja os exercícios ao final deste capítulo.) Após esses cálculos, descobriremos que a eficiência máxima do protocolo é dada por $1/e = 0{,}37$. Isto é, quando um grande número de nós tem muitos quadros a transmitir, então (na melhor das hipóteses) apenas 37 por cento dos intervalos realiza um trabalho útil. Assim, a taxa efetiva de transmissão do canal não é R bps, mas apenas 0,37 R bps! Uma análise semelhante também demonstra que 37 por cento dos intervalos ficam vazios e 26 por cento apresentam colisões. Imagine o pobre do administrador de rede que comprou um sistema slotted ALOHA de 100 Mbps esperando poder usar a rede para transmitir dados entre um grande número de usuários a uma taxa agregada de, digamos, 80 Mbps! Embora o canal tenha a capacidade de transmitir um dado quadro à taxa máxima do canal de 100 Mbps, no final, a vazão que se consegue com esse canal é de menos de 37 Mbps.

Aloha

O protocolo slotted ALOHA requer que todos os nós sincronizem suas transmissões para que comecem no início de um intervalo. O primeiro protocolo ALOHA [Abramson, 1970] era, na realidade, um protocolo sem intervalos e totalmente descentralizado. No ALOHA puro, quando um quadro chega pela primeira vez (isto é, um datagrama de camada de rede é passado para baixo a partir da camada de rede no nó remetente), o nó imediatamente transmite o quadro inteiro ao canal *broadcast*. Se um quadro transmitido sofrer uma colisão com uma ou mais transmissões, o nó retransmitirá imediatamente (após ter concluído a transmissão total do quadro que sofreu a colisão) o quadro com probabilidade p. Caso contrário, o nó esperará por um tempo de transmissão de quadro. Após essa espera, ele então retransmitirá o quadro com probabilidade p ou espera (permanecendo ocioso) por um outro tempo de quadro com probabilidade $1 - p$.

Para determinar a eficiência máxima do ALOHA puro, vamos focalizar um nó individual. Consideraremos as mesmas premissas que adotamos na análise do slotted ALOHA e tomaremos o tempo de transmissão do quadro como a unidade de tempo. A qualquer dado tempo, a probabilidade de que um nó esteja transmitindo um quadro é p. Suponha que esse quadro comece a transmitir no tempo t_0. Como mostrado na Figura 5.12, para que esse quadro seja transmitido com sucesso, nenhum dos outros nós pode começar sua transmissão no intervalo de tempo $[t_0 - 1, t_0]$. Essa transmissão se sobreporia ao início da transmissão do quadro do nó i. A probabilidade de que todos os outros nós não iniciem uma transmissão nesse intervalo é $(1 - p)^{N-1}$. De maneira semelhante, nenhum outro nó pode iniciar uma transmissão enquanto o nó i estiver transmitindo, pois essa transmissão se sobreporia à parte final da transmissão do nó i. A probabilidade de que todos os outros nós não iniciem uma transmissão nesse intervalo é também $(1 - p)^{N-1}$. Assim, a probabilidade de que um dado nó tenha uma transmissão bem-sucedida é $p(1 - p)^{2(N-1)}$. Levando ao limite, como fizemos no caso do slotted ALOHA, descobrimos que a eficiência máxima do protocolo ALOHA puro é de apenas $1/(2e)$ — exatamente a metade da eficiência do slotted ALOHA. É este o preço que se paga por um protocolo ALOHA totalmente descentralizado.

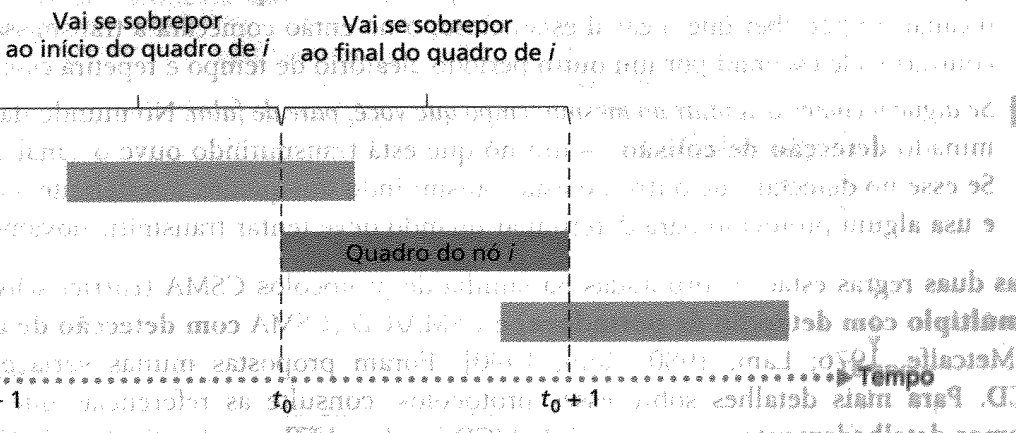
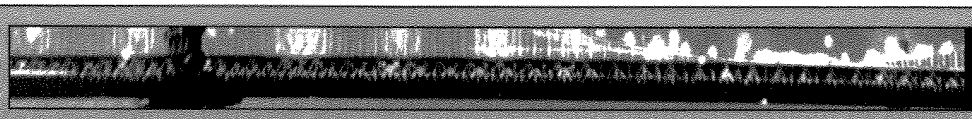


Figura 5.12 Transmissões interferentes no Aloha puro



Norm Abramsom e a ALOHAnet

Norm Abramsom, um doutor em engenharia, era apaixonado por surfe e interessado na comutação de pacotes. Essa combinação de interesses o levou à Universidade do Havaí em 1969. O Havaí é formado por muitas ilhas montanhosas, o que dificulta a instalação e a operação de redes terrestres. Quando não estava surfando, Abramson ficava pensando em como projetar uma rede que fizesse comutação de pacotes por rádio. A rede que ele projetou tinha um hospedeiro central e diversos nós secundários espalhados pelas ilhas havaianas. A rede tinha dois canais, cada um usando uma faixa de freqüência diferente. O canal na direção dos nós secundários fazia *broadcast* de pacotes do hospedeiro central para os hospedeiros secundários; e o canal na direção contrária enviava pacotes dos hospedeiros secundários para o hospedeiro central. Além de enviar pacotes de informação, o hospedeiro central também enviava pelo canal na direção dos nós secundários um reconhecimento para cada pacote recebido com sucesso dos hospedeiros secundários.

Como os hospedeiros secundários transmitiam pacotes de maneira descentralizada, inevitavelmente ocorriam colisões no canal entre eles e o hospedeiro central. Essa observação levou Abramson a inventar, em 1970, o protocolo ALOHA puro, descrito neste capítulo. Em 1970, com o financiamento contínuo da ARPA, Abramson conectou sua ALOHAnet à ARPAnet. O trabalho de Abramson é importante não somente porque foi o primeiro exemplo de uma rede de pacotes por rádio, mas também porque inspirou Bob Metcalfe. Alguns anos após a invenção de Abramson, Metcalfe modificou o protocolo ALOHA e criou o protocolo CSMA/CD e a LAN Ethernet.

CSMA (Acesso múltiplo com detecção de portadora)

Tanto no slotted ALOHA quanto no ALOHA puro, a decisão de transmitir tomada por um nó independe da atividade dos outros nós ligados ao canal *broadcast*. Em particular, um nó não se preocupa se por acaso um outro nó está transmitindo quando ele começa a transmitir nem pára de transmitir se outro nó começar a interferir em sua transmissão. Em nossa analogia do coquetel, os protocolos ALOHA se parecem muito com um convidado mal-educado que continua a tagarelar mesmo quando outras pessoas estão falando. Como seres humanos, temos protocolos que nos levam não somente a nos comportar com mais civilidade, mas também a reduzir o tempo que gastamos 'colidindo' com outros durante a conversação e, conseqüentemente, a aumentar a quantidade de dados que trocamos durante nossas conversas. Especificamente, há duas regras importantes que regem a conversação educada entre seres humanos:

- *Ouça antes de falar.* Se uma pessoa estiver falando, espere até que ela tenha terminado. No mundo das redes, isso é denominado **detecção de portadora** — um nó ouve o canal antes de transmitir. Se um quadro de outro nó estiver correntemente sendo transmitido para dentro do canal, o nó então esperará (se afastará — 'back off') por um período de tempo aleatório e, então, novamente sondará o canal. Se perceber que o canal está ocioso, o nó então começará a transmissão de quadros. Caso contrário, ele esperará por um outro período aleatório de tempo e repetirá esse processo.
- *Se alguém começar a falar ao mesmo tempo que você, pare de falar.* No mundo das redes, isso é denominado **detecção de colisão** — um nó que está transmitindo ouve o canal enquanto transmite. Se esse nó detectar que outro nó está transmitindo um quadro interferente, ele pára de transmitir e usa algum protocolo para determinar quando deve tentar transmitir novamente.

Essas duas regras estão incorporadas na família de protocolos CSMA (*carrier sense multiple access* — **acesso múltiplo com detecção de portadora**) e CSMA/CD (CSMA **com detecção de colisão**) [Kleinrock, 1975b; Metcalfe, 1976; Lam, 1980; Rom, 1990]. Foram propostas muitas variações do CSMA e do CSMA/CD. Para mais detalhes sobre esses protocolos, consulte as referências que acabamos de citar. Estudaremos detalhadamente o esquema CSMA/CD usado na Ethernet na Seção 5.5. Nesta seção, consideraremos algumas das características mais importantes e fundamentais do CSMA e do CSMA/CD.

A primeira pergunta que se poderia fazer sobre o CSMA é a seguinte: se todos os nós realizam detecção de portadora, por que ocorrem colisões? Afinal, um nó vai abster-se de transmitir sempre que perceber que um outro nó está transmitindo. A resposta a essa pergunta pode ser ilustrada utilizando diagramas espaço/tempo [Molle, 1987]. A Figura 5.13 apresenta um diagrama espaço/tempo de quatro nós (A, B, C, D) ligados a um barramento linear de transmissão. O eixo horizontal mostra a posição de cada nó no espaço; o eixo vertical representa o tempo.

No tempo t_0, o nó B percebe que o canal está ocioso, pois nenhum outro nó está transmitindo no momento. Assim, o nó B começa a transmitir e seus bits se propagam em ambas as direções ao longo do meio de transmissão. A propagação para baixo dos bits de B na Figura 5.13 com o aumento do tempo indica que é preciso uma quantidade de tempo de valor diferente de zero para que os bits de B realmente se propaguem (se bem que quase à velocidade da luz) ao longo do meio de transmissão. No tempo t_1 ($t_1 > t_0$), o nó D tem um quadro para enviar. Embora o nó B esteja transmitindo no tempo t_1, os bits que estão sendo transmitidos por B ainda não alcançaram D. Assim, D percebe o canal como ocioso em t_1. De acordo com o protocolo CSMA, D começa então a transmitir seu quadro. Pouco tempo depois, a transmissão de B passa a interferir na transmissão de D em D. Fica evidente na Figura 5.13 que o tempo de **atraso de propagação fim-a-fim de canal** para um canal *broadcast* — o tempo que leva para que um sinal se propague de um dos extremos do canal para outro — desempenhará um papel crucial na determinação de seu desempenho. Quanto mais longo for esse atraso de propagação, maior será a chance de um nó que detecta portadora ainda não poder perceber uma transmissão que já começou em outro nó da rede.

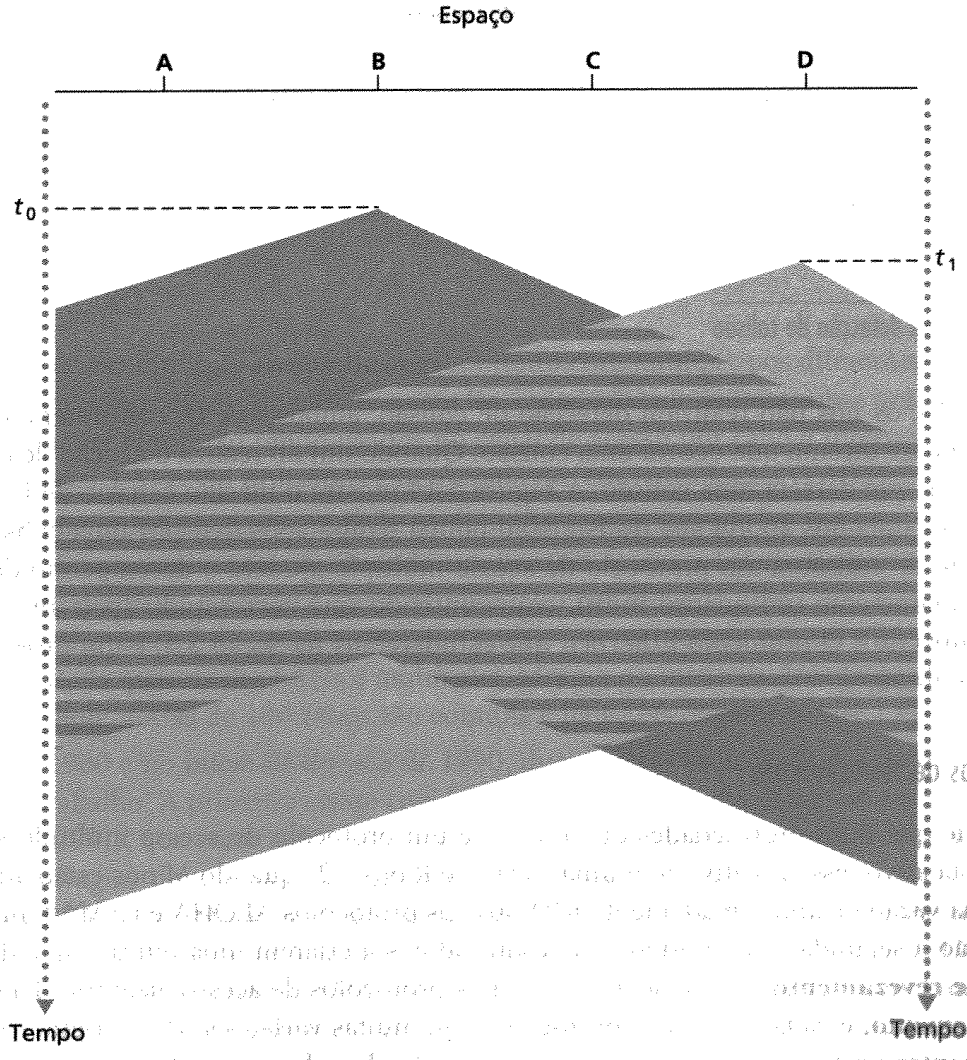


Figura 5.13 Diagrama espaço/tempo de dois nós CSMA com colisão de transmissões

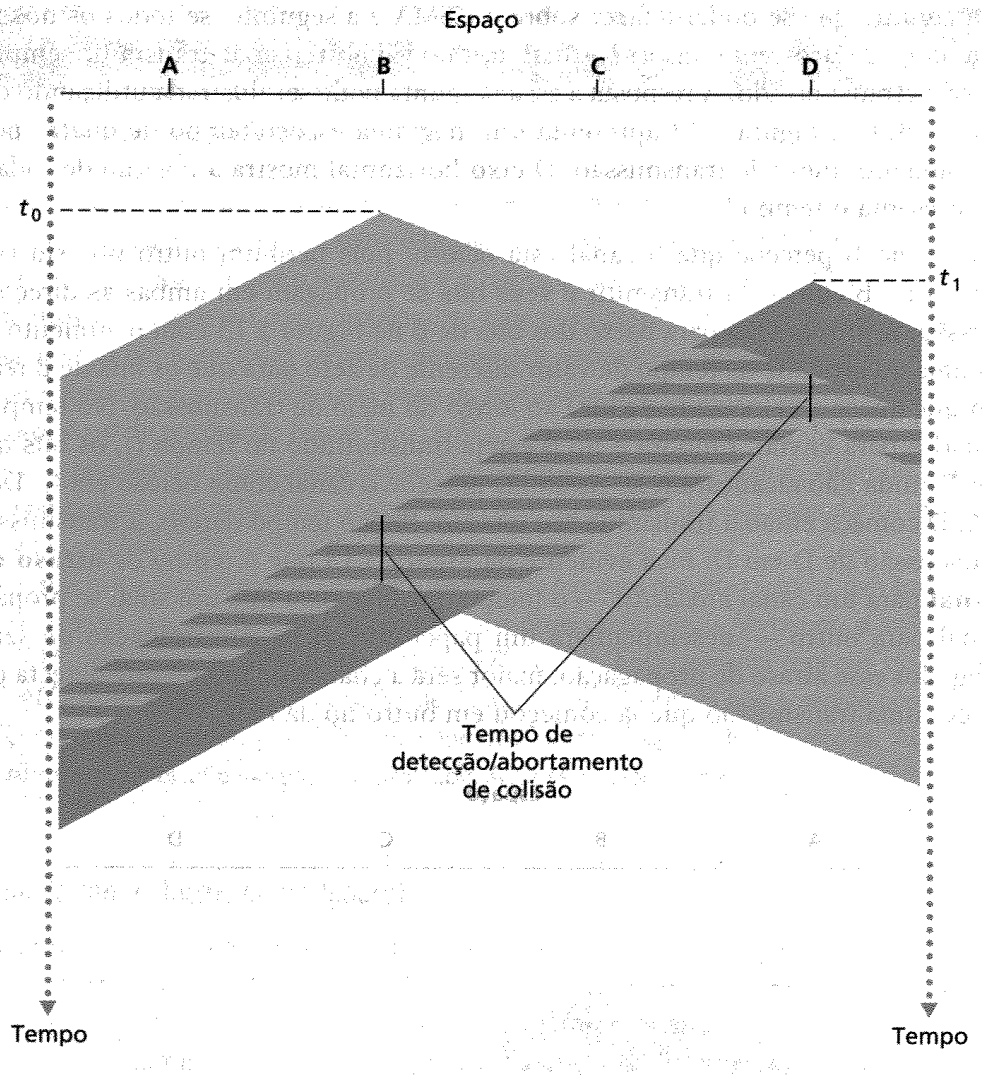


Figura 5.14 CSMA com detecção de colisão

Na Figura 5.13, os nós não realizam detecção de colisão; ambos, B e D, continuam a transmitir seus quadros integralmente mesmo que ocorra uma colisão. Quando um nó realiza detecção de colisão, ele cessa a transmissão assim que detecta uma colisão. A Figura 5.14 mostra o mesmo cenário da Figura 5.13, exceto que cada um dos dois nós aborta sua transmissão pouco tempo após detectar uma colisão. Claramente, adicionar detecção de colisão a um protocolo de acesso múltiplo ajudará o desempenho do protocolo por não transmitir inteiramente um quadro inútil, cujo conteúdo está corrompido (pela interferência de um quadro de um outro nó). O protocolo Ethernet, que estudaremos na Seção 5.5, é um protocolo CSMA que usa detecção de colisão.

5.3.3 Protocolos de revezamento

Lembre-se de que duas propriedades desejáveis de um protocolo de acesso múltiplo são: (1) quando apenas um nó está ativo, esse nó ativo tem uma vazão de R bps; (2) quando M nós estão ativos, então cada nó ativo tem uma vazão de aproximadamente R/M bps. Os protocolos ALOHA e CSMA têm a primeira propriedade, mas não a segunda. Isso motivou os pesquisadores a criarem uma outra classe de protocolos — os **protocolos de revezamento**. Como acontece com os protocolos de acesso aleatório, há dezenas de protocolos de revezamento, e cada um desses protocolos tem muitas variações. Discutiremos nesta seção dois dos mais importantes protocolos. O primeiro é o **protocolo de seleção** (polling). O protocolo de polling requer que um dos nós seja designado como nó mestre. O nó mestre **seleciona** cada um dos nós por alter-

nância circular. Em particular, ele envia primeiramente uma mensagem ao nó 1 dizendo que ele (o nó 1) pode transmitir até um certo número máximo de quadros. Após o nó 1 transmitir alguns quadros, o nó mestre diz ao nó 2 que ele (o nó 2) pode transmitir até um certo número máximo de quadros. (O nó mestre pode determinar quando um nó terminou de enviar seus quadros observando a ausência de um sinal no canal.) O procedimento continua dessa maneira, com o nó mestre escolhendo cada um dos nós de maneira cíclica.

O protocolo de polling elimina as colisões e os intervalos vazios que atormentam os protocolos de acesso aleatório, o que permite que ele tenha uma eficiência muito maior. Mas esse protocolo também tem algumas desvantagens. A primeira desvantagem é que o protocolo introduz um atraso de seleção — o período de tempo requerido para notificar um nó que ele pode transmitir. Se, por exemplo, apenas um nó estiver ativo, então esse nó transmitirá a uma velocidade menor do que R bps, pois o nó mestre tem de escolher cada um dos nós ociosos por vez, cada vez que um nó ativo tiver enviado seu número máximo de quadros. A segunda desvantagem é potencialmente mais séria: se o nó mestre falhar, o canal inteiro ficará inoperante.

O segundo protocolo de revezamento é o **protocolo de passagem de permissão**. Nesse protocolo, não há nó mestre. Um pequeno quadro de finalidade especial conhecido como uma **permissão** (token) é passado entre os nós obedecendo a uma determinada ordem fixa. Por exemplo, o nó 1 poderá sempre enviar a permissão ao nó 2, o nó 2 poderá sempre enviar a permissão ao nó 3, o nó N poderá sempre enviar a permissão ao nó 1. Quando um nó recebe uma permissão, ele a retém somente se tiver alguns quadros para transferir, caso contrário, imediatamente a repassa para o nó seguinte. Se um nó tiver quadros para transmitir quando recebe a permissão, ele enviará um número máximo de quadros e, em seguida, passará a permissão para o nó seguinte. A passagem de permissão é descentralizada e tem uma alta eficiência. Mas também tem seus problemas. Por exemplo, a falha de um nó pode derrubar o canal inteiro. Ou, se um nó acidentalmente se descuida e não libera a permissão, então é preciso chamar algum procedimento de recuperação para colocar a permissão novamente em circulação. Durante muitos anos, foram desenvolvidos muitos protocolos de passagem de permissão, e cada um deles teve de enfrentar esses e outros assuntos delicados; na próxima seção mencionaremos dois desses protocolos, o FDDI e o IEEE 802.5.

5.3.4 Redes locais (LANs)

Protocolos de acesso múltiplo são usados em conjunto com muitos tipos diferentes de canais *broadcast*. Eles têm sido utilizados por canais de satélite e sem fio, cujos nós transmitem sobre um espectro de freqüência comum. Atualmente eles estão sendo usados no canal de acesso por cabo à Internet na direção usuário-provedor (veja a Seção 1.5) e são utilizados extensivamente em redes locais (LANs).

Lembre-se de que uma LAN é uma rede de computadores concentrada em uma área geográfica, tal como um prédio ou um campus universitário. Quando um usuário acessa a Internet a partir de um campus universitário ou de uma empresa, o acesso é quase sempre feito por meio de uma LAN. Especificamente, o acesso se dá do hospedeiro para a LAN, para o roteador, para a Internet, como mostra a Figura 5.15. A velocidade de transmissão, R, da maioria das LANs é muito alta. Mesmo no início da década de 1980 já eram comuns LANs de 10 Mbps; hoje, LANs de 100 Mbps são comuns e há LANs disponíveis de 1 Gbps e 10 Gbps.

Na década de 1980 e no início da década de 1990, duas classes de tecnologias de LAN eram populares nos ambientes de trabalho. A primeira classe consistia nas LANs Ethernet (também conhecidas como LANs 802.3 [Spurgeon, 1999]), que eram redes de acesso aleatório. A segunda classe de tecnologias de LAN compreendiam as tecnologias de passagem de permissão, incluindo a **token ring** (também conhecida como IEEE 802.5) e a **FDDI — interface de dados distribuída de fibra** [Jain, 1994]). Como na Seção 5.5 estudaremos as tecnologias Ethernet com mais detalhes, nesta seção analisaremos apenas as LANs de passagem de permissão. Nossa discussão sobre as tecnologias de passagem de permissão é intencionalmente breve, já que a inexorável concorrência da Ethernet praticamente as extinguiu. Mesmo assim, para dar exemplos de tecnologia de passagem de permissão e apresentar uma pequena perspectiva histórica, será útil falar um pouco sobre anéis de passagem de permissão, ou token rings.

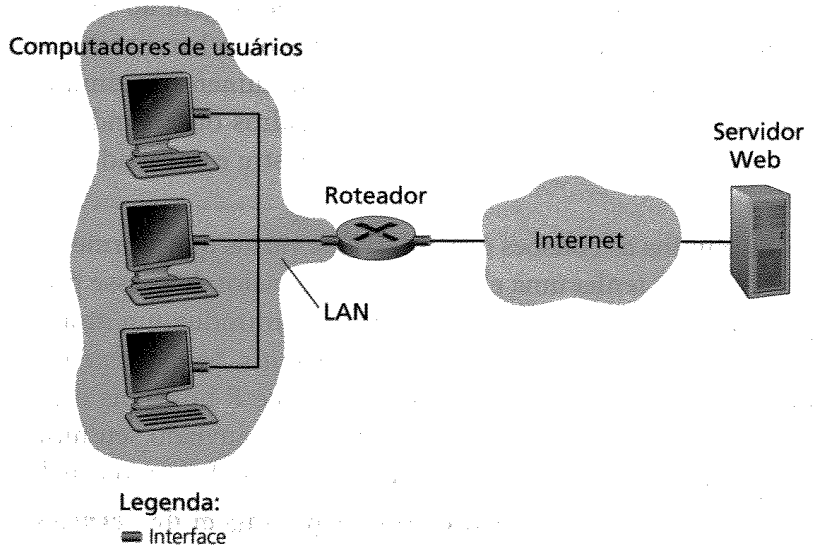


Figura 5.15 Computadores de usuários acessam um servidor Web da Internet por meio de uma LAN. O canal broadcast entre o hospedeiro de um usuário e o roteador consiste em um enlace

Em uma LAN token ring, os *N* nós da LAN (hospedeiros e roteadores) estão conectados em um anel por enlaces diretos. A topologia do anel define a ordem de passagem de permissão. Quando um nó obtém a permissão e envia um quadro, este se propaga ao redor do anel inteiro, criando, dessa maneira, um canal virtual de transmissão *broadcast*. À medida que o quadro se propaga, o nó de destino lê esse quadro no meio de transmissão da camada de enlace. O nó que envia o quadro tem a responsabilidade de remover o quadro do anel. A FDDI foi projetada para LANs de alcance geográfico maior, incluindo as denominadas **redes de área metropolitana** (*metropolitan area networks* — MANs). Para LANs de grande alcance geográfico (que se espalham por muitos quilômetros), é ineficiente permitir que um quadro se propague de volta ao nó remetente tão logo tenha passado do nó de destino. A FDDI faz com que o nó de destino remova o quadro do círculo. (Estritamente falando, a FDDI não é um canal *broadcast* puro, pois nem todos os nós recebem todos os quadros transmitidos.)

5.4 Endereçamento na camada de enlace

Nós — isto é, hospedeiros e roteadores — têm endereços de camada de enlace. Você talvez fique surpreso com isso ao lembrar que dissemos, no Capítulo 4, que nós também têm endereços de camada de rede, e talvez se pergunte por que, afinal de contas, é preciso ter endereços na camada de rede e na camada de enlace. Nesta seção, além de descrevermos a sintaxe e a função dos endereços de camada de enlace, esperamos esclarecer por que as duas camadas de endereços são úteis e, na verdade, indispensáveis.

Além disso, nesta seção estudaremos dois tópicos críticos relacionados a endereços. O primeiro é o Protocolo de Resolução de Endereços (*Address Resolution Protocol* — ARP), que provê um mecanismo que habilita os nós a traduzirem endereços IP para endereços de camada de enlace. O segundo é o Protocolo de Configuração Dinâmica de Hospedeiros (*Dynamic Host Configuration Protocol* — DHCP). Já discutimos o serviço DHCP no Capítulo 4; neste capítulo usaremos o que sabemos sobre endereços de camada de enlace para descrever como o serviço DHCP é implementado.

5.4.1 Endereços MAC

Na verdade, não é o nó (isto é, o hospedeiro ou o roteador) que tem um endereço de camada de enlace, mas o adaptador do nó. Isso é ilustrado na Figura 5.16. Um endereço de camada de enlace é também

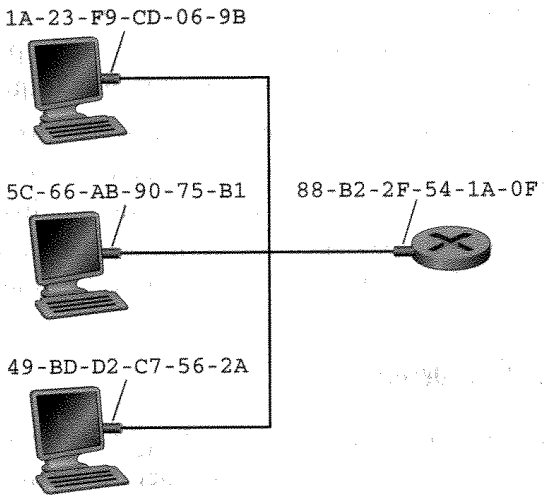


Figura 5.16 Cada adaptador conectado à LAN tem um endereço MAC exclusivo

denominado um **endereço de LAN**, um **endereço físico** ou um **endereço MAC** (*media access control* — controle de acesso ao meio). Como a expressão endereços MAC parece ser o mais popular, daqui em diante nos referiremos a endereços de camada de enlace como endereços MAC. Para a maior parte das LANs (incluindo a Ethernet e as LANs 802.11 sem fio), o endereço MAC tem 6 bytes de comprimento, o que dá 2^{48} possíveis endereços MAC. Como ilustrado na Figura 5.16, esses endereços de 6 bytes são tipicamente expressos em notação hexadecimal, com cada byte do endereço expresso como um par de números hexadecimais. Um fato importante referente aos endereços MAC é que eles são permanentes — quando um adaptador é fabricado, um endereço MAC é gravado na ROM do adaptador.

Uma propriedade interessante dos endereços MAC é que não existem dois adaptadores com o mesmo endereço. Isso pode parecer surpreendente, dado que os adaptadores são fabricados em muitos países por inúmeras empresas diferentes. Como uma empresa fabricante de adaptadores em Taiwan se certifica de que está usando endereços diferentes dos usados por uma empresa fabricante de adaptadores na Bélgica? A resposta é que o IEEE gerencia o espaço físico de endereços MAC. Em particular, quando uma empresa quer fabricar adaptadores, compra, por uma taxa nominal, uma parcela do espaço de endereços que consiste em 2^{24} endereços. O IEEE aloca a parcela de 2^{24} endereços fixando os primeiros 24 bits de um endereço MAC e permitindo que a empresa crie combinações exclusivas com os últimos 24 bits para cada adaptador.

O endereço MAC de um adaptador tem uma estrutura linear (oposta à estrutura hierárquica) e nunca muda, não importando para onde vá o adaptador. Um computador portátil com um cartão Ethernet tem sempre o mesmo endereço MAC, não importando para onde o computador vá. Um PDA com uma interface 802.11 tem sempre o mesmo endereço MAC onde quer que vá. Lembre-se de que, ao contrário, um endereço IP tem uma estrutura hierárquica (isto é, uma parte que é da rede e uma parte que é do hospedeiro) e que o endereço IP de um nó precisa ser trocado quando o hospedeiro muda de lugar. O endereço MAC de um adaptador é análogo ao número do CPF de uma pessoa, que também tem uma estrutura linear e não muda, não importando para onde a pessoa vá. Um endereço IP é análogo ao endereço postal de uma pessoa, que é hierárquico e precisa ser trocado quando a pessoa muda de lugar. Exatamente como uma pessoa pode achar útil ter um endereço postal, bem como um número de CPF, também é útil para um nó ter um endereço de camada de rede, bem como um endereço MAC.

Como descrevemos no início desta seção, quando um adaptador quer enviar um quadro para algum adaptador de destino, o adaptador remetente insere no quadro o endereço MAC do destino e envia o quadro para dentro da LAN. Se a LAN utilizar transmissão *broadcast* (como a LAN 802.11 e muitas LANs Ethernet), o quadro será recebido e processado por todos os outros adaptadores na LAN. Em particular, cada adaptador que recebe o quadro verificará se o endereço MAC de destino que está no quadro combina com seu próprio endereço MAC. Se os endereços combinarem, o adaptador extrairá o datagrama encerrado no

quadro e o passa para cima na pilha de protocolos até seu nó pai. Se os endereços não combinarem, o adaptador descartará o quadro sem passar o datagrama de camada de rede para cima na pilha de protocolos. Assim, somente o adaptador no nó de destino interromperá seu nó pai quando receber um quadro.

No entanto, às vezes um adaptador remetente *quer* que todos os outros adaptadores na LAN recebam e *processem* o quadro que ele está prestes a enviar. Nesse caso, o adaptador remetente insere **um endereço de *broadcast*** MAC especial no campo de endereço do destinatário do quadro. Para LANs que usam endereços de 6 bytes (como a Ethernet e as LANs de passagem de permissão), o endereço de *broadcast* é uma cadeia de 48 bits 1 consecutivos (isto é, FF-FF-FF-FF-FF-FF em notação hexadecimal).

5.4.2 ARP (protocolo de resolução de endereços)

Como existem endereços de camada de rede (por exemplo, endereços IP da Internet) e endereços de camada de enlace (isto é, endereços MAC), é preciso fazer a tradução de um para o outro. Para a Internet, esta é uma tarefa do **protocolo de resolução de endereços** (*address resolution protocol* — ARP) [RFC 826].

Para compreender a necessidade de um protocolo como o ARP, considere a rede mostrada na Figura 5.17. Nesse exemplo simples, cada nó tem um único endereço IP e o adaptador de cada nó tem um único endereço MAC. Como sempre, endereços IP são mostrados em notação decimal com pontos e endereços MAC, em notação hexadecimal. Agora, suponha que o nó com endereço IP 222.222.222.220 queira mandar um datagrama IP para o nó 222.222.222.222. (Por exemplo, o nó de destino 222.222.222.222 pode ser um servidor Web e o nó remetente 222.222.222.220 pode ter determinado o endereço IP do servidor Web por meio do DNS.) Nesse exemplo, os nós de fonte e de destino estão na mesma rede (LAN), no sentido do endereçamento estudado na Seção 4.4.2. Para enviar um datagrama, o nó da fonte deve dar a seu adaptador não somente o datagrama IP, mas também o endereço MAC para o nó de destino 222.222.222.222. Passados o datagrama IP e o endereço MAC, o adaptador do nó remetente montará um quadro de camada de enlace contendo o endereço MAC do nó receptor e enviará o quadro para dentro da LAN.

Princípios na prática

Mantendo a independência das camadas

Há diversas razões por que os nós têm endereços MAC além de endereços de camada de rede. Em primeiro lugar, LANs são projetadas para protocolos de camada de rede arbitrários, e não apenas para IP e para a Internet. Se os adaptadores recebessem endereços IP, e não os 'neutros' endereços MAC, eles não poderiam suportar com facilidade outros protocolos de camada de rede (por exemplo, IPX ou DECNet). Em segundo lugar, se adaptadores usassem endereços de camada de rede, em vez de endereços MAC, o endereço de camada de rede teria de ser armazenado na RAM do adaptador e reconfigurado toda vez que o adaptador mudasse de local (ou fosse ligado). Outra opção é não usar nenhum endereço nos adaptadores e fazer com que cada um deles passe os dados (caracteristicamente, um datagrama IP) de cada quadro que recebe para cima na pilha de protocolos até seu nó pai. O nó pai poderia, então, verificar se o endereço combina com o endereço da camada de rede. Um problema com essa opção é que o nó pai seria interrompido por cada quadro enviado à LAN, bem como por quadros destinados a outros nós na mesma LAN *broadcast*. Em resumo, para que as camadas sejam blocos construtivos em grande parte independentes na arquitetura de uma rede, muitas delas precisam ter seu próprio esquema de endereços. Já vimos até agora três tipos diferentes de endereços: nomes de hospedeiros para a camada de aplicação, endereços IP para a camada de rede e endereços MAC para a camada de enlace.

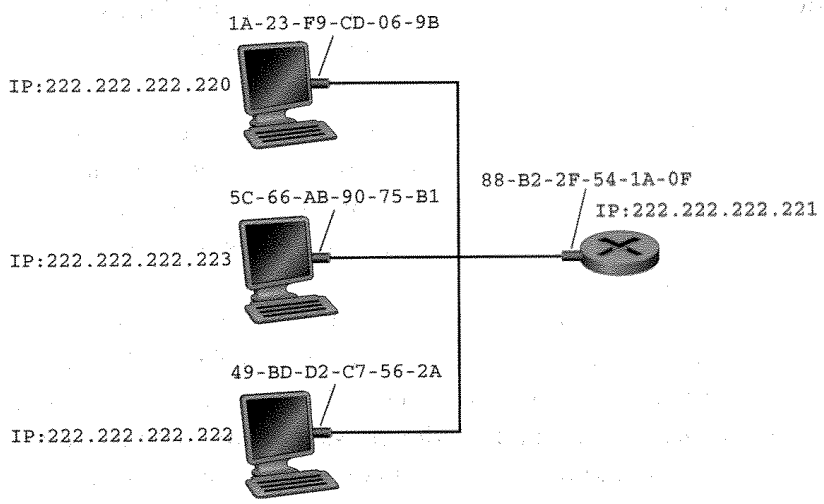


Figura 5.17 Cada nó em uma LAN tem um endereço IP e o adaptador de cada nó tem um endereço MAC

A pergunta importante considerada nesta seção é como o nó remetente determina o endereço MAC para o nó com endereço IP 222.222.222.222? Como você já deve ter adivinhado, ele usa o ARP. Um módulo ARP no nó remetente toma como entrada qualquer endereço IP na mesma LAN e retorna o endereço MAC correspondente. No exemplo em questão, o nó remetente 222.222.222.220 fornece a seu módulo ARP o endereço IP 222.222.222.222 e o módulo ARP retorna endereço MAC correspondente, 49-BD-D2-C7-56-2A.

Assim, vemos que o ARP converte um endereço IP para um endereço MAC. Sob muitos aspectos, o ARP é análogo ao DNS (estudado na Seção 2.5), que converte nomes de hospedeiros para endereços IP. Contudo, uma importante diferença entre os dois conversores é que o DNS converte nomes de hospedeiros para máquinas em qualquer lugar da Internet, ao passo que o ARP converte endereços IP apenas para nós na mesma sub-rede. Se um nó na Califórnia tentasse usar o ARP para converter o endereço IP de um nó no Mississippi, o ARP devolveria um erro.

Agora que já explicamos o que o ARP faz, vamos ver como ele funciona. Cada nó (hospedeiro ou roteador) tem em sua RAM uma **tabela ARP** que contém mapeamentos de endereços IP para endereços MAC. A Figura 5.18 mostra como seria uma tabela ARP no nó 222.222.222.220. A tabela ARP também contém um valor de tempo de vida (TTL) que indica quando cada mapeamento será apagado da tabela. Note que a tabela não contém necessariamente um registro para cada nó da sub-rede; alguns nós podem ter tido registros que já expiraram, ao passo que outros nós podem jamais ter sido registrados na tabela. Um tempo de remoção típico para um registro é de 20 minutos a partir do momento em que foi colocado em uma tabela ARP.

Agora suponha que o nó 222.222.222.220 queira enviar um datagrama que tem endereço IP para outro nó daquela sub-rede. O nó remetente precisa obter o endereço MAC do nó de destino, dado o endereço IP daquele mesmo nó. Essa tarefa será fácil se a tabela ARP do nó remetente tiver um registro para esse nó de destino. Mas, e se, naquele momento, a tabela ARP não tiver um registro para o nó destinatário? Em particular, suponha que o nó 222.222.222.220 queira enviar um datagrama para o nó 222.222.222.222. Nesse caso, o nó remetente usa o protocolo ARP para converter o endereço. Primeiro, o nó remetente monta um pacote especial denominado **ARP Query**. Um pacote ARP tem diversos campos, incluindo os endereços IP e MAC de envio e de recepção.

Endereço IP	Endereço MAC	TTL
222.222.222.221	88-B2-2F-54-1A-0F	13:45:00
222.222.222.223	5C-66-AB-90-75-B1	13:52:00

Figura 5.18 Uma possível tabela ARP no nó 222.222.222.220

e um endereço IP de fonte 'desse hospedeiro', 0.0.0.0. O cliente DHCP passa o datagrama IP para seu adaptador, que encapsula o datagrama em um quadro de camada de enlace. Esse quadro inclui o endereço de *broadcast* MAC (FF-FF-FF-FF-FF-FF) no campo de endereço de destino. O cliente DHCP então envia o quadro de *broadcast* contendo a mensagem de descoberta para dentro da sub-rede. Esse quadro de *broadcast* será recebido por todos os adaptadores na rede. Se um servidor DHCP estiver conectado à mesma sub-rede, ele processará a mensagem de descoberta encapsulada (ver adiante); se um agente de relay DHCP estiver conectado à sub-rede, ele repassará o quadro à rede com um servidor DHCP. (Esse quadro retransmitido terá um endereço de fonte MAC diferente.) A mensagem de descoberta contém um ID de transação que permite que respostas subseqüentes sejam correlacionadas com a requisição de descoberta.

- *Oferta(s) de servidor DHCP.* Um servidor DHCP que recebe uma mensagem de descoberta DHCP responde ao cliente com uma **mensagem de oferta DHCP**. Visto que vários servidores DHCP podem estar presentes na sub-rede, o cliente poderá se encontrar na invejável posição de poder escolher entre diversas ofertas. Cada mensagem de oferta de servidor contém o ID de transação da mensagem de descoberta recebida, o endereço IP proposto para o cliente, a máscara de rede e um **tempo de aluguel de endereço IP** — a quantidade de tempo durante a qual o endereço IP será válido. É comum que o servidor determine um tempo de aluguel de várias horas ou dias [Droms, 1999]. O quadro de camada de enlace que contém o datagrama IP que contém o segmento UDP que contém a mensagem de oferta DHCP é então enviado ao cliente que está chegando. (Se você achar esse encapsulamento aninhado confuso, talvez seja melhor voltar e encontrar seu rumo relendo a Seção 1.7.)
- *Requisição DHCP.* O cliente que acabou de chegar escolherá entre uma ou mais ofertas de servidor e responderá a essa oferta selecionada com uma mensagem de requisição DHCP, ecoando os parâmetros de configuração.
- *DHCP ACK.* O servidor responde à mensagem de requisição DHCP com uma mensagem DHCP ACK, confirmando os parâmetros requisitados.

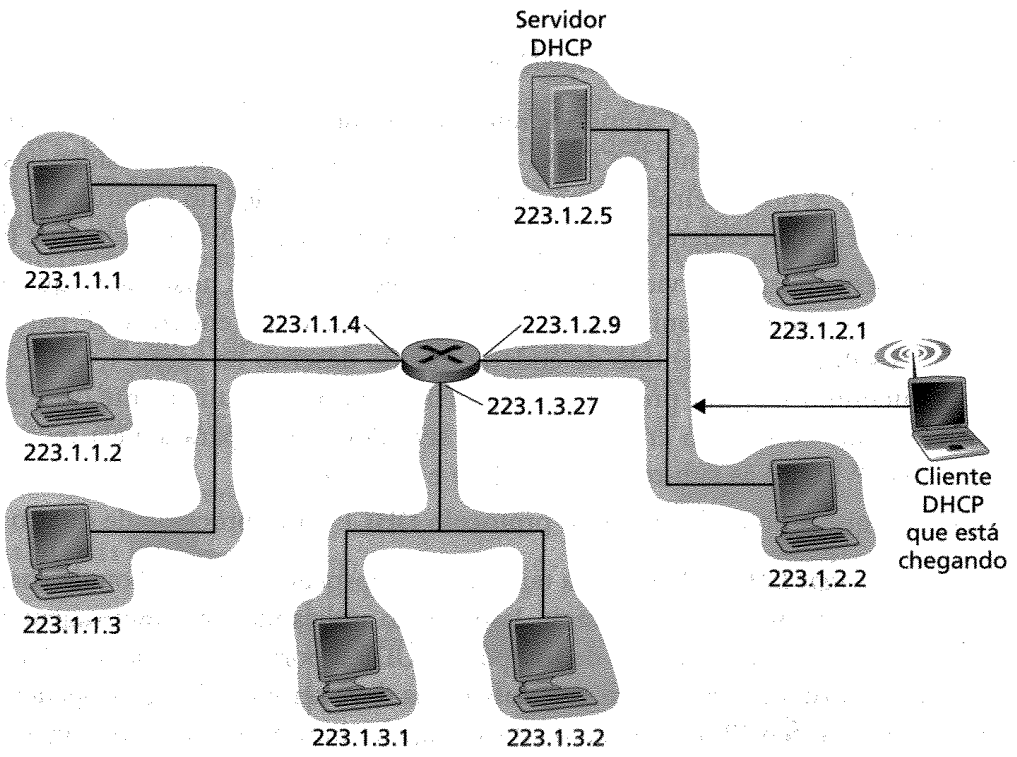


Figura 5.20 Cenário cliente-servidor DHCP

Tão logo o cliente receba a mensagem DHCP ACK, a interação está concluída e o cliente pode usar o endereço IP alocado pelo DHCP durante o período de aluguel. Uma vez que um cliente pode querer usar seu endereço após o término do período de aluguel, o DHCP também provê um mecanismo que permite ao cliente renovar o aluguel de um endereço IP.

Uma interação cliente-servidor DHCP simples é mostrada na Figura 5.21 para o ambiente de rede mostrado na Figura 5.20. Nessa figura, `yiaddr` (que quer dizer 'your Internet address') indica o endereço que está sendo alocado a um cliente que acabou de chegar.

O valor da característica plug-and-play do DHCP é claro. Considere um estudante que leva seu laptop da sala de aula para a biblioteca ou para seu quarto e em cada um desses lugares se conecta a uma nova sub-rede, obtendo, assim, um novo endereço IP. É inadmissível pensar que um administrador de sistema tivesse de reconfigurar laptops em cada localização, e poucos estudantes (exceto os que estão estudando redes de computadores!) teriam a capacidade de configurar seus laptops manualmente. Entretanto, do ponto de vista da mobilidade, o DHCP tem desvantagens. Visto que um novo endereço IP é obtido do DHCP toda vez que um nó se conecta a uma nova sub-rede, não é possível manter uma conexão com uma aplicação remota enquanto um nó móvel passa de uma sub-rede à outra. Examinaremos, no Capítulo 6, o IP móvel — uma extensão recente da infra-estrutura do IP que permite que um nó móvel utilize seu endereço único permanente ao se movimentar entre sub-redes.

Detalhes adicionais sobre DHCP podem ser encontrados em [Droms, 1999] e em [dhc, 2004]. Uma implementação de referência do DHCP com código-fonte aberto pode ser encontrada em Internet Systems Consortium [ISC, 2004].

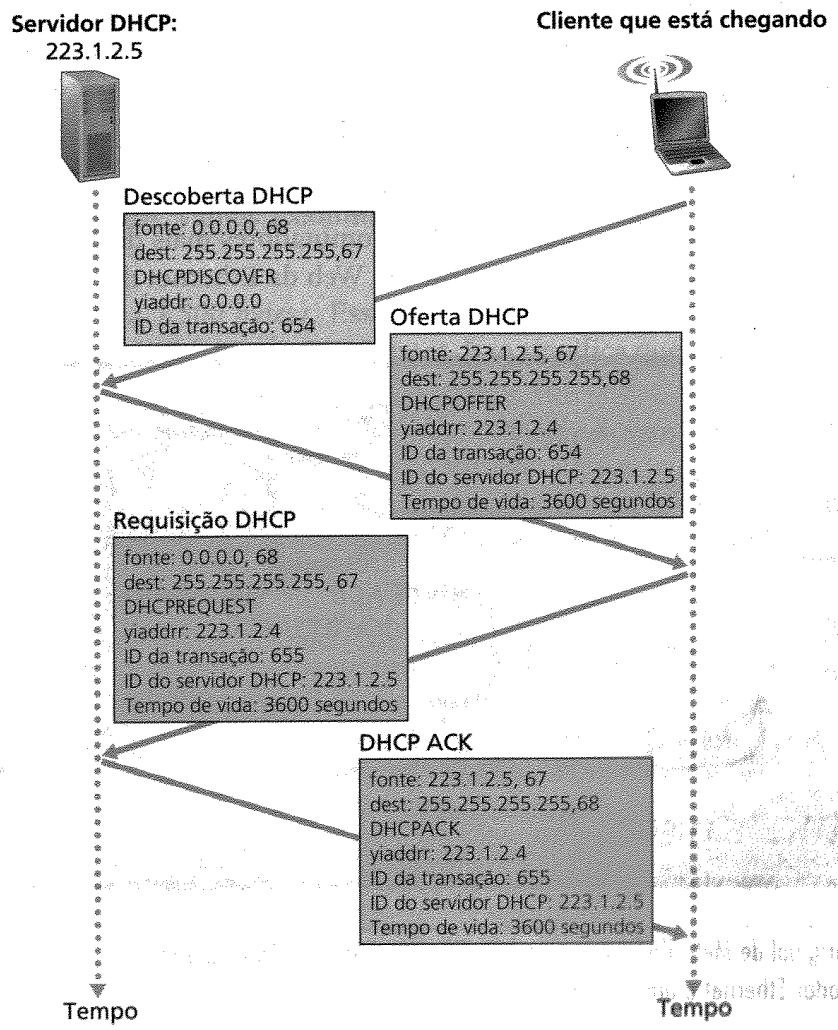


Figura 5.21 Interação cliente-servidor DHCP

5.5 Ethernet

A Ethernet praticamente tomou conta do mercado de LANs com fio. Na década de 1980 e início da década de 1990, ela enfrentou muitos desafios de outras tecnologias LAN, incluindo token ring, FDDI e ATM. Algumas dessas outras tecnologias conseguiram conquistar uma parte do mercado de LANs durante alguns anos. Mas, desde sua invenção, em meados da década de 1970, a Ethernet continuou a se desenvolver e crescer e conservou sua posição dominante no mercado. Hoje, ela é de longe a tecnologia preponderante de LAN com fio e é provável que continue assim no futuro próximo. Podemos dizer que a Ethernet está sendo para a rede local o que a Internet tem sido para a rede global.

Há muitas razões para o sucesso da Ethernet. Em primeiro lugar, ela foi a primeira LAN de alta velocidade amplamente disseminada. Como foi disponibilizada cedo, os administradores de rede ficaram bastante familiarizados com a Ethernet — com suas maravilhas e sutilezas — e relutaram em mudar para outras tecnologias LAN quando estas apareceram em cena. Em segundo lugar, token ring, FDDI e ATM são tecnologias mais complexas e mais caras do que a Ethernet, o que desencorajou ainda mais os administradores na questão da mudança. Em terceiro lugar, a razão mais atraente para mudar para uma outra tecnologia LAN (como FDDI e ATM) era normalmente a velocidade mais alta da nova tecnologia; contudo, a Ethernet sempre se defendeu produzindo versões que funcionavam a velocidades iguais, ou mais altas. E, também, a Ethernet comutada foi introduzida no início da década de 1990, o que aumentou ainda mais sua velocidade efetiva de dados. Finalmente, como a Ethernet se tornou muito popular, o hardware para Ethernet (em particular, adaptadores, hubs e comutadores) se tornou mercadoria comum, de custo muito baixo.

A LAN Ethernet original foi inventada em meados da década de 1970 por Bob Metcalfe e David Boggs. A Figura 5.22 mostra o desenho esquemático de Metcalfe para a Ethernet. Nessa figura você notará que a LAN Ethernet original usava um barramento para interconectar os nós. Na verdade, essa topologia de barramento persistiu durante toda a década de 1980 e por grande parte da década de 1990; em particular, a tecnologia Ethernet 10Base2, que utiliza um cabo coaxial fino para o barramento, era imensamente popular na década de 1990. Contudo, exceto uma ocasional instalação herdada, quase todas as instalações Ethernet de hoje utilizam uma topologia em estrela, como mostra a Figura 5.23. No centro da topologia em estrela está um **hub** ou **comutador**. Em breve discutiremos hubs e comutadores mais detalhadamente. Uma excelente fonte de informação on-line sobre a Ethernet está no site Web da Ethernet de Spurgeon [Spurgeon, 1999].

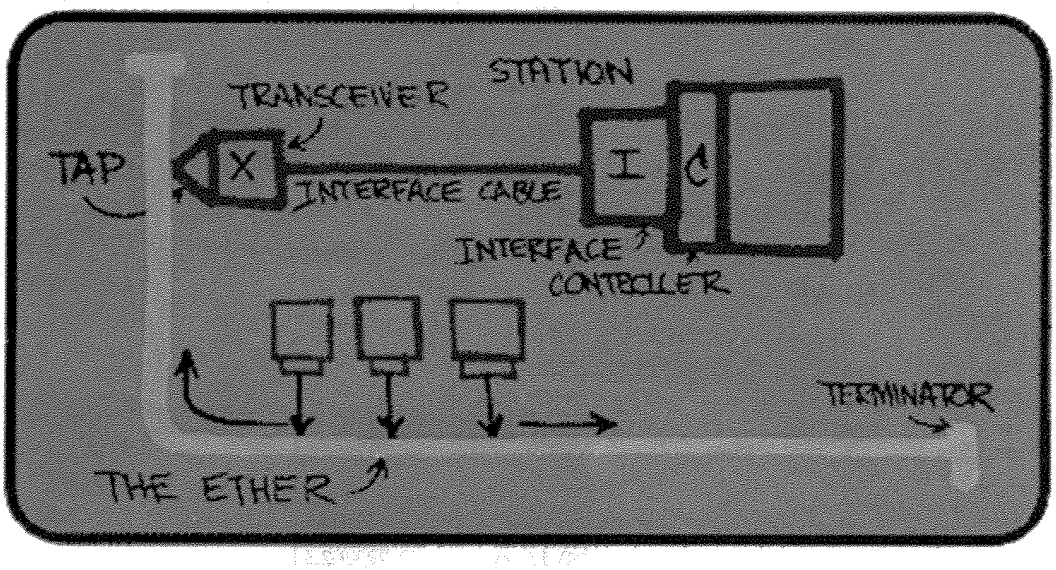


Figura 5.22 O projeto original de Metcalfe levou ao padrão Ethernet 10Base5. Esse padrão incluía um cabo de interface que conectava o adaptador Ethernet a um transceptor externo

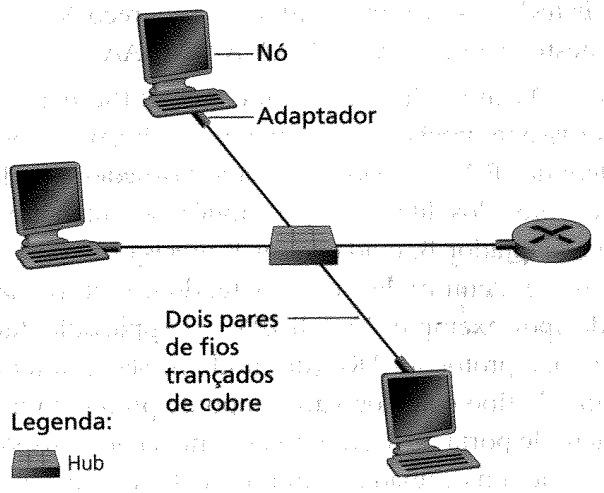


Figura 5.23 Topologia em estrela para Ethernet. Nós são interconectados com um hub

5.5.1 Estrutura do quadro Ethernet

O quadro Ethernet é mostrado na Figura 5.24. Podemos aprender muito sobre a Ethernet examinando o quadro Ethernet. Para colocar nossa discussão de quadros Ethernet em um contexto tangível, vamos considerar o envio de um datagrama IP de um hospedeiro a outro, estando os dois na mesma LAN Ethernet (por exemplo, a LAN Ethernet da Figura 5.23). Embora a carga útil do nosso quadro Ethernet seja um diagrama IP, aproveitamos para comentar que um quadro Ethernet também pode carregar outros pacotes de camada de rede. Seja o adaptador do remetente, adaptador A, com o endereço MAC AA-AA-AA-AA-AA-AA; seja o adaptador receptor, adaptador B, com o endereço MAC BB-BB-BB-BB-BB-BB. O adaptador remetente encapsula o datagrama IP dentro de um quadro Ethernet e passa o quadro à camada física. O adaptador receptor recebe o quadro da camada física, extrai o datagrama IP e o passa para a camada de rede. Nesse contexto, vamos examinar agora os seis primeiros campos do quadro Ethernet, como ilustrados na Figura 5.24.

- *Campo de dados (46 a 1.500 bytes).* Esse campo carrega o datagrama IP. A unidade máxima de transferência (MTU) da Ethernet é 1.500 bytes. Isso significa que, se o datagrama IP exceder 1.500 bytes, o hospedeiro terá de fragmentar o datagrama, como discutimos na Seção 4.4.1. O tamanho mínimo do campo de dados é 46 bytes. Isso significa que, se um datagrama IP tiver menos do que 46 bytes, o campo de dados terá de ser 'recheado' de modo a completar os 46 bytes. Quando se usa recheio (stuffing), os dados passados à camada de rede contêm o recheio, bem como um datagrama IP. A camada de rede usa o campo de comprimento do cabeçalho do datagrama IP para remover o recheio.
- *Endereço de destino (6 bytes).* Esse campo contém os endereços MAC do adaptador de destino, BB-BB-BB-BB-BB-BB. Quando o adaptador B recebe um quadro Ethernet cujo endereço de destino é ou BB-BB-BB-BB-BB-BB, ou o endereço de *broadcast* MAC, ele passa o conteúdo do campo de dados para a camada de rede. Se receber um quadro com qualquer outro endereço MAC, ele o descartará.

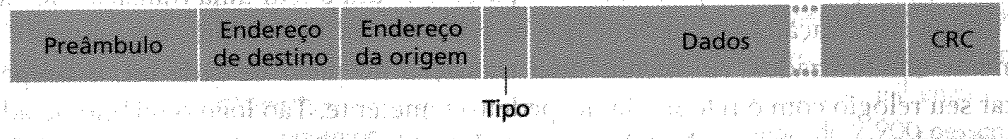


Figura 5.24 Estrutura do quadro Ethernet

- *Endereço da fonte (6 bytes).* Esse campo contém o endereço MAC do adaptador que transmite o quadro para a LAN, neste exemplo, AA-AA-AA-AA-AA-AA.
- *Campo de tipo (2 bytes).* O campo de tipo permite que a Ethernet multiplexe protocolos de camada de rede. Para entender isso, é preciso ter em mente que hospedeiros podem usar outros protocolos de camada de rede além do IP. Na verdade, um dado hospedeiro pode suportar vários protocolos de camada de rede e usar protocolos diferentes para aplicações diferentes. Por essa razão, quando o quadro Ethernet chega ao adaptador B, o adaptador B precisa saber para qual protocolo de camada de rede ele deve passar (isto é, demultiplexar) o conteúdo do campo de dados. O IP e outros protocolos de camada de rede (por exemplo, Novell IPX ou AppleTalk) têm seu próprio número de tipo padronizado. Além disso, o protocolo ARP (discutido na seção anterior) tem seu próprio número de tipo. Note que o campo de tipo é análogo ao campo de protocolo no datagrama de camada de rede e aos campos de número de porta no segmento de camada de transporte; todos esses campos servem para ligar um protocolo de uma camada a um protocolo da camada acima.
- *Verificação de redundância cíclica (CRC) (4 bytes).* Como discutido na Seção 5.2.3, a finalidade do campo de CRC é permitir que o adaptador receptor, o adaptador B, detecte se algum erro foi introduzido no quadro, isto é, se os bits do quadro foram trocados. Entre as causas de erro nos bits estão a atenuação da amplitude do sinal e a energia eletromagnética ambiente que penetra nos cabos e placas de interface Ethernet. A detecção de erros é realizada como segue. Quando o hospedeiro A monta o quadro Ethernet, ele calcula um campo de CRC, que é obtido de um mapeamento dos outros bits no quadro (exceto os bits do preâmbulo). Quando o hospedeiro B recebe o quadro, ele aplica o mesmo mapeamento ao quadro e verifica se o resultado do mapeamento é igual ao que está no campo de CRC. Essa operação é denominada, no hospedeiro receptor, **verificação de CRC**. Se a verificação de CRC falhar (isto é, se o resultado do mapeamento não for igual ao conteúdo do campo de CRC), então o hospedeiro B saberá que há um erro no quadro.
- *Preâmbulo (8 bytes).* O quadro Ethernet começa com um campo de preâmbulo de 8 bytes. Cada um dos primeiros 7 bytes do preâmbulo tem um valor de 10101010; o último byte é 10101011. Os primeiros 7 bytes do preâmbulo servem para 'despertar' os adaptadores receptores e sincronizar seus relógios com o relógio do remetente. Por que os relógios poderiam estar fora de sincronia? Não esqueça que o adaptador A visa transmitir o quadro a 10 Mbps, 100 Mbps ou 1 Gbps, dependendo do tipo de LAN Ethernet. Contudo, como nada é absolutamente perfeito, o adaptador A não transmitirá o quadro exatamente à mesma velocidade-alvo; sempre haverá alguma variação em relação à velocidade-alvo, uma variação que não é conhecida a priori pelos outros adaptadores na LAN. Um adaptador receptor pode sincronizar com o relógio do adaptador A simplesmente sincronizando os bits dos primeiros 7 bytes do preâmbulo. Os dois últimos bits do oitavo byte do preâmbulo (os primeiros dois 1s consecutivos) alertam o adaptador B de que 'algo importante' está chegando. Quando o hospedeiro B percebe os dois 1s consecutivos, sabe que os próximos 6 bytes são o endereço de destino. Um adaptador pode perceber quando um quadro termina simplesmente detectando a ausência de cadeia de bits.

A Ethernet usa transmissão em banda-base, isto é, o adaptador envia um sinal digital diretamente ao canal *broadcast*. A placa de interface não desloca o sinal para outra banda de freqüência, como é feito nos sistemas ADSL e de modem a cabo. Muitas tecnologias Ethernet (por ex., 10BaseT) também usam a codificação Manchester, como mostra a Figura 5.25. Com codificação Manchester cada bit contém uma transição; um 1 tem uma transição de cima para baixo, ao passo que um 0 tem uma transição de baixo para cima. A razão para o uso da codificação Manchester é que os relógios nos adaptadores remetentes e receptores não estão perfeitamente sincronizados. Ao incluir uma transição no meio de cada bit, o hospedeiro receptor pode sincronizar seu relógio com o relógio do hospedeiro remetente. Tão logo o relógio do adaptador receptor esteja sincronizado, o receptor pode delinear cada bit e determinar se é um 1 ou um 0. A codificação Manchester é mais uma operação de camada física do que de camada de enlace; contudo, descrevemos essa operação brevemente aqui porque ela é muito usada na Ethernet.

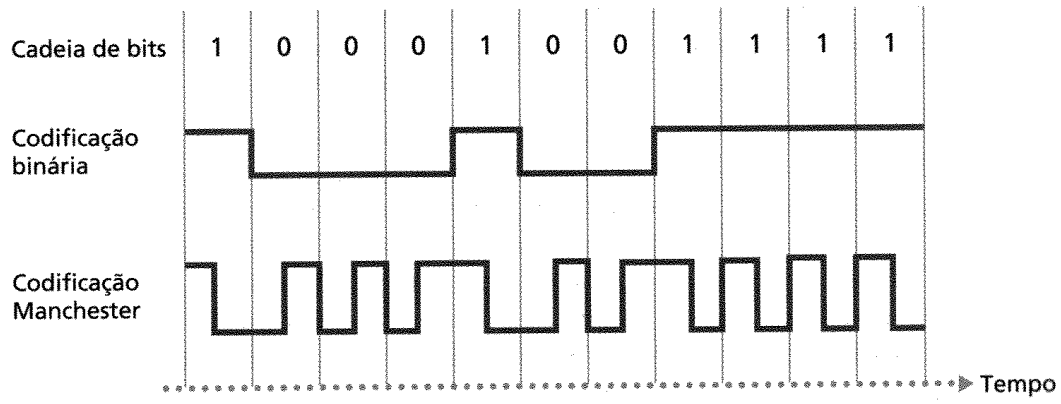
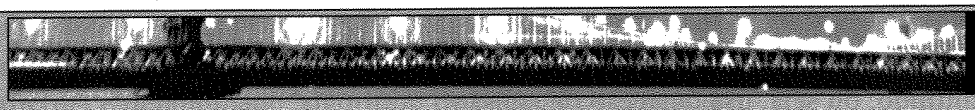


Figura 5.25 Codificação Manchester

Um serviço não confiável não orientado para conexão

Todas as tecnologias Ethernet fornecem **serviço não orientado para conexão** à camada de rede. Isto é, quando o adaptador A quer enviar um datagrama ao adaptador B, o adaptador A encapsula o datagrama em um quadro Ethernet e envia o quadro à LAN, sem se conectar previamente a B. Esse serviço de camada 2 não orientado para conexão é análogo ao serviço de datagrama de camada 3 do IP e ao serviço de camada 4 não orientado para conexão do UDP.

Todas as tecnologias Ethernet fornecem um **serviço não confiável** à camada de rede. Especificamente, quando o adaptador B recebe um quadro do adaptador A, ele submete o quadro a uma verificação de CRC, mas não envia um reconhecimento quando um quadro passa na verificação CRC nem um reconhecimento



História

Bob Metcalfe e a Ethernet

Quando era estudante de doutorado na Universidade Harvard, no início da década de 1970, Bob Metcalfe trabalhava na ARPAnet no M.I.T. Durante seus estudos, ele tomou conhecimento do trabalho de Abramson com o ALOHA e os protocolos de acesso aleatório. Após ter concluído seu doutorado e pouco antes de começar um trabalho no PARC da Xerox (Palo Alto Research Center), fez uma visita de três meses a Abramson e seus colegas da Universidade do Havaí, quando pôde observar, em primeira mão, a ALOHAnet. No PARC da Xerox, Metcalf conheceu os computadores Alto, que, em muitos aspectos, foram os predecessores dos computadores pessoais da década de 1980. Ele entendeu a necessidade de montar esses computadores em rede de um modo que não fosse dispendioso. Assim, munido com o que conhecia sobre ARPAnet, ALOHAnet e protocolos de acesso aleatório, Metcalfe, juntamente com seu colega David Boggs, inventou a Ethernet.

A Ethernet original de Metcalfe e Boggs executava a 2,94 Mbps e interligava até 256 hospedeiros a distâncias de até 1,5 km. Metcalfe e Boggs conseguiram que a maioria dos pesquisadores do PARC da Xerox se comunicassem por meio de seus computadores Alto. Então, Metcalfe forjou uma aliança entre a Xerox, a Digital e a Intel para estabelecer a Ethernet de 10 Mbps como padrão da Ethernet, ratificado pelo IEEE. A Xerox não demonstrou muito interesse em comercializar a Ethernet. Em 1979, Metcalfe abriu sua própria empresa, a 3Com, para desenvolver e comercializar tecnologia de rede, incluindo a tecnologia Ethernet. Em particular, a 3Com desenvolveu e comercializou cartões de Ethernet no início da década de 1980 para os então imensamente populares PCs da IBM. Metcalfe deixou a 3Com em 1990, quando a empresa tinha dois mil empregados e 400 milhões de dólares de receita. Até novembro de 2003, a 3Com empregava mais de 2.900 pessoas no mundo inteiro.

negativo quando um quadro não passa na verificação de CRC. Quando um quadro não passa na verificação de CRC, o adaptador B simplesmente o descarta. Assim, o adaptador A não têm a mínima idéia se o quadro que transmitiu passou na verificação CRC. Essa falta de transporte confiável (na camada de enlace) ajuda a tornar a Ethernet simples e barata. Mas também significa que a seqüência de datagramas passada à camada de rede pode ter lacunas.

Se houver lacunas devido a quadros Ethernet descartados, o protocolo de camada de aplicação do hospedeiro B também verá essas lacunas? Como aprendemos no Capítulo 3, isso depende exclusivamente de a aplicação estar usando UDP ou TCP. Se a aplicação estiver usando UDP, então a aplicação no hospedeiro B realmente sofrerá lacunas nos dados. Por outro lado, se a aplicação estiver usando TCP, então o TCP no hospedeiro B não reconhecerá os dados contidos em quadros descartados, fazendo com que o TCP no hospedeiro A retransmita. Note que, quando o TCP retransmite dados, esses dados eventualmente retornarão ao adaptador Ethernet no qual foram descartados. Assim, nesse sentido, a Ethernet realmente retransmite dados, embora não saiba se está transmitindo um datagrama novo com dados novos, ou um datagrama que contém dados que já foram transmitidos pelo menos uma vez.

5.5.2 CSMA/CD: o protocolo de acesso múltiplo da Ethernet

Quando os nós estão interconectados com um hub (e não a um comutador de camada de enlace) a LAN Ethernet é uma verdadeira LAN de *broadcast* — isto é, quando um adaptador transmite um quadro, todos os adaptadores na LAN recebem o quadro. Como pode empregar *broadcast*, a Ethernet precisa de um protocolo de acesso múltiplo — ela usa o famoso protocolo de acesso múltiplo CSMA/CD. Lembre-se de que, na Seção 5.3 mostramos que o CSMA/CD faz o seguinte:

1. Um adaptador pode começar a transmitir a qualquer tempo, ou seja, não são usados compartimentos.
2. Um adaptador nunca transmite um quadro quando percebe que algum outro adaptador está transmitindo, ou seja, ele usa detecção de portadora.
3. Um adaptador que está transmitindo aborta sua transmissão quando percebe que algum outro adaptador está transmitindo, ou seja, usa detecção de colisão.
4. Antes de tentar uma retransmissão, um adaptador espera um período de tempo aleatório que é caracteristicamente pequeno em comparação com o tempo de transmissão de um quadro.

Esses mecanismos conferem ao CSMA/CD um desempenho muito melhor, em ambiente LAN, do que o do slotted ALOHA. De fato, se o atraso máximo de propagação entre estações for muito pequeno, a eficiência do CSMA/CD poderá ficar próxima a 100 por cento. Mas note que o segundo e o terceiro mecanismos que citamos requerem que cada adaptador Ethernet seja capaz de (1) perceber quando algum outro adaptador está transmitindo e (2) detectar uma colisão enquanto estiver transmitindo. Adaptadores Ethernet realizam essas duas tarefas medindo os níveis de tensão antes e durante a transmissão.

Cada adaptador executa o protocolo CSMA/CD sem coordenação explícita com os outros adaptadores na Ethernet. Dentro de um adaptador específico, o protocolo CSMA/CD funciona como segue:

1. O adaptador obtém um datagrama de camada de rede de seu nó pai, prepara um quadro Ethernet e o coloca no buffer do adaptador.
2. Se o adaptador percebe que o canal está ocioso (isto é, não há nenhuma energia de sinal entrando do canal para o adaptador durante 96 tempos de bits), ele começa a transmitir o quadro. Se o adaptador percebe que o canal está ocupado, espera até perceber que não há nenhuma energia de sinal (mais 96 tempos de bits) e então começa a transmitir o quadro.
3. Enquanto transmite, o adaptador monitora a presença de energia de sinal vinda de outros adaptadores. Se o adaptador transmitir o quadro inteiro sem detectar energia de sinal vinda de outros adaptadores, ele considerará concluída a operação com o quadro.

4. Se o adaptador detectar energia de sinal vinda de outros adaptadores enquanto está transmitindo, ele parará de transmitir seu quadro e, em vez dele, transmitirá um sinal de reforço de colisão de 48 bits.
5. Após abortar (isto é, transmitir o sinal de reforço), o adaptador entra em fase de **backoff exponencial**. Especificamente, quando está transmitindo um dado quadro, após sofrer a enésima colisão em seguida para esse quadro, o adaptador escolhe um valor para K aleatoriamente de $\{0,1,2, \ldots, 2^m - 1\}$, onde $m := \min(n,10)$. O adaptador estão espera $K \cdot 512$ tempos de bits e então retorna à etapa 2.

Alguns comentários adicionais sobre o protocolo CSMA/CD se fazem necessários. A finalidade do sinal de reforço de colisão é garantir que todos os outros adaptadores que estejam transmitindo fiquem cientes da colisão. Vamos ver um exemplo. Suponha que o adaptador A comece a transmitir um quadro e, exatamente antes de o sinal de A alcançar o adaptador B, o adaptador B comece a transmitir. Assim, B terá transmitido apenas alguns bits quando abortar sua transmissão. Esses poucos bits, na verdade, propagar-se-ão até A, mas podem não constituir energia suficiente para que A detecte a colisão. Para garantir que A detecte a colisão (de modo que ele também possa abortar), B transmite o sinal de reforço de colisão de 48 bits.

Em seguida, considere o algoritmo de backoff exponencial. A primeira coisa a notar aqui é que um tempo de bit (isto é, o tempo para transmitir um único bit) é muito curto; para uma Ethernet de 10 Mbps, um tempo de bit é 0,1 microssegundo. Agora, vamos considerar um exemplo. Suponha que um adaptador tente transmitir um quadro pela primeira vez e que, enquanto está transmitindo, detecte uma colisão. O adaptador então escolhe $K = 0$ com probabilidade de 0,5 ou escolhe $K = 1$ com probabilidade de 0,5. Se o adaptador escolher $K = 0$, imediatamente passará para a etapa 2 após transmitir o sinal de reforço de colisão. Se escolher $K = 1$, ele esperará 51,2 microssegundos antes de voltar à etapa 2. Após uma segunda colisão, K é escolhido com igual probabilidade no conjunto $\{0,1,2,3\}$. Após três colisões, K é escolhido com igual probabilidade no conjunto $\{0,1,2,3,4,5,6,7\}$. Após dez ou mais colisões,* K é escolhido com igual probabilidade no conjunto $\{0,1,2,\ldots,1023\}$. Assim, o tamanho dos conjuntos dos quais K é escolhido cresce exponencialmente com o número de colisões (até $n = 10$); é por essa razão que o algoritmo de backoff da Ethernet é denominado 'algoritmo de backoff exponencial'.

O padrão Ethernet impõe limites à distância entre dois nós quaisquer. Esses limites garantem que, se o adaptador A escolher um valor de K mais baixo do que escolheram todos os outros adaptadores envolvidos em uma colisão, ele poderá transmitir seu quadro sem sofrer uma nova colisão. Estudaremos essa propriedade mais detalhadamente nos exercícios ao final deste capítulo.

Por que usar backoff exponencial? Por que, por exemplo, não escolher um K entre $\{0,1,2,3,4,5,6,7\}$ após cada colisão? A razão é que, quando um adaptador sofre sua primeira colisão, ele não tem idéia de quantos adaptadores estão envolvidos na colisão. Se houver somente um pequeno número de adaptadores colidindo, terá sentido escolher K de um pequeno conjunto de valores pequenos. Por outro lado, se muitos adaptadores estiverem envolvidos na colisão, terá sentido escolher K dentro de um conjunto de valores maiores e mais dispersos (por quê?). Aumentando o tamanho do conjunto após cada colisão, o adaptador se ajusta adequadamente a esses diferentes cenários.

Mencionamos também que cada vez que um adaptador prepara um novo quadro para transmissão, ele executa o algoritmo CSMA/CD que acabamos de apresentar. Em particular, o adaptador não leva em conta nenhuma colisão que possa ter ocorrido no passado recente. Assim, é possível que um adaptador com um novo quadro consiga uma transmissão imediatamente enquanto vários outros adaptadores estão no estado de backoff exponencial.

* Após dez tentativas, o adaptador descarta o quadro. (N. do R.T.)

Eficiência da Ethernet

Quando apenas um nó tem um quadro para enviar, ele pode transmitir à velocidade total da tecnologia Ethernet (10 Mbps, 100 Mbps ou 1 Gbps). Contudo, se muitos nós tiverem quadros para transmitir, a velocidade efetiva de transmissão do canal poderá ser muito menor. Definimos a **eficiência da Ethernet** como a fração final de tempo durante a qual os quadros estão sendo transmitidos no canal, sem colisões, quando há um grande número de nós ativos, e cada nó tem um grande número de quadros a enviar. Para uma aproximação da eficiência da Ethernet, vamos designar como t_{prop} o tempo máximo para que a energia de sinal se propague entre dois adaptadores quaisquer. Seja t_{trans} o tempo que leva para transmitir um quadro Ethernet de tamanho máximo (aproximadamente 1,2 milissegundo para uma Ethernet de 10 Mbps). Uma derivação da eficiência da Ethernet ultrapassa o escopo deste livro — para mais detalhes, consulte [Lam, 1980] e [Bertsekas, 1991]. Nesta subseção, vamos apenas mencionar a seguinte aproximação:

$$\text{Eficiência} = \frac{1}{1 + 5t_{prop}/t_{trans}}$$

Vemos por essa fórmula que, quando t_{prop} se aproxima de 0, a eficiência se aproxima de 1. Isso está de acordo com a intuição que temos de que, se o atraso de propagação for 0, os nós em colisão vão abortar imediatamente sem desperdiçar o canal. Além disso, à medida que t_{trans} vai ficando muito grande, a eficiência se aproxima de 1. Isso também é intuitivo, porque, quando um quadro captura o canal, vai prendê-lo por muito tempo; assim, o canal estará realizando um trabalho produtivo durante a maior parte do tempo.

5.5.3 Tecnologias Ethernet

Em 2004, as tecnologias Ethernet mais comuns eram a 10BaseT e a 100BaseT, que usam pares de fios trançados de cobre em topologia estrela e têm velocidades de transmissão de 10 Mbps e 100 Mbps, respectivamente. Essas tecnologias Ethernet estão padronizadas pelos grupos de trabalho IEEE 802.3. Por essa razão, uma LAN Ethernet quase sempre é denominada LAN 802.3.

A Figura 5.26 ilustra as tecnologias 10BaseT e 100BaseT. Cada adaptador em cada nó tem uma conexão direta, ponto-a-ponto com o hub. Essa conexão consiste em dois pares de fios trançados de cobre, um para transmitir e o outro para receber. Na extremidade da conexão, há um conector RJ-45, parecido com o conector RJ-11 usado para telefones comuns. O 'T' em 10BaseT e em 100BaseT quer dizer 'par de fios trançados'. Tanto para a 10BaseT quanto para a 100BaseT, o comprimento máximo da conexão entre um adaptador e o hub é 100 metros; assim, a distância máxima entre quaisquer dois nós é 200 metros. Como discu-

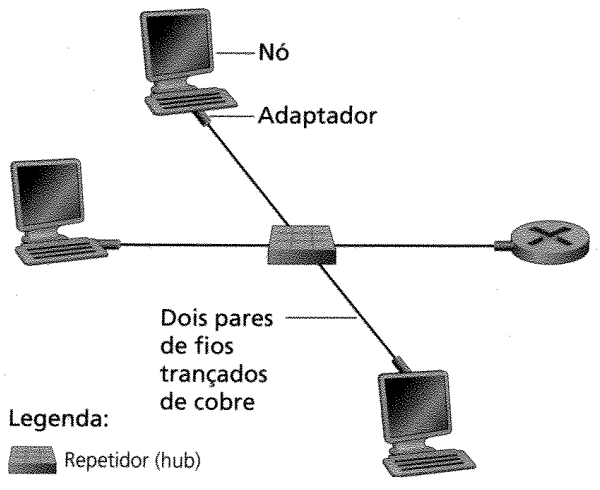


Figura 5.26 Topologia em estrela para 10BaseT e 100BaseT

tiremos na próxima seção, essa distância máxima pode ser aumentada distribuindo hubs ou comutadores em camadas e por enlaces de fibra.

Um **repetidor** (hub) é um dispositivo de camada física que atua sobre bits individuais, em vez de sobre quadros, e tem duas ou mais interfaces. Quando um bit, representando um 0 ou um 1, chega de uma interface, o repetidor simplesmente recria o bit, reforça sua energia e transmite o bit para todas as outras interfaces. É importante ter em mente que repetidores não implementam detecção de portadora ou qualquer outra parte do CSMA/CD; um repetidor repete um bit que está entrando em todas as interfaces de saída mesmo que haja energia de sinal em algumas dessas interfaces. Como repetidores fazem transmissão *broadcast* de bits, cada adaptador em uma Ethernet 10/100 BaseT pode (1) sondar o canal para determinar se ele está ocioso e (2) detectar uma colisão enquanto está transmitindo.

Repetidores também oferecem características de gerenciamento de rede. Por exemplo, se um adaptador está funcionando mal e envia continuamente quadros Ethernet (denominado adaptador tagarela [jabbering]), a Ethernet 10/100BaseT continuará a funcionar, porque o hub detectará o problema e desconectará internamente o adaptador que está funcionando mal. Com essa característica, o administrador de rede não tem de sair da cama e voltar a seu local de trabalho para corrigir o problema. Além disso, a maioria dos hubs pode reunir informações e repassá-las para um hospedeiro que está conectado diretamente ao hub. Como discutiremos no Capítulo 9 (Gerenciamento de Rede), esse hospedeiro monitor provê uma interface gráfica que apresenta dados estatísticos e grafos, como utilização de largura de banda, índices de colisão, tamanhos médios de quadros e assim por diante. Os administradores de rede podem usar essas informações não só para depurar e corrigir problemas, mas também para planejar como a LAN deverá evoluir no futuro.

Hoje, muitos adaptadores Ethernet são adaptadores de 10/100 Mbps. Isso significa que podem ser usados tanto para a tecnologia 10BaseT como para a 100BaseT. A 100BaseT caracteristicamente utiliza par trançado de categoria 5 (um par trançado de alta qualidade com muitas tranças). Diferentemente da rede 10BaseT, a rede 100BaseT não usa a codificação Manchester, mas uma codificação mais eficiente, denominada 4B5B: todo grupo de cinco períodos de relógio é usado para enviar 4 bits com a finalidade de fornecer transições suficientes para permitir sincronização de relógio.

Mencionamos brevemente aqui que as tecnologias Ethernet 10 Mbps e 100 Mbps podem empregar enlaces de fibra. Um enlace de fibra é freqüentemente utilizado para interconectar hubs que estão em prédios diferentes no mesmo campus. A fibra é cara devido ao custo de seus conectores, mas tem uma excelente imunidade a ruído. Os padrões IEEE 802 permitem que uma LAN tenha um alcance geográfico mais amplo quando se usa fibra para conectar os nós do backbone.

Gigabit Ethernet e Ethernet 10Gbps

A rede Gigabit Ethernet é uma extensão dos muito bem-sucedidos padrões 10 Mbps e 100 Mbps. Oferecendo uma velocidade bruta de 1000 Mbps, a Gigabit Ethernet mantém total compatibilidade com a imensa base instalada de equipamentos Ethernet. A Gigabit Ethernet apresenta o seguinte padrão de funcionamento, denominado IEEE 802.3z:*

- Usa o formato-padrão do quadro Ethernet (Figura 5.24) e é compatível com as tecnologias 10BaseT e 100BaseT. Isso permite fácil integração da Gigabit Ethernet com a base instalada de equipamentos Ethernet.
- Permite enlaces ponto-a-ponto, bem como canais *broadcast* compartilhados. Enlaces ponto-a-ponto usam comutadores (veja a Seção 5.6), ao passo que canais *broadcast* usam hubs, como descrito anteriormente para a 10BaseT e para a 100BaseT. No jargão da Gigabit Ethernet, os hubs são denominados *distribuidores com buffer.*

* O padrão mais importante é o 802.3ab. (N. do R.T.)

- Utiliza CSMA/CD para canais *broadcast* compartilhados. Para conseguir eficiência aceitável, a distância máxima entre os nós deve ser severamente limitada.
- Permite operação *full-duplex* a 1.000 Mbps em ambas as direções para canais ponto-a-ponto.

Como a 10BaseT e a 100BaseT, a Gigabit Ethernet tem topologia em estrela, tendo como centro um hub ou um comutador. (Comutadores Ethernet serão discutidos na Seção 5.6.) A Gigabit Ethernet serve freqüentemente como backbone para interconectar múltiplas LANs Ethernet de 10 Mbps e de 100 Mbps. Operando inicialmente sobre fibra ótica, a Gigabit Ethernet também pode usar cabeamento UTP categoria 5.

A 10-Gigabit Ethernet ampliou ainda mais a popular tecnologia Ethernet com produtos que surgiram em 2001. Além disso, o padrão da 10-Gigabit Ethernet, 802.3ae, estende a tecnologia Ethernet aos enlaces ponto-a-ponto de WANs. Consulte o site Web de Spurgeon [Spurgeon, 2004] sobre Ethernet se quiser boas informações e links para Gigabit Ethernet e 10-Gigabit Ethernet.

5.6 Interconexões: hubs e comutadores

Instituições — incluindo empresas, universidades e escolas — são, em geral, constituídas de muitos departamentos; cada departamento tem e administra sua própria LAN Ethernet. Naturalmente, uma instituição vai querer que seus departamentos interconectem seus segmentos de LANs departamentais. Nesta seção, consideraremos duas abordagens para conectar LANs: hubs e comutadores.* Ambas são bastante usadas atualmente.

5.6.1 Hubs

O modo mais simples de interconectar LANs é utilizar hubs. A Figura 5.27 mostra como três departamentos de uma universidade podem interconectar suas LANs. Nessa figura, cada um dos três departamentos tem uma Ethernet 10BaseT que fornece acesso à rede ao corpo acadêmico, ao pessoal e aos estudantes do departamento. Cada hospedeiro em um departamento tem uma conexão ponto-a-ponto com o hub do departamento. Um quarto hub, denominado **hub de backbone**, tem conexões ponto-a-ponto com os hubs dos departamentos, interconectando as LANs dos três departamentos. O desenho mostrado na Figura 5.27 é um **projeto de hub multinível**, porque os hubs são arranjados hierarquicamente. Também é possível criar projetos multiníveis com mais de dois níveis — por exemplo, um nível para os departamentos, um nível para as escolas dentro da universidade (escola de engenharia, escola de administração etc.) e um nível mais alto para o ambiente universitário.

Em um projeto multinível, referimo-nos à rede interconectada como uma LAN e a cada uma das parcelas departamentais da LAN (isto é, a cada hub departamental e aos hospedeiros conectados a ele) como um **segmento de LAN**. É importante notar que todos os segmentos de LAN da Figura 5.27 pertencem ao mesmo domínio de colisão, isto é, sempre que um ou mais nós nos segmentos de LAN transmitem ao mesmo tempo, há colisão e todos os nós transmissores entram em backoff exponencial.

Uma LAN departamental interconectada a um hub de backbone tem muitos benefícios. Em primeiro lugar, e mais importante, fornece comunicação interdepartamental entre os hospedeiros dos vários departamentos. Em segundo lugar, amplia a distância máxima entre qualquer par de nós da LAN. Por exemplo, com uma 10BaseT, a distância máxima entre um nó e seu hub é 100 metros; portanto, para um único segmento de LAN, a distância máxima entre qualquer par de nós é 200 metros. Interconectando os hubs, essa distância máxima pode ser aumentada, já que a distância entre hubs diretamente conectados também pode ser de 100 metros quando se usa par trançado (e maior ainda quando se usa fibra). Um terceiro benefício é que o projeto multinível provê um certo grau de degradação suave. Especificamente, se qualquer um dos hubs departamentais começar a funcionar mal, o hub de backbone poderá detectar o problema e desconectar o

* A expressão 'comutador' é empregada aqui como tradução de 'switch'. (N. do R. T.)

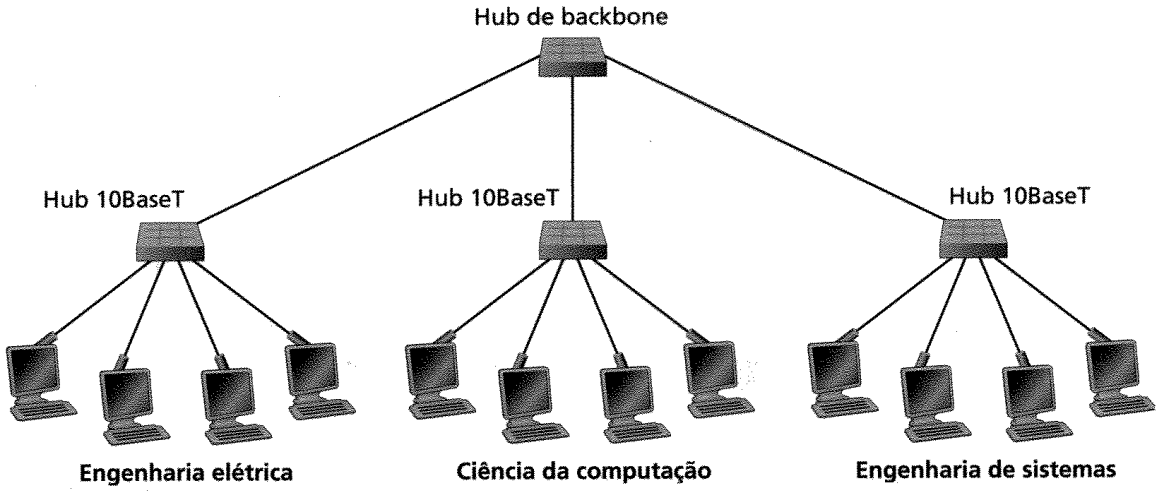


Figura 5.27 Três Ethernets departamentais interconectadas com um hub

hub departamental da LAN; desse modo, os departamentos restantes podem continuar a funcionar e a se comunicar enquanto o hub departamental defeituoso é consertado.

Embora um hub de backbone seja um dispositivo de interconexão útil, ele tem três sérias limitações que dificultam sua disseminação. A primeira limitação, e talvez a mais importante, é que quando LANs departamentais estão interconectadas por um hub, os domínios de colisão dos departamentos (anteriormente independentes) se transformam em um grande domínio de colisão comum. Vamos estudar esse assunto no contexto da Figura 5.27. Antes de interconectar os três departamentos, cada LAN departamental tinha uma vazão máxima de 10 Mbps, de modo que a vazão agregada máxima das três LANs era de 30 Mbps. Mas, assim que as três LANs são interconectadas por um hub, todos os hospedeiros dos três departamentos passam a pertencer ao mesmo domínio de colisão e a vazão agregada máxima fica reduzida a 10 Mbps.

Uma segunda limitação é que, se os vários departamentos usarem tecnologias Ethernet diferentes, então poderá ser impossível interconectar os hubs departamentais a um hub de backbone. Por exemplo, se alguns departamentos usam 10BaseT e os departamentos restantes usam 100BaseT, é impossível interconectar todos os departamentos sem fazer algum buffer de quadros no ponto de interconexão; uma vez que um hub é, essencialmente, um repetidor que não armazena quadros, ele não pode interconectar segmentos de LAN que funcionam em velocidades diferentes.

Uma terceira limitação é que cada uma das tecnologias Ethernet (10Base2, 10BaseT, 100BaseT e assim por diante) tem restrições quanto ao número máximo de nós permitidos em um domínio de colisão, quanto à distância máxima entre dois nós dentro de um domínio de colisão e quanto ao número máximo de níveis permitidos em um projeto multinível. Essas restrições limitam tanto o número total de hospedeiros que podem se conectar a uma LAN multinível quanto o alcance geográfico de uma LAN multinível.

5.6.2 Comutadores de camada de enlace

Em contraste com os hubs, que são dispositivos de camada física, comutadores de camada de enlace — denominados simplesmente **comutadores** (switches) — agem sobre quadros Ethernet e, portanto, são dispositivos de camada 2. De fato, como são comutadores de pacotes totalmente capacitados, eles repassam quadros com base em endereços da LAN de destino. Quando um quadro chega à interface de um comutador, o comutador examina o endereço de destino de camada 2 do quadro e tenta repassá-lo para a interface que leva a esse destino.

A Figura 5.28 mostra como os três departamentos acadêmicos de nosso exemplo anterior poderiam se interconectar usando um comutador. Os três números próximos ao comutador são os números de suas três interfaces. Quando os departamentos estão interconectados por um comutador, como na Figura 5.28, nova-

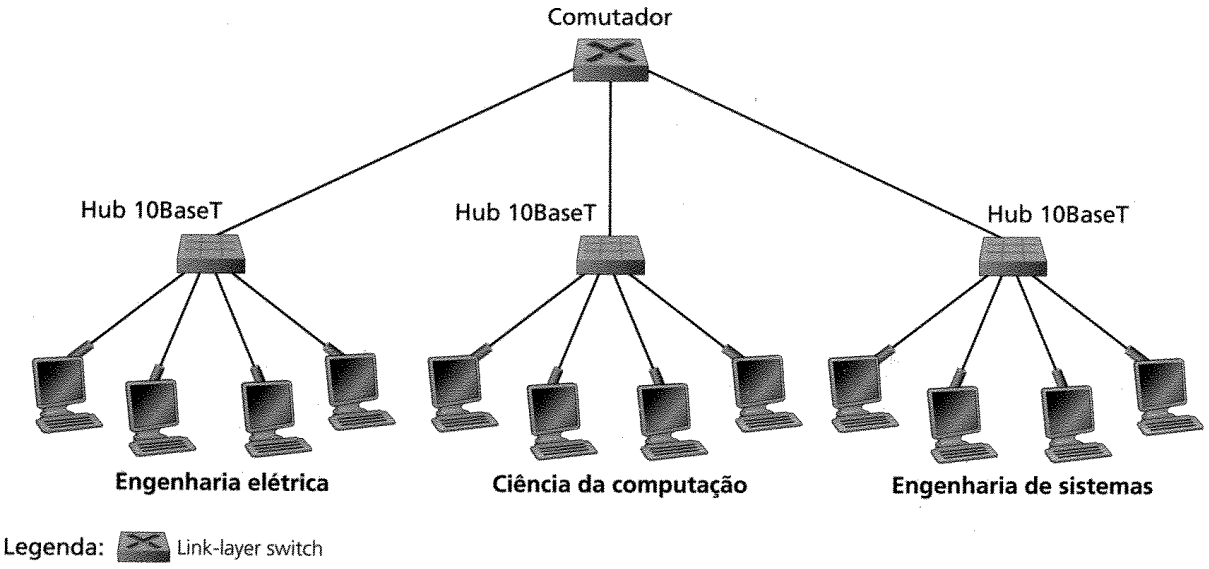


Figura 5.28 Três LANs departamentais interconectadas com um comutador

mente nos referimos à rede interconectada inteira como uma LAN e, mais uma vez, nos referimos a cada uma das parcelas departamentais da rede como segmentos de LAN. Mas, em contraste com o projeto de hub multinível da Figura 5.27, cada segmento de LAN agora é um domínio de colisão isolado.

Comutadores podem superar muitos problemas que atormentam os hubs. Primeiramente, eles permitem comunicação interdepartamental, preservando, ao mesmo tempo, domínios de colisão isolados para cada um dos departamentos. Em segundo lugar, eles podem interconectar diferentes tecnologias de LAN, incluindo as Ethernets 10BaseT, 100BaseT e a Gigabit Ethernet. Em terceiro lugar, não há limites ao tamanho possível de uma LAN quando são usados comutadores para interconectar segmentos de LAN; teoricamente, utilizando comutadores, é possível montar uma LAN que abranja o globo terrestre inteiro. E também, como discutiremos no final desta seção, comutadores funcionam em *full-duplex* e fornecem comutação acelerada (*cut-through switching*).

A Figura 5.29 mostra como uma instituição com diversos departamentos e vários servidores críticos poderia disponibilizar uma combinação de hubs, comutadores e roteadores. Nessa figura, cada um dos três departamentos tem seu próprio segmento Ethernet de 10 Mbps com seu próprio hub. Como cada hub departamental tem uma conexão com o comutador, todo o tráfego intradepartamental fica confinado no segmento Ethernet do departamento. Cada servidor Web e de correio tem acesso dedicado de 100 Mbps com o comutador. Note que esse comutador tem, no mínimo, três interfaces de 10 Mbps e três interfaces de 100 Mbps.

Repasse e filtragem por comutadores

Filtragem é a capacidade de um comutador determinar se um quadro deve ser repassado para alguma interface ou se deve apenas ser descartado. **Repasse** é a capacidade de determinar as interfaces para as quais um quadro deve ser dirigido e então dirigir o quadro a essas interfaces. Filtragem e repasse por comutadores são feitos com uma **tabela de comutação**. A tabela de comutação contém registros para alguns nós da LAN, mas não necessariamente para todos. Um registro de um nó na tabela de comutação contém (1) o endereço MAC do nó, (2) a interface do comutador que leva em direção ao nó e (3) o horário em que o registro para o nó foi colocado na tabela. Um exemplo de tabela de comutação para a LAN da Figura 5.28 é mostrado na Figura 5.30. Embora essa descrição de repasse de quadros possa parecer semelhante à nossa discussão de repasse de datagramas no Capítulo 4, veremos brevemente que há diferenças significativas. Observamos aqui que os endereços usados por comutadores são endereços MAC, e não endereços de camada de rede. Também veremos brevemente que a montagem de uma tabela de comutação é muito diferente da montagem de tabelas de roteamento.

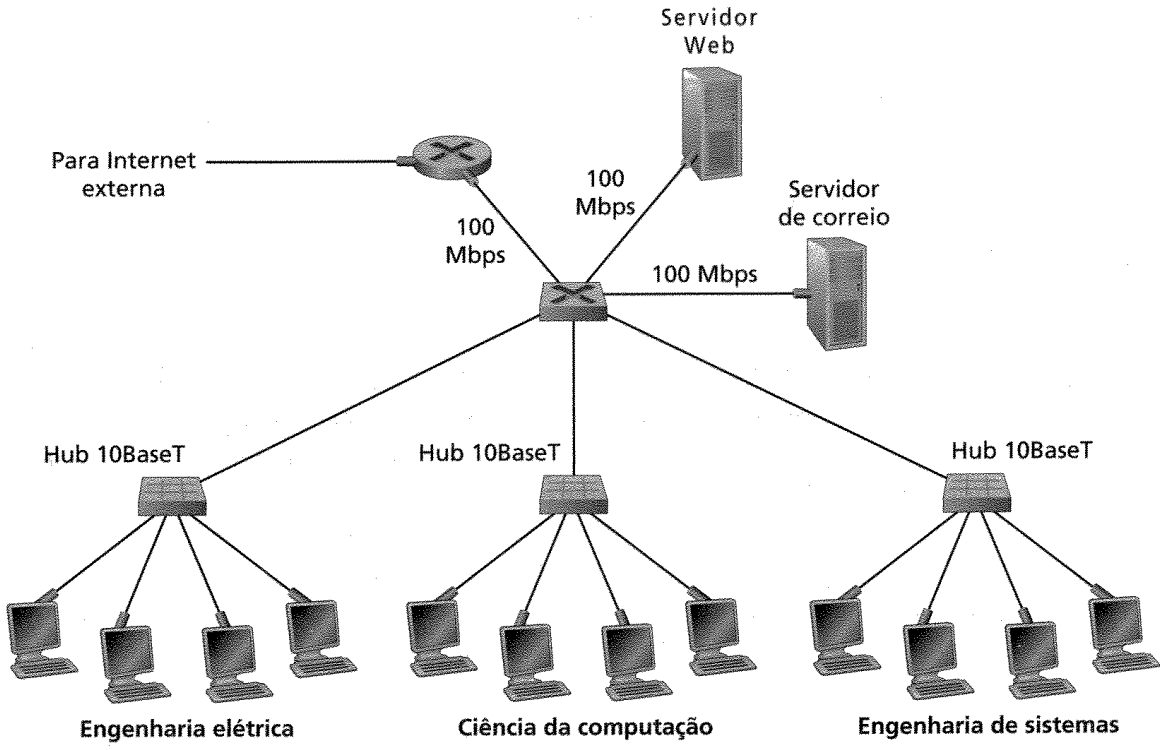


Figura 5.29 Uma rede institucional que utiliza uma combinação de hubs, comutadores Ethernet e um roteador

Para entender como funciona a filtragem e o repasse por comutadores, suponha que um quadro com endereço de destino DD-DD-DD-DD-DD-DD chegue ao comutador na interface *x*. O comutador indexa sua tabela com o endereço MAC DD-DD-DD-DD-DD-DD e descobre sua interface correspondente *y*, que é conhecida como aquela que conduz ao endereço destinatário DD-DD-DD-DD-DD-DD. (Veremos brevemente o que acontece se o endereço DD-DD-DD-DD-DD-DD não estiver na tabela.)

- Se *x* for igual a *y*, então o quadro estará vindo de um segmento de LAN que contém o adaptador DD-DD-DD-DD-DD-DD. Não havendo necessidade de repassar o quadro a qualquer outra das interfaces, o comutador realiza a função de filtragem descartando o quadro.
- Se *x* não for igual a *y*, então o quadro precisará ser repassado para o segmento de LAN ligado à interface *y*. O comutador realiza sua função de repasse colocando o quadro em um buffer de saída que precede a interface *y*.

Essas regras simples permitem que um comutador preserve domínios de colisão separados para cada um dos diferentes segmentos de LAN conectados a suas interfaces. Elas também permitem que dois conjuntos de nós de diferentes segmentos de LAN se comuniquem simultaneamente sem interferência mútua.

Vamos examinar essas regras para a rede da Figura 5.28 e sua tabela de comutação na Figura 5.30. Suponha que um quadro com endereço de destino 62-FE-F7-11-89-A3 chegue ao comutador vindo da interface 1. O comutador examina sua tabela e vê que o destino está no segmento de LAN conectado à interface 1 (isto é, à LAN do departamento de engenharia elétrica). Isso significa que o quadro já foi transmitido por *broadcast* no segmento de LAN que contém o destino. Por conseguinte, o comutador filtra (isto é, descarta) o quadro. Agora suponha que um quadro com o mesmo endereço de destino chegue da interface 2. O comutador novamente examina sua tabela e verifica que o destino está na direção da interface 1; por conseguinte, ele repassa o quadro para o buffer de saída que precede a interface 1. Deve ficar claro, por esse exemplo, que, enquanto a tabela de comutação permanecer completa e precisa, o comutador isolará os domínios de colisão departamentais, permitindo que os departamentos se comuniquem.

Endereço	Interface	Horário
62–FE–F7–11–89–A3	1	9h32
7C–BA–B2–B4–91–10	3	9h36
....	....	....

Figura 5.30 Parte de uma tabela de comutador para a LAN da Figura 5.28

Hubs versus comutadores

Lembre-se de que, quando um hub repassa um quadro para um enlace, ele apenas envia os bits para o enlace sem se preocupar em sondar se há outra transmissão ocorrendo ao mesmo tempo no enlace. Ao contrário, quando um comutador quer repassar um quadro para um enlace, ele executa o algoritmo CSMA/CD discutido na Seção 5.3. Em particular, o comutador se abstém de transmitir se percebe que há algum outro nó transmitindo no segmento de LAN para o qual quer enviar um quadro. Além disso, usa backoff exponencial quando uma de suas transmissões resulta em colisão. Assim, interfaces de comutador se comportam de modo muito semelhante a adaptadores de nós. Mas, em termos técnicos, comutadores não são adaptadores de nós, porque nem eles nem suas interfaces têm endereços MAC. Lembre-se de que um adaptador de nó (isto é, um hospedeiro ou um roteador) sempre insere seu endereço MAC no endereço de fonte de todos os quadros que transmite. Um comutador, por outro lado, não troca o endereço de fonte do quadro.

Uma característica significativa dos comutadores é que eles podem ser usados para combinar segmentos Ethernet que usam diferentes tecnologias. Por exemplo, se na Figura 5.28 o departamento de engenharia elétrica tiver uma Ethernet 10Base2, o de ciência da computação uma Ethernet 100BaseT e o de engenharia de sistemas uma Ethernet 10BaseT, é possível comprar um comutador para interconectar as três LANs. Com comutadores Gigabit Ethernet é possível ter uma conexão adicional de 1 Gbps com um roteador que, por sua vez, conecta-se a uma rede universitária de maior porte. Como mencionamos anteriormente, essa capacidade de poder interconectar enlaces de velocidades diferentes não se consegue com os hubs.

Além disso, quando são usados comutadores como dispositivos de interconexão, não há limite teórico para o alcance geográfico de uma LAN. Teoricamente, podemos montar uma LAN que abranja o globo terrestre interconectando hubs em uma longa topologia linear, com cada par de hubs vizinhos interconectado por um comutador. Com esse projeto, cada um dos hubs tem seu próprio domínio de colisão e não há limite para o comprimento da LAN. No entanto, em breve veremos que não é bom montar redes muito grandes usando exclusivamente comutadores como dispositivos de interconexão — redes de grande porte necessitam de roteadores também.

Aprendizagem automática

Um comutador tem a maravilhosa propriedade (em especial, para o administrador de rede, que quase sempre está sobrecarregado) de montar sua tabela automática, dinâmica e autonomamente — sem nenhuma intervenção de um administrador de rede ou de um protocolo de configuração. Em outras palavras, comutadores são **autodidatas**. Essa capacidade é conseguida como segue:

1. A tabela de comutação inicialmente está vazia.
2. Quando um quadro chega a uma das interfaces e o endereço de destino do quadro não está na tabela, o comutador repassa cópias do quadro para os buffers de saída que precedem todas as outras interfaces. (Em cada uma dessas outras interfaces, o quadro é transmitido para o segmento de LAN usando CSMA/CD.)
3. Para cada quadro recebido em uma interface, o comutador armazena em sua tabela (1) o endereço MAC que está no *campo de endereço de fonte* do quadro, (2) a interface da qual veio o quadro e (3) o horário corrente. Dessa maneira, o comutador registra em sua tabela o segmento MAC no qual reside o nó remetente. Se cada nó da LAN eventualmente enviar um quadro, então cada nó será eventualmente registrado na tabela.

Endereço	Interface	Horário
01–12–23–34–45–56	2	9h39
62–FE–F7–11–89–A3	1	9h32
7C–BA–B2–B4–91–10	3	9h36
....	....	....

Figura 5.31 O comutador aprende a localização do adaptador com endereço 01-12-23-34-45-56

4. Quando um quadro chega em uma das interfaces e o endereço de destino do quadro está na tabela, o comutador repassa o quadro para a interface apropriada.
5. O comutador apagará um endereço na tabela se nenhum quadro que tenha aquele endereço como endereço de fonte for recebido após um certo período de tempo (o **tempo de envelhecimento**). Desse modo, se um PC for substituído por outro PC (com um adaptador diferente), o endereço MAC do PC original acabará sendo expurgado da tabela de comutação.

Vamos examinar a propriedade de aprendizagem automática para a rede da Figura 5.28 e sua tabela de comutação correspondente apresentada na Figura 5.30. Suponha que no horário 9h39 um quadro com endereço de fonte 01-12-23-34-45-56 venha da interface 2. Suponha também que esse endereço não esteja na tabela de comutação. Então, o comutador anexa um novo registro à tabela, como mostra a Figura 5.31.

Continuando com esse mesmo exemplo, suponha ainda que o tempo de envelhecimento para esse comutador seja 60 minutos e que nenhum quadro com endereço de fonte 62-FE-F7-11-89-A3 chegue ao comutador entre 9h32 e 10h32. Então, no horário 10h32, o comutador remove esse endereço de sua tabela.

Comutadores são **dispositivos do tipo plug-and-play** porque não requerem a intervenção de um administrador de rede ou de um usuário. Um administrador de rede que quiser instalar um comutador não precisa fazer nada mais do que conectar os segmentos de LAN às interfaces do comutador. O administrador não precisa configurar as tabelas de comutação na hora da instalação nem quando um hospedeiro é removido de um dos segmentos de LAN.

Acesso dedicado e full-duplex

Um comutador com um grande número de interfaces facilita conexões diretas entre hospedeiros e o comutador. Quando um hospedeiro tem uma ligação direta com um comutador (e não uma conexão LAN compartilhada), dizemos que ele tem **acesso dedicado**. Na Figura 5.32, um comutador fornece acesso dedicado a seis hospedeiros.

Um comutador e os hospedeiros diretamente conectados a ele funcionam no modo *full-duplex*. Vejamos como isso acontece. Para levar a discussão para um terreno concreto, suponha que cada conexão na rede da Figura 5.32 usa dois pares de fios trançados de cobre (como em 10BaseT e 100BaseT), um par para transmitir do hospedeiro ao comutador e o outro par para transmitir do comutador ao hospedeiro. Por causa do acesso dedicado, quando um hospedeiro A transmite um quadro pelo par de fios trançados na direção do comutador, não há possibilidade de o quadro colidir com uma transmissão de qualquer outro hospedeiro ou do comutador. De modo semelhante, como comutadores 'armazenam e repassam', o comutador transmitirá no máximo um quadro por vez por qualquer um dos pares trançados na direção do hospedeiro. Assim, com conexões diretas do comutador aos hospedeiros e dos hospedeiros ao comutador, não são necessárias nem detecção de colisão, nem detecção de portadora. Na verdade, cada enlace torna-se um enlace ponto-a-ponto, evitando a necessidade de qualquer protocolo de acesso ao meio. Desse modo, com a desativação em cada adaptador da detecção de portadora, da detecção de colisão e do retorno de dados transmitidos para a entrada do receptor, é criado um canal *full-duplex* entre cada hospedeiro e seu comutador. Por exemplo, na Figura 5.32, o hospedeiro A pode enviar um arquivo a A enquanto B está enviando um arquivo a B' e C está enviando um arquivo a C'. Se cada um dos hospedeiros tiver uma placa de adaptador de 10 Mbps, então a vazão agregada durante as três transferências simultâneas de arquivos será 30 Mbps. Se A e A' tiverem adap-

tadores de 100 Mbps e os hospedeiros restantes tiverem adaptadores de 10 Mbps, então a vazão agregada durante as três transferências simultâneas de arquivos será 120 Mbps.

Comutação acelerada (cut-through switching)

Além do grande número de interfaces, suporte para uma infinidade de tipos de meios físicos e taxas de transmissão e características atraentes de gerenciamento de rede, os fabricantes de comutadores Ethernet freqüentemente anunciam que seus comutadores usam **comutação acelerada** (cut-through), e não comutação de pacotes do tipo armazenagem e repasse (store-and-forward), usada por roteadores. A diferença entre comutação acelerada e de armazenagem e repasse é sutil. Para entendê-la, considere um pacote que está sendo repassado através de um comutador de pacotes (isto é, um roteador, ou um comutador). Como discutimos na Seção 4.4, o pacote chega ao comutador em uma porta de entrada e sai do comutador em uma porta de saída. Na porta de saída pode haver ou não outros pacotes enfileirados no buffer da porta de saída. Quando há pacotes no buffer de saída, não há absolutamente nenhuma diferença entre comutação de armazenagem e repasse e comutação acelerada. As duas técnicas somente são diferentes quando o buffer de saída está vazio.

Lembre-se de que dissemos no Capítulo 1 que, quando um pacote é repassado através de um comutador de pacotes de armazenagem e repasse, ele é primeiramente coletado e armazenado integralmente antes que o comutador comece a transmiti-lo por seu enlace de saída. Quando o buffer de saída fica vazio antes de o pacote inteiro chegar ao comutador, essa coleta gera um atraso de armazenagem e repasse no comutador — um atraso que contribui para o atraso total fim-a-fim (veja a Seção 1.6). Um limite máximo para esse atraso é L/R, onde L é o comprimento do pacote e R é a taxa de transmissão no enlace de entrada. Note que um pacote somente fica sujeito a um atraso de armazenagem e repasse se o buffer de saída fica vazio antes de o pacote inteiro chegar ao comutador, do contrário o atraso é maior.

Com comutação acelerada, se o buffer ficar vazio antes de o pacote inteiro chegar, o comutador poderá começar a transmitir a parte da frente do pacote enquanto a parte traseira dele continua a chegar. É claro que, antes de transmitir o pacote pelo enlace de saída, deve chegar a porção do pacote que contém o endereço de destino. (Esse pequeno detalhe é inevitável para todos os tipos de comutadores, pois é o comutador que deve determinar o enlace de saída apropriado.) Em resumo, com comutação acelerada, um pacote não precisa ser completamente armazenado antes de ser repassado; em vez disso, ele é repassado através do comutador quando o enlace de saída está livre. Se o enlace de saída é uma rede de acesso múltiplo compartilhada com outros hospedeiros (por exemplo, o enlace de saída se conecta com um hub), então o comutador também tem de sondar se o enlace está ocioso antes de iniciar a retransmissão do pacote por comutação acelerada.

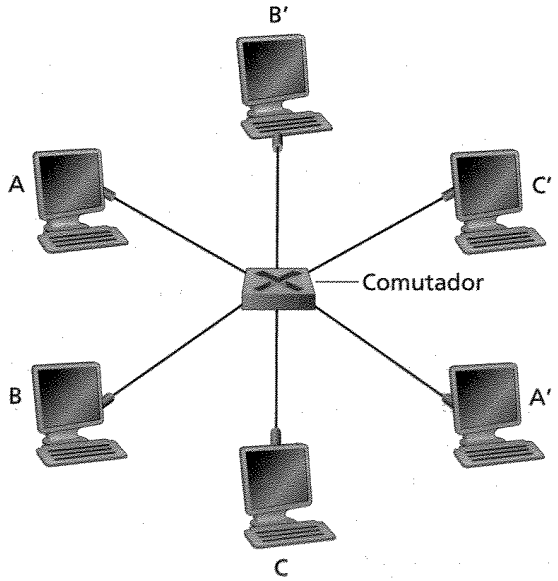


Figura 5.32 Um comutador que provê acesso dedicado Ethernet a seis hospedeiros

Para melhorar um pouco a percepção da diferença entre comutação de armazenagem e repasse e comutação acelerada, vamos recordar a analogia do comboio apresentada na Seção 1.6. Nessa analogia, existe uma rodovia com alguns postos de pedágio, e cada posto de pedágio tem um único operador. Na rodovia há um comboio de dez carros viajando juntos, todos com a mesma velocidade constante. Os carros do comboio são os únicos na estrada. Cada posto de pedágio atende aos carros com velocidade constante, de modo que, quando os carros saem do posto de pedágio, o espaço entre eles continua o mesmo. Como antes, podemos imaginar o comboio como se fosse um pacote, cada carro do comboio como se fosse um bit e a velocidade de atendimento do posto de pedágio como se fosse a taxa de transmissão de um enlace. Considere agora o que os carros do comboio fazem quando chegam a uma cabine de pedágio. Se cada carro se dirige diretamente à cabine ao chegar, então esse pedágio é um pedágio 'acelerado'. Se, por outro lado, cada carro espera na entrada até que os outros carros do comboio cheguem, então o pedágio é de armazenagem e repasse. O pedágio de armazenagem e repasse claramente atrasa o comboio mais do que o pedágio acelerado.

Um comutador acelerado pode reduzir o atraso fim-a-fim do pacote. Mas em quanto? Como mencionamos antes, o máximo atraso de armazenagem e repasse é L/R, onde L é o comprimento do pacote e R é a taxa de transmissão no enlace de entrada. O atraso máximo é aproximadamente 1,2 milissegundo para Ethernet de 10 Mbps e de 0,12 milissegundo para Ethernet de 100 Mbps (correspondente ao tamanho máximo de um pacote Ethernet). Assim, um comutador acelerado reduz o atraso em apenas 0,12 a 1,2 milissegundo, e essa redução ocorre apenas quando o enlace de saída está levemente carregado. Quão significativo é esse atraso? É provável que não seja muito para a maioria das aplicações práticas; portanto, é bom pensar bem antes de vender sua casa para investir na característica de comutação acelerada.

Comutadores versus roteadores

Como aprendemos no Capítulo 4, roteadores são comutadores de pacotes do tipo armazena-e-repassa que transmitem pacotes usando endereços de camada de rede. Embora um comutador também seja um comutador de pacotes do tipo armazena-e-repassa, ele é fundamentalmente diferente de um roteador, pois repassa pacotes usando endereços MAC. Enquanto um roteador é um comutador de pacotes de camada 3, um comutador opera com protocolos de camada 2.

Mesmo sendo fundamentalmente diferentes, é comum que os administradores de rede tenham de optar entre um comutador e um roteador ao instalar um dispositivo de interconexão. Por exemplo, para a rede da Figura 5.28, o administrador de rede poderia facilmente ter optado por usar um roteador, em vez de um comutador. Na verdade, um roteador também teria mantido separados os três domínios de colisão, permitindo, ao mesmo tempo, a comunicação interdepartamental. Dado que ambos, comutadores e roteadores, são candidatos a dispositivos de interconexão, quais são os prós e os contras das duas abordagens?

Vamos considerar, inicialmente, os prós e os contras de comutadores. Como mencionamos antes, comutadores são do tipo plug-and-play, uma propriedade que é apreciada por todos os administradores de rede atarefados do mundo. Comutadores também podem ter velocidades relativamente altas de filtragem e repasse — como mostra a Figura 5.33, eles têm de processar quadros apenas até a camada 2, enquanto roteadores têm de processar pacotes até a camada 3. Por outro lado, a topologia de uma rede de comutação está restrita a uma *spanning tree*. E mais, uma rede de comutação de grande porte exigiria, nos nós, tabelas ARP também de grande porte e geraria tráfego e processamento ARP substanciais. Além do mais, comutadores não oferecem nenhuma proteção contra tempestades de *broadcast* — se um hospedeiro se desorganiza e transmite uma corrente sem fim de quadros Ethernet *broadcast*, os comutadores repassam todos esses quadros, causando o colapso da rede inteira.

Agora, vamos considerar os prós e os contras dos roteadores. Como na rede o endereçamento muitas vezes é hierárquico (e não linear, como o endereçamento MAC), os pacotes normalmente não ficam circulando nos roteadores, mesmo quando a rede tem trajetos redundantes. (Na verdade, eles podem circular quando as tabelas de roteadores estão mal configuradas; mas, como aprendemos no Capítulo 4, o IP usa um campo de cabeçalho de datagrama especial para limitar a circulação.) Assim, pacotes não ficam restritos a uma topologia de *spanning tree* e podem usar o melhor trajeto entre fonte e destino. Como roteadores não

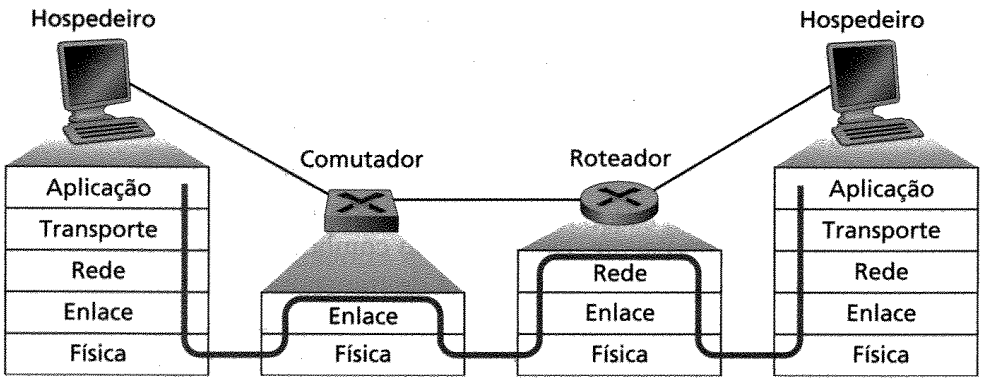


Figura 5.33 Processamento de pacotes em comutadores, roteadores e hospedeiros

sofrem a limitação da topologia *spanning tree*, em árvore, eles permitiram que a Internet fosse montada com uma topologia rica que inclui, por exemplo, múltiplos enlaces ativos entre a Europa e a América do Norte. Outra característica dos roteadores é que eles fornecem proteção de firewall contra as tempestades de *broadcast* de camada 2. Talvez a desvantagem mais significativa dos roteadores seja o fato de que eles não são do tipo plug-and-play — eles e os hospedeiros que a eles se conectam precisam de seus endereços IP para ser configurados. Além disso, roteadores muitas vezes apresentam tempo de processamento por pacote maior do que comutadores porque têm de processar até os campos da camada 3. Finalmente, há dois modos diferentes de pronunciar a palavra 'router': 'rúter' ou 'ráuter', e as pessoas perdem muito tempo discutindo qual é a melhor pronúncia [Perlman, 1999].

Dado que comutadores e roteadores têm seus prós e contras, quando uma rede institucional (por exemplo, uma rede de um campus universitário ou uma rede empresarial) deveria usar comutadores e quando deveria usar roteadores? Em geral, redes pequenas, com algumas centenas de hospedeiros, têm uns poucos segmentos de LAN. Para essas redes pequenas comutadores serão satisfatórios, pois localizam o tráfego e aumentam a vazão agregada sem exigir nenhuma configuração de endereços IP. Mas redes maiores, com milhares de hospedeiros, tipicamente incluem roteadores na rede (além de comutadores). Roteadores fornecem isolamento de tráfego mais robusto, controlam tempestades de *broadcast* e usam rotas 'mais inteligentes' entre os hospedeiros da rede.

Aprendemos nesta seção que hubs, comutadores e roteadores podem ser usados como dispositivos de interconexão para hospedeiros e segmentos de LAN. A Tabela 5.1 apresenta um resumo das características de cada um desses dispositivos de interconexão. O site Web da Cisco fornece numerosas comparações entre as diferentes tecnologias de interconexão [Cisco LAN Switches, 2004].

5.7 PPP: o protocolo ponto-a-ponto

Grande parte de nossa discussão sobre protocolos de camada de enlace esteve voltada, até aqui, aos protocolos para canais *broadcast*. Nesta seção, examinaremos um protocolo de camada de dados para enlaces ponto-a-ponto: o PPP. Como o PPP é o protocolo comumente escolhido para o enlace discado de hospedeiros residenciais, ele é, sem dúvida, um dos protocolos de camada de enlace mais utilizados hoje em dia. O outro importante protocolo de camada de dados em uso atualmente é o protocolo HDLC (*high-level data link control* — controle de enlace de dados de alto nível); consulte [Spragins, 1991] para detalhes sobre o HDLC. Nossa discussão do protocolo PPP, um protocolo mais simples, nos permitirá explorar muitas das mais importantes características de um protocolo ponto-a-ponto de camada de enlace.

Como o nome indica, o PPP [RFC 1661; RFC 2153] é um protocolo de camada de enlace que opera sobre um **enlace ponto-a-ponto** — um enlace que conecta diretamente dois nós, um em cada extremidade do enlace. O enlace ponto-a-ponto sobre o qual o PPP opera pode ser uma linha telefônica discada serial (por exemplo, uma conexão de modem de 56K), um enlace SONET/SDH, uma conexão X.25 ou um circuito ISDN

(*integrated service digital network* — rede digital de serviço integrado). Como observado antes, o PPP se tornou o protocolo preferido para conectar usuários residenciais a seus ISPs por meio de uma conexão discada.

Antes de apresentarmos os detalhes do PPP, é instrutivo examinar as exigências originais da IETF para o projeto do PPP [RFC 1547]:

- *Enquadramento do pacote.* No protocolo PPP de camada de enlace, o remetente deve ser capaz de pegar um pacote da camada de rede e encapsulá-lo dentro do quadro PPP da camada de enlace de tal modo que o receptor possa identificar o início e o fim do quadro de camada de enlace e também do pacote da camada de rede que está dentro do quadro.
- *Transparência.* O protocolo PPP não deve impor nenhuma restrição sobre os dados que aparecem no pacote da camada de rede (cabeçalhos ou dados). Assim, por exemplo, ele não pode proibir a utilização de certos padrões de bits no pacote da camada de rede. Voltaremos a esse assunto em breve, quando discutirmos enchimento de bytes.
- *Múltiplos protocolos de camada de rede.* O protocolo PPP deve estar habilitado a suportar múltiplos protocolos de camada de rede (por exemplo, IP e DECnet) que executam sobre o mesmo enlace físico e ao mesmo tempo. Exatamente como o protocolo IP tem de multiplexar diferentes protocolos de nível de transporte (por exemplo, TCP e UDP) sobre uma única conexão fim-a-fim, o PPP também deve estar habilitado a multiplexar diferentes protocolos de camada de rede sobre uma única conexão ponto-a-ponto. Essa exigência significa que, no mínimo, o PPP provavelmente necessitará de um campo de 'tipo de protocolo' ou de algum mecanismo similar para que seu lado receptor possa demultiplexar um quadro recebido até o protocolo de camada de rede apropriado.
- *Múltiplos tipos de enlaces.* Além de poder carregar múltiplos protocolos de nível mais alto, o PPP deve também poder operar sobre uma grande variedade de tipos de enlaces, incluindo enlaces que são seriais (transmitem um bit por vez em uma dada direção), paralelos (transmitem bits em paralelo), síncronos (transmitem um sinal de relógio juntamente com os dados de bits) ou assíncronos, de baixa velocidade ou de alta velocidade, elétricos ou óticos.
- *Detecção de erros.* Um receptor PPP deve estar habilitado a detectar erros de bits no quadro recebido.
- *Vida da conexão.* O PPP deve estar habilitado a detectar uma falha no nível de enlace (por exemplo, a incapacidade de transferir dados do lado remetente do enlace para o lado receptor) e sinalizar essa condição de erro à camada de rede.
- *Negociação do endereço de camada de rede.* O PPP deve fornecer um mecanismo para que as camadas de rede comunicantes (por exemplo, IP) aprendam ou configurem mutuamente seus endereços de camada de rede.
- *Simplicidade.* Além de atender às características relacionadas, o PPP tem de atender a uma série de requisitos adicionais. E, como se tudo isso não bastasse, a exigência mais primordial e mais importante é a simplicidade. O RFC 1547 estabelece que "a marca de um protocolo ponto-a-ponto deve ser a simplicidade". Uma ordem realmente difícil de ser cumprida, dadas todas as outras exigências feitas quanto ao projeto do PPP! Mais de 50 RFCs definem atualmente os vários aspectos desse protocolo 'simples'.

	Hubs	Roteadores	Comutadores
Isolamento de tráfego	não	sim	sim
Plug-and-play	sim	não	sim
Roteamento ótimo	não	sim	não
Comutação acelerada	sim	não	sim

Tabela 5.1 Comparação entre as características típicas de dispositivos de interconexão populares

Embora possa parecer que tenham sido muitas as exigências impostas ao projeto do PPP, na verdade a situação poderia ter sido muito pior! As especificações de projeto do PPP também estabeleceram explicitamente as funcionalidades de protocolo que o PPP *não* é obrigado a implementar.

- *Correção de erros.* Exige-se que o PPP detecte erros de bits, mas não se exige que ele os corrija.
- *Controle de fluxo.* Espera-se que um receptor PPP possa receber quadros à velocidade total da camada física subjacente. Se uma camada superior não puder receber pacotes a essa velocidade total, então será problema da camada superior descartar os pacotes ou regular o remetente na camada superior. Isto é, em vez de fazer com que o remetente PPP regule sua própria velocidade de transmissão, é responsabilidade de um protocolo de camada superior regular a velocidade na qual os pacotes são entregues ao PPP para envio.
- *Seqüenciamento.* Não se exige que o PPP entregue quadros ao enlace receptor na mesma ordem em que foram enviados pelo enlace remetente. É interessante notar que, embora essa flexibilidade seja compatível com o modelo de serviço IP (que permite que pacotes IP sejam entregues fim-a-fim em qualquer ordem), outros protocolos de camada de rede que operam sobre PPP exigem entrega de pacotes fim-a-fim seqüenciada.
- *Enlaces multiponto.* O PPP precisa operar apenas sobre enlaces que têm um único remetente e um único receptor. Outros protocolos de camada de enlace (por exemplo, o HDLC) podem atender a múltiplos receptores (por exemplo, em um cenário semelhante ao da Ethernet) em um enlace.

Agora que consideramos as metas de projeto (e as 'não-metas') para o PPP, vamos ver como o projeto do PPP atinge essas metas.

5.7.1 Enquadramento de dados PPP

A Figura 5.34 mostra um quadro de dados PPP que utiliza um enquadramento semelhante ao do HDLC [RFC 1662]. O quadro PPP contém os seguintes campos:

- *Campo de flag.* Todo quadro PPP começa e termina com um campo de flag de um byte de valor 01111110.
- *Campo de endereço.* O único valor possível para esse campo é 11111111.
- *Campo de controle.* O único valor possível para esse campo é 00000011. Como os campos de endereço e de controle somente podem assumir um valor fixo, é de se perguntar, antes de mais nada, por que foram definidos. A especificação PPP [RFC 1662] estabelece que outros valores "poderão ser definidos mais tarde", embora nenhum tenha sido definido até agora. Como os campos assumem valores fixos, o PPP permite que o remetente simplesmente não envie os bytes de endereço e de controle, economizando, dessa maneira, 2 bytes de sobrecarga no quadro PPP.
- *Protocolo.* O campo de protocolo diz ao receptor PPP a qual protocolo de camada superior pertencem os dados encapsulados recebidos (isto é, o conteúdo do campo de informação do quadro PPP). Ao receber um quadro PPP, o receptor PPP verifica se o quadro está correto e, então, repassa os dados encapsulados ao protocolo apropriado. O RFC 1700 e o RFC 3232 definem os códigos de protocolo de 16 bits usados pelo PPP. São de nosso interesse o protocolo IP (isto é, os dados encapsulados no quadro PPP em um datagrama IP), que tem um valor hexadecimal de 21, outros protocolos de camada de rede como o AppleTalk (29) e o DECnet (27), o protocolo de controle de enlace PPP (C021 hexadecimal), que discutiremos detalhadamente na seção seguinte, e o protocolo de controle IP (8021). Esse último protocolo é chamado pelo PPP quando um enlace é ativado pela primeira vez para configurar a conexão IP entre os dispositivos que utilizam o IP em cada extremidade do enlace (veja a seguir).

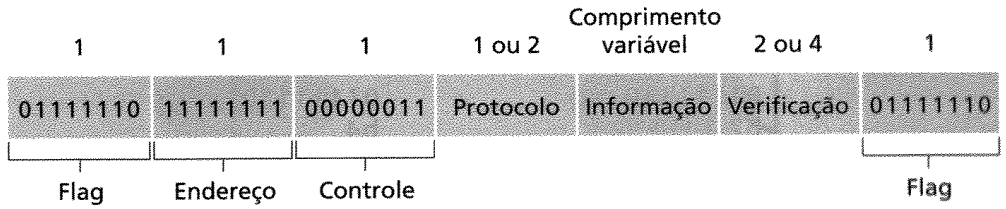


Figura 5.34 Formato do quadro de dados PPP

Informação. Esse campo contém o pacote encapsulado (dados) que está sendo enviado por um protocolo de camada superior (por exemplo, IP) pelo enlace PPP. O comprimento máximo padrão do campo de informação é 1.500 bytes, embora isso possa ser mudado quando o enlace é configurado pela primeira vez, como discutido a seguir.

Soma de verificação. O campo de soma de verificação é usado para detectar erros de bits em um pacote transmitido. Ele usa um código de redundância cíclica padrão HDLC* de 2 ou 4 bytes.

Byte Stuffing

Antes de encerrarmos nossa discussão sobre o enquadramento PPP, vamos considerar um problema que surge quando qualquer protocolo usa um padrão específico de bits em um campo de flag para indicar o começo ou o fim do quadro. O que acontece se o padrão do flag ocorre em outro local do pacote? Por exemplo, o que acontece se o valor do campo de flag 01111110 aparece no campo de informação? O receptor vai detectar incorretamente o final do quadro PPP?

Uma das maneiras de resolver esse problema seria o PPP proibir que o protocolo de camada superior enviasse dados com o padrão de bits do campo de flag. A exigência de transparência do PPP, discutida anteriormente, impede essa possibilidade. Uma solução alternativa, que é usada pelo PPP e por muitos outros protocolos, é a técnica conhecida como **byte stuffing**.

O PPP define um byte de controle especial de escape, 01111101. Se a seqüência do flag 01111110 aparecer em outro lugar do quadro, exceto no campo do flag, o PPP precederá aquele exemplar do padrão de flag com o byte de controle de escape. Em outras palavras, ele 'recheia' (adiciona) com um byte de controle de escape a seqüência de dados transmitida, antes de 01111110, para mostrar que o 01111110 seguinte não é um valor de flag, mas sim dados. Um receptor que vir um 01111110 precedido de um 01111101 removerá, é claro, o byte de controle de escape introduzido para reconstruir os dados originais. De maneira semelhante, se o próprio padrão do byte de controle de escape aparecer como dados, ele também deverá ser precedido por um byte de controle de escape introduzido. Assim, quando o receptor vir um único byte de controle de escape isolado na corrente de dados, ele saberá que o byte foi introduzido na corrente de dados. Um par de bytes de controle de escape que ocorrem um atrás do outro significa que um exemplar do byte de controle de escape aparece nos dados originais que estão sendo enviados. A Figura 5.35 ilustra o byte stuffing PPP. (Na realidade, o PPP também realiza uma operação *ou exclusivo* entre o byte de dados afetado pelo caractere de escape e o valor 20 hexadecimal, um detalhe que omitimos aqui para simplificar.)

* O HDLC utiliza um código de redundância cíclica de 16 bits. (N. do R.T.)

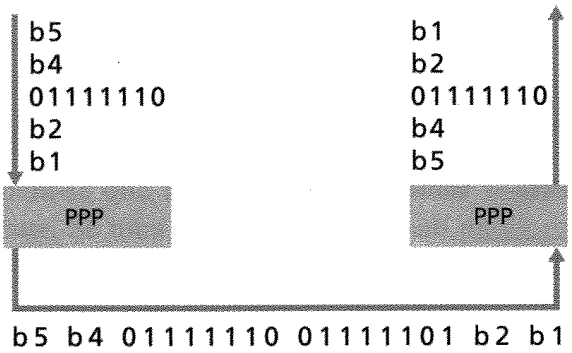


Figura 5.35 Byte stuffing

5.7.2 Protocolo de controle de enlace (LCP) e protocolos de controle de rede do PPP (NCP)

Até aqui, vimos como o PPP enquadra os dados que estão sendo enviados pelo enlace ponto-a-ponto. Mas como o enlace é configurado quando um hospedeiro ou um roteador em uma das extremidades do enlace PPP é ligado pela primeira vez? A abertura, a manutenção, a indicação de erro e o fechamento de um enlace PPP são realizados usando o **protocolo de controle de enlace** (*link-control protocol* — LCP) do PPP e a família de protocolos de controle de rede do PPP.

Antes que quaisquer dados sejam trocados sobre um enlace PPP, os dois pares (um em cada extremidade do enlace PPP) devem primeiramente rodar uma quantidade considerável de trabalho para configurar o enlace, algo muito parecido com a apresentação de três vias que um remetente e um destinatário TCP devem rodar (veja a Seção 3.5) para estabelecer os parâmetros da conexão TCP antes de transmitir dados e segmentos TCP. A Figura 5.36 ilustra o diagrama de estado de transição para o protocolo LCP para configurar, manter e encerrar o enlace PPP.

O enlace PPP sempre começa e termina em estado inativo. Quando um evento como uma detecção de portadora ou uma intervenção do administrador de rede indica que uma camada física está presente e pronta para ser usada, o PPP entra no estado de estabelecimento de enlace. Nesse estado, uma extremidade do enlace envia suas opções de configuração de enlace desejadas usando um quadro LCP `configure-request` de solicitação de configuração (um quadro PPP com o campo de protocolo com o valor do LCP e cujo campo de informação PPP contém a solicitação específica de configuração). O outro lado então responde com um quadro `configure-ack` (todas as opções aceitáveis), um quadro `configure-nak` (todas as configurações entendidas, mas não aceitáveis) ou um quadro `configure-reject` (opções não reconhecidas ou não aceitáveis para negociação). As opções de configuração LCP incluem um tamanho máximo de quadro para o enlace, a especificação de um protocolo de autenticação a ser usado (se houver) e uma opção para omitir a utilização dos campos de endereço e de controle dos quadros PPP.

Logo que esteja estabelecido o enlace, negociadas as opções de enlace e realizada a autenticação (se houver), os dois lados do enlace PPP trocam pacotes específicos de controle de camada de rede para cada protocolo de rede. Se o IP estiver rodando sobre o enlace PPP, o protocolo de controle IP [RFC 1332] será usado para configurar os módulos do protocolo IP em cada extremidade do enlace PPP. Dados IPCP são carregados dentro de um quadro PPP (com um campo de valor de protocolo igual a 8021), exatamente como dados LCP são carregados em um quadro PPP. O IPCP permite que os dois módulos IP troquem ou configurem seus endereços IP e negociem se seus datagramas serão ou não enviados em modo comprimido. Protocolos de controle de rede semelhantes são definidos para outros protocolos de camada de rede, como o DECnet [RFC 1762] e o AppleTalk [RFC 1378]. Logo que a camada de rede esteja configurada, o PPP pode começar a enviar datagramas de camada de rede — o enlace está no estado aberto e os dados começaram a fluir pelo enlace PPP. O quadro de solicitação de eco `(echo-request)` e o quadro de resposta de eco `(echo-reply)` LCP podem ser trocados entre as duas extremidades do PPP para verificar o estado do enlace.

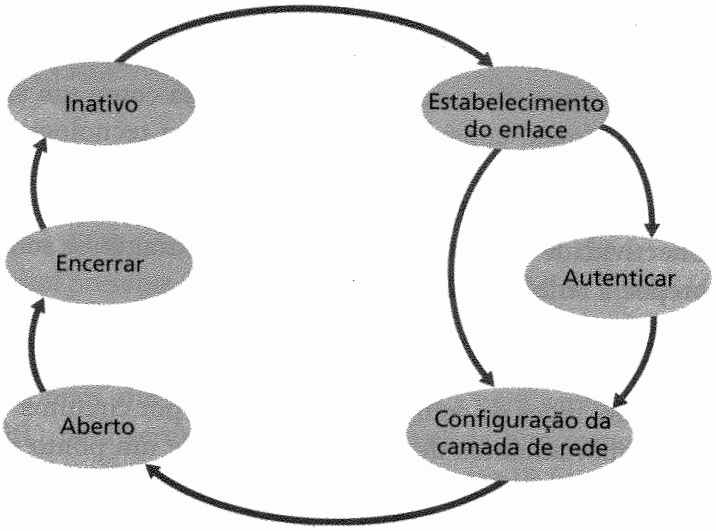


Figura 5.36 Estados do protocolo de controle de enlace PPP

O enlace PPP permanece configurado para comunicação até que um pacote de solicitação de encerramento (`terminate-request`) seja enviado. Se um quadro LCP de solicitação de encerramento (`terminate-request`) for enviado por uma extremidade do enlace PPP e respondido com um quadro LCP de reconhecimento de encerramento (`terminate-ack`), o enlace então entrará no estado inativo.

Em resumo, o PPP é um protocolo de camada de enlace de dados pelo qual dois pares comunicantes de nível de enlace, um em cada extremidade do enlace ponto-a-ponto, trocam quadros PPP contendo datagramas de camada de rede. Os principais componentes do PPP são:

- *Enquadramento.* Um método para encapsular dados em um quadro PPP, identificar o início e o final do quadro e detectar erros no quadro.
- Protocolo de controle de enlace. Um protocolo para inicializar, manter e encerrar o enlace PPP.
- *Protocolos de controle de rede.* Uma família de protocolos, um para cada protocolo de camada de rede superior, que permite que os módulos de camada de rede se configurem antes que os datagramas de camada de rede comecem a fluir pelo enlace PPP.

5.8 Virtualização de enlace: uma rede como camada de enlace

Como este capítulo trata de protocolos de camada de enlace, e dado que estamos chegando ao fim do capítulo, vamos refletir um pouco sobre como evoluiu o que entendemos como *enlace*. Começamos este capítulo considerando um enlace como um fio físico conectando dois hospedeiros comunicantes, como ilustrado na Figura 5.2. Quando estudamos protocolos de acesso múltiplo (Figura 5.9), vimos que vários hospedeiros podiam ser conectados por um fio compartilhado e que o 'fio' que conectava os hospedeiros podia ser o espectro de rádio ou qualquer outro meio. Isso nos levou a considerar o enlace, de um modo um pouco mais abstrato, como um canal, em vez de um fio. Quando estudamos LANs Ethernet (Figuras 5.26-5.28) vimos que, na verdade, os meios de interconexão poderiam ser uma infra-estrutura de comutação bastante complexa. Durante toda essa evolução, entretanto, os hospedeiros sempre mantiveram a visão do meio de conexão simplesmente como um canal de camada de enlace conectando dois ou mais hospedeiros. Vimos, por exemplo, que um hospedeiro Ethernet pode ficar alegremente inconsciente do fato de estar ligado a outros hospedeiros de LAN por um único segmento curto de LAN (Figura 5.9) ou por uma LAN comutada geograficamente dispersa (Figura 5.28).

Na Seção 5.7 vimos que o protocolo PPP é freqüentemente usado sobre uma conexão de modem entre dois hospedeiros. Nesse caso, o enlace que conecta os dois hospedeiros é, na verdade, a rede de telefonia —

uma rede global de telecomunicações logicamente separada, com seus próprios comutadores, enlaces e pilhas de protocolos para transferência e sinalização de dados. Entretanto, da perspectiva da camada de rede da Internet, a conexão discada por meio da rede de telefonia é vista como um simples 'fio'. Nesse sentido, a Internet virtualiza a rede de telefonia, considerando-a como uma tecnologia de camada de enlace que provê conectividade de camada de enlace entre dois hospedeiros da Internet. Lembre-se de que, quando discutimos redes de sobreposição no Capítulo 2, dissemos que, de modo semelhante, essa rede vê a Internet como um meio de prover conectividade entre nós sobrepostos, procurando se sobrepor à Internet do mesmo modo que a Internet se sobrepõe à rede de telefonia.

Nesta seção consideraremos redes ATM (modo de transferência assíncrono) e redes MPLS (*Multiprotocol Label Switching*). Diferentemente da rede de telefonia de comutação de circuitos, ambas as redes, ATM e MPLS, são, de direito, redes de comutação de pacotes por circuitos virtuais. Elas têm seus próprios formatos de pacotes e comportamentos de repasse. Assim, de um ponto de vista pedagógico, é bem coerente discutir ATM e MPLS quando estudamos camada de rede ou camada de enlace. Todavia, da perspectiva da Internet, podemos considerar ATM e MPLS, assim como a rede de telefonia e a Ethernet comutada, como tecnologias de camada de enlace que servem para interconectar dispositivos IP. Assim, consideraremos ambas, MPLS e ATM, ao discutirmos a camada de enlace. Redes frame-relay também podem ser usadas para interconectar dispositivos IP, embora representem uma tecnologia ligeiramente mais antiga (mas ainda disponibilizada), que não será discutida aqui; se quiser saber mais detalhes consulte o livro, de fácil leitura, de [Goralski, 1999]. Nosso estudo de ATM e MPLS será necessariamente breve, pois livros inteiros podem ser escritos (e foram) sobre essas redes. Aqui, focalizaremos primordialmente como essas redes servem para interconectar dispositivos IP, embora as tecnologias subjacentes também sejam examinadas com um pouco mais de profundidade.

5.8.1 Redes ATM (Modo de Transferência Assíncrono)

Os padrões para as redes **ATM** (*asynchronous transfer mode*) começaram a ser desenvolvidos em meados da década de 1980, tendo como meta o projeto de uma tecnologia de rede simples que transportasse áudio e vídeo, bem como texto, e-mail e arquivos de imagem em tempo real. Dois grupos, o ATM Fórum [ATM Forum, 2004] e a União Internacional de Telecomunicações [International Telecommunications Union — ITU, 2004], envolveram-se no desenvolvimento de padrões ATM. Eles definiram um padrão fim-a-fim completo, abrangendo desde a especificação da interface de aplicação para ATM até o enquadramento de dados ATM no nível de bit por várias camadas físicas, de fibra, de cobre e de rádio. Na prática, a rede ATM tem sido utilizada primordialmente em redes de telefonia e redes IP, servindo, por exemplo, como tecnologia de camada de enlace para conectar roteadores, como já discutimos anteriormente.

Principais características do ATM

Como discutimos na Seção 4.1, o ATM suporta diversos modelos de serviço, incluindo serviço de taxa constante de bits, serviço de taxa variável de bits, serviço de taxa disponível de bits e serviço de taxa não especificada de bits. ATM é uma arquitetura de comutação de pacotes por circuitos virtuais (CV). Lembre-se de que estudamos CVs com razoável detalhamento na Seção 4.2.1. A arquitetura geral do ATM é organizada em três camadas, como mostra a Figura 5.37.

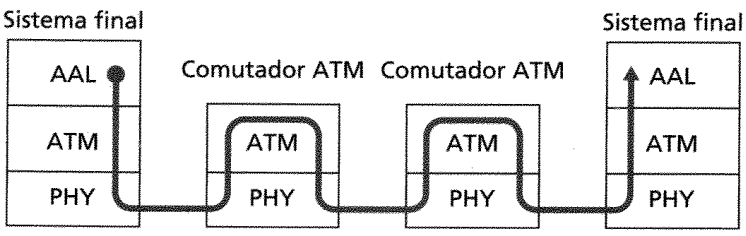


Figura 5.37 As três camadas ATM. A camada AAL está presente apenas nas bordas da rede ATM

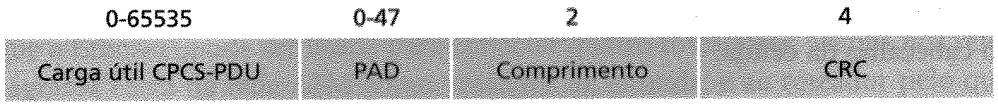


Figura 5.38 PDU-AAL5

A **camada de adaptação ATM** (*ATM adaptation layer* — AAL) é, em termos gerais, análoga à camada de transporte da Internet e está presente apenas nos dispositivos ATM na borda da rede ATM. No lado remetente, a AAL recebe dados passados por uma aplicação ou protocolo de nível mais alto (tal como IP, se a rede ATM estiver sendo utilizada para conectar dispositivos IP). No lado receptor, ela passa dados para o protocolo ou aplicação da camada mais alta. Foram definidas AALs para serviços de taxa constante de bits e de emulação de circuito (AAL1) para serviços de taxa variável de bits como vídeo de taxa variável de bits (AAL2) e para serviços de dados como transporte de datagramas IP (AAL5). Entre os serviços executados pela AAL estão detecção de erros e segmentação/remontagem. A unidade de dados manipulada pela AAL é conhecida pelo nome bastante genérico de **unidade de dados de protocolo AAL** (PDU AAL), que é aproximadamente equivalente a um segmento UDP ou TCP.

A PDU-AAL5 é mostrada na Figura 5.38. Os campos da PDU são relativamente diretos. O PAD garante que a PDU seja um número inteiro de 48 bytes, porque ela será segmentada para caber nas cargas úteis de 48 bytes dos pacotes ATM subjacentes (conhecidos como *células* ATM). O comprimento do campo identifica o tamanho da carga útil da PDU, de modo que o PAD possa ser removido no receptor. O campo CRC provê detecção de erro usando a mesma verificação de redundância cíclica da Ethernet. O campo de carga útil pode ter até 65.535 bytes de comprimento.

Agora vamos descer uma camada e considerar a **camada ATM**, que está no coração da arquitetura ATM. A camada ATM define a estrutura da célula ATM e o significado dos campos dentro da célula. A célula ATM é tão importante para uma rede ATM quanto o datagrama para uma rede IP. Os primeiros 5 bytes da célula constituem o cabeçalho ATM; os demais 48 bytes constituem a carga útil ATM. A Figura 5.39 mostra a estrutura do cabeçalho da célula ATM.

Os campos da célula ATM têm as seguintes funções:

- **Identificador do canal virtual (VCI).** Indica o canal virtual ao qual a célula pertence. Como acontece com a maioria das tecnologias de rede que usam circuitos virtuais, o VCI de uma célula é modificado de enlace a enlace (veja a Seção 4.2.1).
- **Tipo da carga útil (PT).** Indica o tipo de carga útil que a célula contém. Há diversos tipos de carga útil de dados, diversos tipos de carga útil de manutenção e um tipo de carga útil de célula inativa. O campo PT também inclui um bit que serve para indicar que a última célula é uma PDU-AAL fragmentada.
- **Bit de prioridade de perda de célula (CLP).** Pode ser estabelecido pela fonte para diferenciar o tráfego de alta prioridade do tráfego de baixa prioridade. Se ocorrer congestionamento e um comutador ATM tiver de descartar células, ele poderá usar esse bit para descartar primeiramente o tráfego de baixa prioridade.

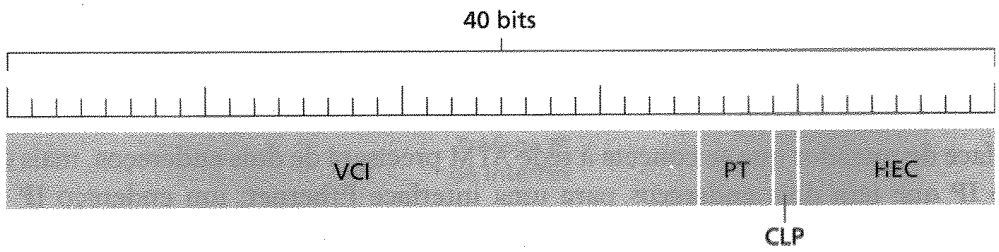


Figura 5.39 Formato do cabeçalho de célula ATM

- **Byte de controle de erro no cabeçalho (HEC).** Bits de detecção e correção de erros que protegem o cabeçalho da célula.

Antes que uma fonte possa começar a enviar células a um destino, a rede ATM deve criar um canal virtual desde a fonte até o destino. Um canal virtual nada mais é do que um circuito virtual, como descrito na Seção 4.2.1. Cada canal virtual é um caminho que consiste em uma seqüência de enlaces entre a fonte e o destino. Há um identificador de canal virtual (VCI) associado com cada enlace do canal virtual. Sempre que um canal virtual for estabelecido ou encerrado, as tabelas de tradução de canais virtuais devem ser atualizadas (veja a Seção 4.3.1). Se forem utilizados canais virtuais permanentes, não há necessidade de estabelecimento e encerramento dinâmicos de canais virtuais. Quando for preciso estabelecimento e encerramento dinâmicos de canais virtuais, o protocolo Q.2931 [Black, 1997, ITU-T Q.2931] fornece a sinalização necessária entre os comutadores ATM e sistemas finais.

A **camada física ATM** está no ponto mais baixo da pilha de protocolos ATM e trata de tensões, temporização de bits e enquadramento no meio físico. Uma boa parte da camada física depende das características físicas do enlace. Há duas classes gerais de camadas físicas: as que têm uma estrutura de transmissão de quadros (por exemplo, T1, T3, Sonet e SDH) e as que não têm. Se a camada física tiver uma estrutura de quadros, então ela será responsável pela geração e pelo delineamento de quadros. O termo 'quadros', usado nesta seção, não deve ser confundido com os quadros de camada de enlace (por ex., Ethernet), utilizados nas seções anteriores deste capítulo. O quadro de transmissão é um mecanismo da camada física semelhante ao mecanismo TDM, que serve para organizar os bits enviados por um enlace. Algumas camadas físicas possíveis incluem:

- Sonet/SDH (rede ótica síncrona/hierarquia digital síncrona) por fibra ótica monomodo. Como o T1 e o T3, a Sonet e a SDH têm estruturas de quadros que estabelecem sincronização de bits entre o transmissor e o receptor nas duas extremidades do enlace. Há várias velocidades padronizadas, incluindo:
  - OC-1: 51,84 Mbps
  - OC-3: 155,52 Mbps
  - OC-12: 622,08 Mbps
  - OC-48: 2,5 Gbps
- Quadros T1/T3 sobre fibra, microondas e cobre.
- Baseadas em células sem quadros. Nesse caso, o relógio no receptor é derivado de um sinal transmitido.

IP sobre ATM

Agora vamos considerar como uma rede ATM pode ser usada para prover conectividade entre dispositivos IP. A Figura 5.40 mostra um backbone ATM com quatro pontos de entrada/saída para tráfego IP de Internet. Observe que cada ponto de entrada/saída é um roteador. Um backbone ATM pode abranger um continente inteiro e ter dezenas ou até centenas de comutadores ATM. A maioria dos backbones ATM tem um canal virtual permanente entre cada par de pontos de entrada/saída. Usando canais virtuais permanentes, células ATM são roteadas do ponto de entrada ao ponto de saída sem ter de estabelecer e encerrar canais virtuais dinamicamente. Canais virtuais permanentes, no entanto, são viáveis apenas quando o número de pontos de entrada/saída é relativamente pequeno. Para n pontos de entrada, são necessários $n(n-1)$ canais virtuais permanentes para conectar diretamente os n pontos de entrada/saída.

Cada interface de roteador que se conecta à rede ATM precisará de dois endereços, mais ou menos como um hospedeiro IP que tem dois endereços para uma interface Ethernet: um endereço IP e um endereço MAC. De modo semelhante, uma interface ATM terá um endereço IP e um endereço ATM. Considere agora um datagrama IP que está atravessando a rede ATM mostrada na Figura 5.40. No caso mais simples, a rede

ATM aparece como um enlace lógico único — o ATM interconecta esses quatro roteadores exatamente do mesmo modo que a Ethernet pode ser usada para conectar quatro roteadores. Vamos nos referir ao roteador pelo qual o datagrama entra na rede ATM como 'roteador de entrada' e ao roteador pelo qual o datagrama sai da rede como 'roteador de saída'. O roteador de entrada faz o seguinte:

1. Examina o endereço de destino do datagrama.
2. Indexa sua tabela de roteamento e determina o endereço IP do roteador de saída (isto é, o próximo roteador na rota do datagrama).
3. Para levar o datagrama até o roteador de saída, o roteador de entrada vê a rede ATM simplesmente como um outro protocolo de camada de enlace. Para transportar o datagrama até o roteador seguinte, deve ser determinado o endereço físico do roteador do próximo salto. Lembre-se de que dissemos na Seção 5.4.2 que isso é feito usando ARP. No caso de uma interface ATM, o roteador de entrada indexa uma tabela ARP ATM com o endereço IP do roteador de saída e determina o endereço ATM desse roteador de saída. O protocolo ATMARP é descrito em [RFC 2225].
4. O IP do roteador de entrada então passa o datagrama, juntamente com o endereço ATM do roteador de saída, para baixo, para a camada de enlace (ou seja, ATM).

Após a conclusão dessas quatro etapas, a tarefa de levar o datagrama até o roteador de saída sai das mãos do IP e passa para as mãos do ATM. O ATM agora deve levar o datagrama até o endereço de destino ATM obtido na etapa 3. Essa tarefa tem duas subtarefas:

1. Determinar o VCI para o canal virtual que leva ao endereço de destino ATM.
2. Segmentar o datagrama em células no lado remetente do canal virtual (isto é, no roteador de entrada) e remontar as células do datagrama original no lado receptor do canal virtual (isto é, no roteador de saída).

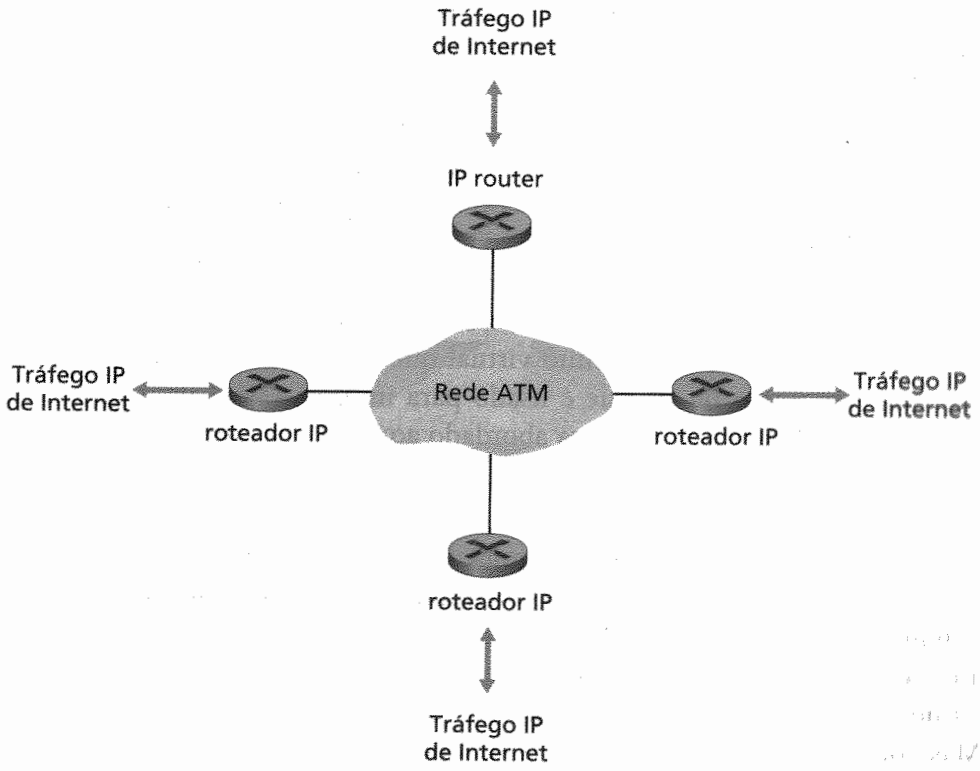


Figura 5.40 Rede ATM no núcleo de um backbone de Internet

A primeira subtarefa é direta. A interface do lado remetente mantém uma tabela que mapeia endereços ATM para os VCIs. Como estamos presumindo que os canais virtuais são permanentes, essa tabela está atualizada e é estática. (Se os canais virtuais não fossem permanentes, então seria preciso o protocolo de sinalização ATM Q.2931 para estabelecê-los e encerrá-los dinamicamente.) A segunda subtarefa merece consideração mais cuidadosa. Uma abordagem é usar fragmentação IP, como discutido na Seção 4.4. Com fragmentação IP, o roteador remetente primeiramente fragmentaria o datagrama original — cada fragmento não seria maior do que 48 bytes para poder caber na carga útil da célula ATM. Mas essa abordagem de fragmentação apresenta um grande problema — cada fragmento IP tem, caracteristicamente, 20 bytes de cabeçalho, de modo que uma célula ATM que carregasse um fragmento teria 25 bytes de cabeçalho e apenas 28 bytes de informação útil. O ATM usa a AAL5 para prover um meio mais eficiente de segmentar e remontar um datagrama.

A rede ATM, então, leva cada célula através da rede até o endereço de destino ATM. Em cada comutador ATM entre a fonte ATM e o destino ATM, a célula ATM é processada pela camada física ATM e pela camada ATM, mas não pela camada AAL. Em cada comutador, o VCI normalmente é traduzido (veja a seção 4.2.1) e o HEC é recalculado. Quando as células chegam ao endereço de destino ATM, elas são direcionadas ao buffer da AAL que foi alocado para esse canal virtual específico. A PDU-AAL5 é então reconstruída e o datagrama IP é extraído e passado, pilha de protocolos acima, para a camada IP.

5.8.2 Comutação de Rótulos Multiprotocolo (MPLS)

A Comutação de Rótulos Multiprotocolo (MPLS) evoluiu dos inúmeros esforços realizados pela indústria de meados ao final da década de 1990 para melhorar a velocidade de repasse de roteadores IP adotando um conceito fundamental do mundo das redes de circuitos virtuais: um rótulo de tamanho fixo. O objetivo não era abandonar a infra-estrutura de repasse de datagramas IP com base no destino em favor de rótulos de tamanho fixo e circuitos virtuais, mas aumentá-la rotulando datagramas seletivamente e permitindo que roteadores repassassem datagramas com base em rótulos de tamanho fixo (em vez de endereços de destino IP), quando possível. O importante é que essas técnicas trabalhavam de mãos dadas com o IP, usando endereçamento e roteamento IP. A IETF reuniu esses esforços no protocolo MPLS [RFC 3031, RFC 3032] misturando efetivamente técnicas de circuitos virtuais em uma rede de datagramas com roteadores.

Vamos começar nosso estudo do MPLS considerando o formato de um quadro de camada de enlace que é manipulado por um roteador habilitado para MPLS. A Figura 5.41 mostra que um quadro de camada de enlace transmitido por um enlace PPP ou LAN (tal como a Ethernet) tem um pequeno cabeçalho MPLS adicionado entre o cabeçalho de camada 2 (isto é, PPP ou Ethernet) e o cabeçalho de camada 3 (isto é, IP). O RFC 3032 define o formato do cabeçalho MPLS para esses enlaces; cabeçalhos de redes ATM e de frame relay também são definidos em outros RFCs. Entre os campos no cabeçalho MPLS estão o rótulo (que desempenha o papel de identificador de circuito virtual que estudamos na Seção 4.2.1), 3 bits reservados para uso experimental, um único bit, S, que é usado para indicar o final de uma série de rótulos MPLS 'empilhados' (um tópico avançado que não será abordado aqui), e um campo de tempo de vida.

A Figura 5.41 deixa imediatamente evidente que um quadro melhorado com MPLS somente pode ser enviado entre roteadores habilitados para MPLS (já que um roteador não habilitado para MPLS ficaria bastante confuso ao encontrar um cabeçalho MPLS onde esperava encontrar o cabeçalho IP!). Um roteador habilitado para MPLS é comumente denominado **roteador de comutação de rótulos**, pois repassa um quadro MPLS consultando o rótulo MPLS em sua tabela de repasse e, então, passando imediatamente o datagrama para a interface de saída apropriada. Assim, o roteador habilitado para MPLS *não* precisa extrair o endereço de destino e executar uma busca para fazer a compatibilização com o prefixo mais longo na tabela de repasse. Mas como um roteador sabe se seu vizinho é realmente habilitado para MPLS e como sabe qual rótulo associar com o dado destino IP? Para responder a essas perguntas, precisaremos estudar a interação entre um grupo de roteadores habilitados para MPLS.

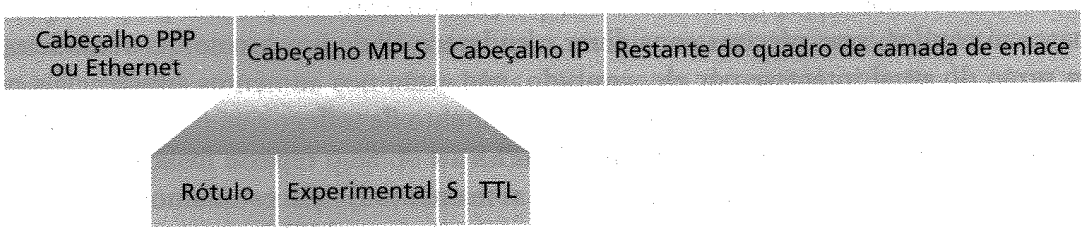


Figura 5.41 Cabeçalho MPLS: localizado entre os cabeçalhos de camada de enlace e camada de rede

No exemplo da Figura 5.42, os roteadores R1 a R4 são habilitados para MPLS. R5 e R6 são roteadores IP padrão. R1 anunciou a R2 e R3 que ele (R1) pode rotear para o destino A, e que um quadro recebido com rótulo MPLS 6 será repassado ao destino A. O Roteador R3 anunciou ao Roteador R4 que ele (R4) pode rotear para os destinos A e D e que os quadros que estão chegando e que portam os rótulos MPLS 10 e 12, respectivamente, serão comutados na direção desses destinos. O Roteador R2 também anunciou ao Roteador R4 que ele (R2) pode alcançar o destino A e que um quadro recebido portando o rótulo MPLS 8 será comutado na direção de A. Note que o Roteador R4 agora está na interessante posição de ter *dois* caminhos MPLS para chegar até A: via a interface 0 com rótulo de saída MPLS 10 e via a interface 1 com um rótulo MPLS 8. O quadro geral apresentado na Figura 5.42 mostra que o modo como R4, R5, A e D estão conectados em conjunto via uma infra-estrutura MPLS (roteadores habilitados a MPLS R1, R2, R3 e R4) é praticamente o mesmo modo pelo qual uma LAN comutada ou uma rede ATM podem conectar dispositivos IP entre si. E, do mesmo modo que uma LAN comutada ou uma rede ATM, os roteadores R1 a R4 habilitados para MPLS *fazem isso sem jamais tocar o cabeçalho IP de um pacote.*

Nessa discussão, não especificamos o protocolo específico utilizado para distribuir rótulos entre roteadores habilitados para MPLS, pois os detalhes dessa sinalização estariam muito além do escopo deste livro. Observamos, entretanto, que o grupo de trabalho da IETF para o MPLS especificou no RFC 3468 que uma extensão do protocolo RSVP (que estudaremos no Capítulo 7), conhecido como RSVP-TE [RFC 3209], será o foco de seus esforços para a sinalização MPLS. Assim, aconselhamos o leitor interessado a consultar o RFC 3209.

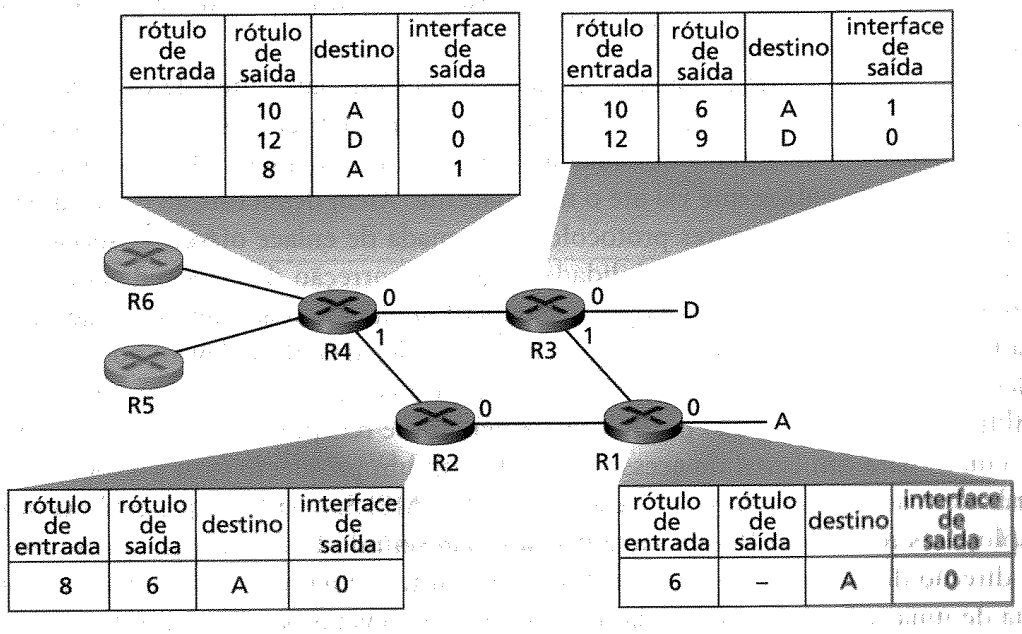


Figura 5.42 Repasse melhorado com MPLS

Até aqui, a ênfase de nossa discussão sobre o MPLS tem sido o fato de que esse protocolo executa comutação com base em rótulos, sem precisar considerar o endereço IP de um pacote. As verdadeiras vantagens do MPLS e a razão do atual interesse por ele, contudo, não estão nos aumentos substanciais nas velocidades de comutação, mas nas novas capacidades de gerenciamento de tráfego que o MPLS proporciona. Como observamos anteriormente, o R4 tem *dois* caminhos MPLS até A. Se o repasse fosse executado até a camada IP tendo como base o endereço IP, os protocolos de roteamento IP que estudamos no Capítulo 4 especificariam somente um único caminho de menor custo até A. Assim, o MPLS provê a capacidade de repassar pacotes por rotas que não seria possível utilizar com protocolos padronizados de roteamento IP. Essa é somente uma forma simples de **engenharia de tráfego** usando o MPLS [RFC 3346; Xiao, 2000], com a qual um operador de rede pode suplantar o roteamento IP normal e obrigar que uma parte do tráfego dirigido a um dado destino siga por um caminho, e que outra parcela do tráfego dirigido ao mesmo destino siga por um outro caminho (seja por política, por desempenho ou por alguma outra razão).

Também é possível utilizar MPLS para muitas outras finalidades. O protocolo pode ser usado para realizar restauração rápida de caminhos de repasse MPLS, por exemplo, mudar a rota do tráfego que passa sobre um caminho previamente calculado, em resposta à falha de enlace [Kar, 2000; Huang, 2002; RFC 3469]. O MPLS também pode ser utilizado para implementar a estrutura de serviço diferenciado ('diffserv'), que estudaremos no Capítulo 7. Finalmente, observamos que o MPLS pode ser, e tem sido, utilizado para implementar as denominadas **redes virtuais privadas** (*virtual private networks* — VPN). Ao implementar uma VPN para um cliente, um ISP utiliza uma rede habilitada para MPLS para conectar as várias redes do cliente. O MPLS também pode ser usado para isolar os recursos e o endereçamento utilizados pela VPN do cliente dos outros usuários que estão passando através da rede do ISP. Para mais detalhes, veja [DeClercq, 2002].

Nossa discussão sobre o MPLS foi necessariamente breve e aconselhamos o leitor a consultar as referências que mencionamos. Observamos que são tantas as utilizações possíveis do MPLS que ele parece estar se tornando rapidamente o canivetinho suíço da engenharia de tráfego da Internet!

5.9 Resumo

Neste capítulo, examinamos a camada de enlace — seus serviços, os princípios subjacentes à sua operação e vários protocolos específicos importantes que usam esses princípios na implementação dos serviços da camada de enlace.

Vimos que o serviço básico da camada de enlace é mover um datagrama de camada de rede de um nó (roteador ou hospedeiro) até um nó adjacente. Vimos também que todos os protocolos de camada de enlace operam encapsulando um datagrama de camada de rede dentro de um quadro de camada de enlace antes de transmitir o quadro pelo enlace que vai até o nó adjacente. Além dessa função comum de enquadramento, contudo, aprendemos que diferentes protocolos de camada de enlace oferecem serviços muito diferentes de acesso ao enlace, de entrega (confiabilidade, detecção/correção de erros), de controle de fluxo e de transmissão (por exemplo, serviços *full-duplex* versus *half-duplex*). Essas diferenças são devidas, em parte, à vasta variedade de tipos de enlaces sobre os quais os protocolos de enlace devem operar. Um enlace ponto-a-ponto simples tem um único remetente e um único receptor comunicando-se por um único 'fio'. Um enlace de acesso múltiplo é compartilhado por muitos remetentes e receptores; conseqüentemente, o protocolo de camada de enlace para um canal de acesso múltiplo tem um protocolo (seu protocolo de acesso múltiplo) para coordenar o acesso ao enlace. Nos casos de ATM e MPLS, o 'enlace' que conecta dois nós adjacentes (por exemplo, dois roteadores IP que são adjacentes em sentido IP — ou seja, são roteadores IP do próximo salto na direção do destino) pode, na realidade, constituir uma rede em si e por si próprio. Em certo sentido, a idéia de uma rede ser considerada um 'enlace' não deveria parecer estranha. Um enlace telefônico que conecta um modem/computador residencial a um modem/roteador remoto, por exemplo, na verdade é um caminho que atravessa uma sofisticada e complexa rede de telefonia.

Dentre os princípios subjacentes à comunicação por camada de enlace, examinamos técnicas de detecção e correção de erros, protocolos de acesso múltiplo, endereçamento de camada de enlace e a construção de redes locais ampliadas por hubs e comutadores. No caso da detecção/correção de erros, examinamos como é possível adicionar bits ao cabeçalho de um quadro para detectar e algumas vezes corrigir erros de mudança de bits que podem ocorrer quando o quadro é transmitido pelo enlace. Analisamos esquemas simples de paridade e de soma de verificação, bem como o esquema mais robusto de verificação da redundância cíclica. Passamos, então, para o tópico dos protocolos de acesso múltiplo. Identificamos e estudamos três abordagens amplas para coordenar o acesso a um canal *broadcast*: abordagens de divisão de canal (TDM, FDM, CDMA), abordagens de acesso aleatório (os protocolos ALOHA e os protocolos CSMA) e abordagens de revezamento (polling e passagem de permissão). Vimos que uma conseqüência do compartilhamento de um canal *broadcast* por múltiplos nós é a necessidade de prover endereços aos nós no nível da camada de enlace. Aprendemos que endereços físicos são bastante diferentes de endereços de camada de rede e que, no caso da Internet, um protocolo especial (o ARP — protocolo de resolução de endereço) é usado para fazer o mapeamento entre esses dois modos de endereçamento. Em seguida, examinamos como os nós que compartilham um canal *broadcast* formam uma LAN e como várias LANs podem ser conectadas para formar uma LAN de maior porte — tudo isso sem a intervenção de roteamento de camada de rede para a interconexão desses nós locais.

Também estudamos detalhadamente vários protocolos específicos de camada de enlace — Ethernet e PPP. Encerramos nosso estudo da camada de enlace focalizando como redes ATM e MPLS fornecem serviços de camada de enlace quando interconectadas com roteadores IP. Com isso, concluímos nossa jornada de cima para baixo da pilha de protocolos. É claro que a camada física fica abaixo da camada de enlace, mas provavelmente será melhor deixar os detalhes da camada física para outro curso (por exemplo, teoria da comunicação, em vez de redes de computadores). Contudo, discutimos brevemente vários aspectos da camada física — neste capítulo, por exemplo, discutimos a codificação Manchester na Seção 5.5, e no Capítulo 1, meios físicos na Seção 1.4. Consideraremos novamente a camada física quando estudarmos as características de enlaces sem fio no próximo capítulo.

Embora nossa jornada de cima para baixo da pilha de protocolos esteja encerrada, nosso estudo sobre rede de computadores ainda não chegou ao fim. Nos quatro capítulos seguintes, examinaremos rede sem fio, rede multimídia, segurança da rede e gerenciamento de rede. Esses quatro tópicos não se ajustam convenientemente a uma única camada; na verdade, cada um deles atravessa muitas camadas. Assim, entender esses tópicos (às vezes tachados de 'tópicos avançados' em alguns textos sobre redes) requer uma boa base sobre todas as camadas da pilha de protocolos — uma base que se completou com nosso estudo sobre a camada de enlace de dados!

Exercícios de fixação

Capítulo 5 Questões de revisão

Seções 5.1 a 5.2

1. Se todos os enlaces da Internet fornecessem serviço confiável de entrega, o serviço confiável de entrega TCP seria redundante? Justifique sua resposta.

2. Cite alguns possíveis serviços que um protocolo de camada de enlace pode oferecer à camada de rede. Quais desses serviços de camada de enlace têm serviços correspondentes no IP? E no TCP?

Seção 5.3

3. Suponha que dois nós comecem a transmitir ao mesmo tempo um pacote de comprimento L por um canal *broadcast* de velocidade R. Denote o atraso de propagação entre os dois nós como t_{prop}. Haverá uma colisão se $t_{prop} < L/R$? Por quê?

4. Na Seção 5.3, relacionamos quatro características desejáveis de um canal *broadcast*. O slotted ALOHA tem quais dessas características? E o protocolo de passagem de permissão, tem quais dessas características?
5. Descreva os protocolos de polling e de passagem de permissão usando a analogia com as interações ocorridas em um coquetel.
6. Por que o protocolo de passagem de permissão seria ineficiente se uma LAN tivesse um perímetro muito grande?

Seção 5.4

7. Que tamanho tem o espaço de endereço MAC? E o espaço de endereço IPv4? E o espaço de endereço IPv6?
8. Suponha que cada um dos nós A, B e C esteja ligado à mesma LAN *broadcast* (por meio de seus adaptadores). Se A enviar milhares de datagramas IP a B com quadro de encapsulamento endereçado ao endereço MAC de B, o adaptador de C processará esses quadros? Se processar, ele passará os datagramas IP desses quadros para C (isto é, para o nó pai do adaptador)? O que mudaria em suas respostas se A enviasse quadros com o endereço MAC de *broadcast*?
9. Por que uma pesquisa ARP é enviada dentro de um quadro *broadcast*? Por que uma resposta ARP é enviada dentro de um quadro com um endereço MAC de destino específico?
10. Na rede da Figura 5.19, o roteador tem dois módulos ARP, cada um com sua própria tabela ARP. É possível que o mesmo endereço MAC apareça em ambas as tabelas?

Seção 5.5

11. Compare as estruturas de quadro das redes Ethernet 10BaseT, 100BaseT e Gigabit Ethernet. Quais as diferenças entre elas?
12. Suponha que um adaptador de 10 Mbps envie para dentro de um canal uma cadeia infinita de 1s usando a codificação Manchester. Quantas transições por segundo terá o sinal que emerge do adaptador?
13. Em CSMA/CD, após a quinta colisão, qual é a probabilidade de que um nó escolha $K = 4$? O resultado $K = 4$ corresponde a um atraso de quantos segundos em uma rede Ethernet de 10 Mbps?

Problemas

1. Suponha que o conteúdo de informação de um pacote seja o padrão de bits 1010101010101011 e que um esquema de paridade par esteja sendo usado. Qual seria o valor do campo de soma de verificação para o caso de um esquema de paridade bidimensional? Sua resposta deve ser tal que seja usado um campo de soma de verificação de comprimento mínimo.
2. Dê um exemplo (que não seja o da Figura 5.6) mostrando que verificações de paridade bidimensional podem corrigir e detectar um erro de bit único. Dê um outro exemplo mostrando um erro de bit duplo que pode ser detectado, mas não corrigido.
3. Suponha que a porção de informação de um pacote (D da Figura 5.4) contém 10 bytes consistindo na representação binária sem sinal dos números inteiros de 0 a 9. Calcule a soma de verificação da Internet para esses dados.
4. Considere o gerador de 4 bits G mostrado na Figura 5.8 e suponha que D tenha o valor de 10101010. Qual é o valor de R?
5. Na Seção 5.3, fornecemos um esboço da derivação da eficiência do slotted ALOHA. Neste problema, concluiremos a derivação.

a. Lembre-se de que, quando há N nós ativos, a eficiência do slotted ALOHA é $Np(1 - p)^{N-1}$. Ache o valor de p que maximize essa expressão.

b. Usando o valor de p encontrado em (a), ache a eficiência do slotted ALOHA fazendo com que N tenda ao infinito. *Dica*: $(1 - 1/N)^N$ tende a $1/e$ quando N tende ao infinito.

6. Mostre que a eficiência máxima do ALOHA puro é $1/(2e)$. *Obs.*: Este problema é fácil se você tiver concluído o anterior.

7. Plote em gráfico a eficiência do slotted ALOHA e do ALOHA puro como uma função de p, para $N = 100$.

8. Considere um canal *broadcast* com N nós e uma taxa de transmissão de R bps. Suponha que o canal *broadcast* use protocolo de polling (com um nó de polling adicional) para acesso múltiplo. Suponha que o intervalo de tempo entre o momento em que o nó conclui a transmissão e o momento em que o nó subseqüente é autorizado a transmitir (isto é, o atraso de polling) seja t_{poll}. Suponha ainda que, dentro de uma rodada de polling, um dado nó seja autorizado a transmitir, no máximo, Q bits. Qual é a vazão máxima do canal *broadcast*?

9. Considere três LANs interconectadas por dois roteadores, como mostra o diagrama a seguir:

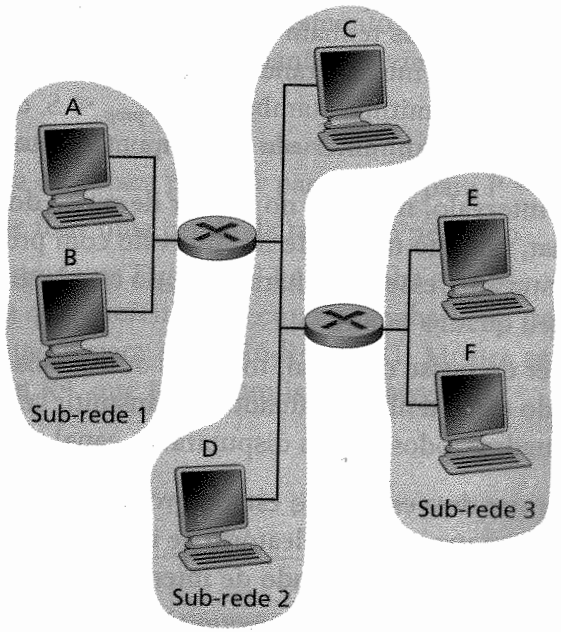
a. Faça um novo desenho do diagrama incluindo os adaptadores.

b. Atribua endereços a todas as interfaces. Para a Sub-rede 1, use endereços do tipo 111.111.111.xxx; para a Sub-rede 2, use endereços do tipo 122.222.222.xxx, e para a Sub-rede 3 use endereços do tipo 133.133.133.xxx.

c. Atribua endereços de MAC a todos os adaptadores.

d. Considere o envio de um datagrama IP do hospedeiro A ao hospedeiro F. Suponha que todas as tabelas ARP estejam atualizadas. Enumere todas as etapas como foi feito no exemplo de um único roteador na Seção 5.4.2.

e. Repita (d), admitindo agora que a tabela ARP do hospedeiro remetente esteja vazia (e que as outras tabelas estejam atualizadas).



10. Lembre-se de que, com o protocolo CSMA/CD, o adaptador espera $K \cdot 512$ tempos de bits após uma colisão, onde K é escolhido aleatoriamente. Para $K = 100$, quanto tempo o adaptador espera até voltar à etapa 2 para uma Ethernet de 10 Mbps? E para uma Ethernet de 100 Mbps?

11. Suponha que os nós A e B estejam no mesmo segmento Ethernet de 10 Mbps e que o atraso de propagação entre os dois nós seja de 225 tempos de bit. Suponha que o nó A comece a transmitir um quadro e que, antes de terminar, o nó B comece a transmitir um quadro. O nó A pode terminar de transmitir antes de detectar que B transmitiu? Por quê? Se a resposta for sim, então A acredita, incorretamente, que seu quadro foi transmitido com sucesso, sem nenhuma colisão.
Dica: suponha que no tempo $t = 0$ tempo de bit, A comece a transmitir um quadro. No pior dos casos, A transmite um quadro de tamanho mínimo de 512 + 64 tempos de bit. Portanto, A terminaria de transmitir o quadro em $t = 512 + 64$ tempos de bit. Então, a resposta será não, se o sinal de B chegar a A antes do tempo de bit $t = 512 + 64$ bits. No pior dos casos, quando o sinal de B chega a A?

12. Suponha que os nós A e B estejam no mesmo segmento de uma Ethernet de 10 Mbps e que o atraso de propagação entre os dois nós seja de 225 tempos de bit. Suponha também que A e B enviem quadros ao mesmo tempo, que os quadros colidam e que, então, A e B escolham valores diferentes de K no algoritmo CSMA/CD. Admitindo que nenhum outro nó esteja ativo, as retransmissões de A e B podem colidir? Para nossa finalidade, é suficiente resolver o seguinte exemplo. Suponha que A e B comecem a transmitir em $t = 0$ tempo de bit. Ambos detectam colisões em $t = 225$ tempos de bit. Eles terminam de transmitir um sinal de reforço de colisão em $t = 225 + 48 = 273$ tempos de bit. Suponha que $K_A = 0$ e $K_B = 1$. Em que tempo B programa sua retransmissão? Em que tempo A começa a transmissão? (*Nota*: os nós devem esperar por um canal ocioso após retornar à etapa 2 — veja o protocolo.) Em que tempo o sinal de A chega a B? B se abstém de transmitir em seu tempo programado?

13. Considere uma Ethernet 100BaseT de 100 Mbps. Para ter uma eficiência de 0,50, qual deve ser a distância máxima entre um nó e o hub? Admita um comprimento de quadro de 64 bytes e que não há repetidores. Essa distância máxima também garante que um nó transmissor A poderá detectar se outro nó transmitiu enquanto A estava transmitindo? Justifique sua resposta. Como se compara sua distância máxima com a estabelecida pelo próprio padrão de 100 Mbps?

14. Neste problema, você derivará a eficiência de um protocolo de acesso múltiplo semelhante ao CSMA/CD. Nesse protocolo, o tempo é segmentado e todos os adaptadores estão sincronizados com os intervalos. Diferentemente do slotted ALOHA, contudo, o comprimento de um intervalo (em segundos) é muito menor do que um tempo de quadro (o tempo para transmitir um quadro). Seja S o comprimento de um intervalo. Suponha que todos os quadros tenham comprimento constante $L = kRS$, onde R é a taxa de transmissão do canal e k é um número inteiro grande. Suponha que haja N nós, cada um com um número infinito de quadros para enviar. Admitimos também que $t_{prop} < S$, de modo que todos os nós podem detectar uma colisão antes do final de um intervalo de tempo. O protocolo é como segue:

- Se, para um dado intervalo, nenhum nó estiver de posse do canal, todos os nós disputam o canal; em particular, cada nó transmite no intervalo com probabilidade p. Se exatamente um nó transmitir no intervalo, esse nó tomará posse do canal para os $k - 1$ intervalos subseqüentes e transmitirá seu quadro inteiro.
- Se algum nó estiver de posse do canal, todos os outros nós evitarão transmitir até que o nó que está de posse do canal tenha terminado de transmitir seu quadro. Assim que esse nó tiver transmitido seu quadro, todos os nós disputarão o canal.

Note que o canal se alterna entre dois estados: o estado produtivo, que dura exatamente k intervalos, e o estado não produtivo, que dura um número aleatório de intervalos. A eficiência do canal é, claramente, a razão $k/(k + x)$, onde x é o número esperado de intervalos consecutivos não produtivos.

a. Para N e p fixos, determine a eficiência desse protocolo.

b. Para N fixo, determine p que maximiza a eficiência.

c. Usando o p (que é uma função de N) encontrado em (b), determine a eficiência quando N tende ao infinito.

d. Mostre que essa eficiência se aproxima de 1 quando o comprimento do quadro é grande.

15. Suponha que dois nós, A e B, estejam ligados às extremidades opostas de um cabo de 900 metros e que cada um tenha um quadro de 1.000 bits (incluindo todos os cabeçalhos e preâmbulos) para transmitir um ao outro. Ambos os nós tentam transmitir no tempo $t = 0$. Suponha que haja quatro repetidores entre A e B, cada um inserindo um atraso de 20 bits. Admita que a taxa de transmissão é 10 Mbps e que é usado um CSMA/CD com intervalos de backoff de múltiplos de 512 bits. Após a primeira colisão, A sorteia $K = 0$ e B sorteia $K = 1$ no protocolo de backoff exponencial. Ignore o sinal de reforço e o tempo de atraso de 96 bits.

a. Qual é o atraso de propagação em um sentido (incluindo os atrasos de repetidores) entre A e B em segundos? Suponha que a velocidade de propagação do sinal seja $2 \cdot 10^8$ m/s.

b. Em que tempo (em segundos) o pacote de A é completamente entregue em B?

c. Agora suponha que somente A tenha um pacote para enviar e que os repetidores sejam substituídos por comutadores. Suponha que cada comutador tenha um atraso de processamento de 20 bits além do atraso de armazenagem e repasse. Em que tempo, em segundos, o pacote de A é entregue em B?

16. Lembre-se de que o ATM usa pacotes de 53 bytes consistindo em 5 bytes de cabeçalho e 48 bytes de carga útil. Cinqüenta e três bytes é um tamanho excepcionalmente pequeno para pacotes de comprimento fixo; a maioria dos protocolos de rede (IP, Ethernet, Frame Relay e assim por diante) usa pacotes que são, em média, significativamente maiores. Uma das desvantagens de um pacote de tamanho pequeno é que uma grande fração de largura de banda de enlace é consumida por bytes de cabeçalho; no caso do ATM, quase 10 por cento da largura de banda é 'desperdiçada' pelo cabeçalho ATM. Neste problema, investigamos por que foi escolhido esse pacote tão pequeno. Com essa finalidade, suponha que a célula ATM consista em P bytes de dados (possivelmente diferente de 48) e 5 bytes de cabeçalho.

a. Considere o envio de uma fonte de voz codificada digitalmente diretamente sobre ATM. Suponha que a fonte esteja codificada a uma taxa constante de 64 Kbps. Admita que cada célula esteja integralmente cheia antes de a fonte enviá-la para dentro da rede. O tempo exigido para encher uma célula é o **atraso de empacotamento**. Determine, em termos de P, o atraso de empacotamento em milissegundos. Lembre que o tamanho da célula é igual a $P + 5$ bytes.

b. Os atrasos de empacotamento maiores do que 20 milissegundos podem causar ecos perceptíveis e desagradáveis. Determine o atraso de empacotamento para $P = 1.500$ bytes (correspondente, aproximadamente, a um pacote Ethernet de tamanho máximo) e para $P = 48$ bytes (correspondente a uma célula ATM).

c. Calcule o atraso de armazenagem e repasse em um único comutador ATM para uma taxa de enlace $R = 155$ Mbps (uma taxa de enlace popular para ATM) para $P = 1.500$ bytes e $P = 48$ bytes.

d. Comente as vantagens de usar uma célula de tamanho pequeno.

17. Considere a rede MPLS mostrada na Figura 5.42 e suponha que os roteadores R5 e R6 agora são habilitados para MPLS. Suponha que queremos executar engenharia de tráfego de modo que pacotes de R6 destinados a A sejam comutados para A via R6-R4-R2-R1. Mostre as tabelas MPLS em R5 e R6, bem como a tabela modificada em R4, que tornariam isso possível.

Questões dissertativas

Você deve navegar na Web para responder às perguntas a seguir.

1. Qual é a faixa de preço aproximada atual para um adaptador Ethernet de 10/100 Mbps? E para um adaptador Gigabit Ethernet? Compare esses preços com o preço de um modem discado de 56 kbps ou de um modem ASDL.

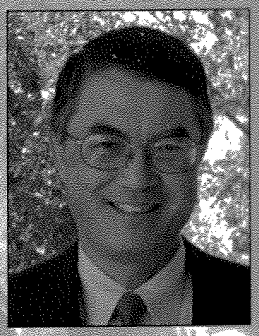
2. O preço de hubs e de comutadores geralmente é determinado segundo o número de interfaces (também denominadas *portas* no jargão da LAN). Qual é a faixa de preço aproximada atual por interface para um hub de 10 Mbps? Para um hub de 100 Mbps? Para um comutador composto apenas de interfaces de 10 Mbps? Para um comutador somente com interfaces de 100 Mbps?
3. Muitas das funções de um adaptador podem ser realizadas em software que executa no nó da CPU. Quais as vantagens e as desvantagens de deslocar essa funcionalidade do adaptador para o nó?
4. Utilize a Web para achar os números de protocolo usados em um quadro Ethernet para um datagrama IP e para um pacote ARP.
5. Leia as referências [Xiao, 2000; Huang 2002 e RFC 3346] sobre engenharia de tráfego utilizando MPLS. Relacione um conjunto de metas para a engenharia de tráfego. Quais dessas metas podem ser cumpridas somente com MPLS e quais delas são alcançadas utilizando protocolos existentes que não são MPLS? No último caso, quais são as vantagens que o MPLS oferece?

Ethereal Lab

No Companion Website, `www.aw.com/kurose_br` *você encontrará duas tarefas de Ethereal Lab, em inglês, para este capítulo. A primeira examina a operação do protocolo IEEE 802.3 e o formato do quadro Ethernet. A segunda explora a utilização do protocolo DHCP que estudamos na Seção 5.4.3*

Entrevista

Simon S. Lam



Simon S. Lam é professor e regente da cadeira de ciência da computação na Universidade do Texas, em Austin. De 1971 a 1974 trabalhou no ARPA Network Measurement Center na UCLA, com comutação por satélite e por rádio. Liderou um grupo de pesquisa que inventou os sockets seguros e construiu o primeiro protótipo da camada de sockets seguros (denominada Programação de Rede Segura) em 1993. Os pontos de interesse de sua pesquisa são o projeto e a análise de protocolos de rede e serviços de segurança. Graduou-se em engenharia elétrica na Washington State University e é mestre e doutor pela UCLA.

O que o fez se decidir pela especialização em redes?

Quando cheguei à UCLA, recém-formado, no outono de 1969, minha intenção era estudar teoria de controle. Então assisti às aulas de Leonard Kleinrock sobre teoria das filas e ele me impressionou muito. Durante um certo tempo trabalhei em controle adaptativo de sistemas de fila como possível tópico de tese. No início de 1972, Larry Roberts iniciou o projeto ARPAnet Satellite System (mais tarde denominado Packet Satellite). O professor Kleinrock perguntou se eu queria participar do projeto. A primeira coisa que fiz foi introduzir um algoritmo de backoff simples, mas realista, no protocolo da slotted ALOHA. Pouco tempo depois, encontrei muitos problemas interessantes para pesquisar, como o problema da instabilidade da ALOHA e a necessidade de backoff, que formariam o núcleo central da minha tese.

O senhor teve participação ativa nos primórdios da Internet na década de 1970, desde seus tempos de estudante na UCLA. Como era o panorama naquela época? As pessoas faziam alguma idéia do que a Internet se tornaria?

Na verdade, a atmosfera não era diferente da de outros projetos de construção de sistemas que já vi na indústria e no ambiente acadêmico. A meta inicial determinada para a ARPAnet era bastante modesta, isto é, prover acesso a computadores caros a partir de localizações remotas de modo que mais cientistas pudessem utilizá-los. Contudo, com o início do projeto Packet Satellite em 1972 e do projeto Packet Radio em 1973, a meta da ARPA foi ampliada substancialmente. Em 1973, a ARPA estava montando, ao mesmo tempo, três redes de pacotes diferentes e tornou-se necessário que Vint Cerf e Bob Kahn desenvolvessem uma estratégia de interconexão.

Naquela época, todos esses desenvolvimentos progressivos na área de redes eram vistos (acredito eu) mais como lógicos do que mágicos. Ninguém poderia ter previsto a escalada da Internet e o poder dos computadores de hoje. Transcorreu uma década antes do aparecimento dos primeiros PCs. Para colocar as coisas em perspecti-

va, a maioria dos estudantes apresentava seus programas de computador em cartões perfurados, para processamento em lote. Somente alguns deles tinham acesso direto a computadores, que normalmente eram abrigados em áreas restritas. Os modens eram lentos e ainda raros. No meu tempo de estudante, eu tinha apenas um telefone sobre minha mesa e usava papel e lápis para fazer a maior parte do meu trabalho.

Em sua opinião, qual é o futuro do campo das redes e da Internet?

No passado, a simplicidade do protocolo IP da Internet era sua maior força para vencer a concorrência e se tornar o padrão de *facto* do trabalho em redes. Diferentemente de seus concorrentes, como X-25 na década de 1980 e ATM na década de 1990, o IP pode rodar sobre qualquer tecnologia de camada de enlace porque oferece apenas um serviço datagrama de melhor esforço. Assim, qualquer rede de pacotes pode se conectar com a Internet.

Infelizmente, a maior força do IP agora é uma deficiência. O IP é como uma camisa-de-força que confina o desenvolvimento da Internet a direções específicas. A camada IP é muito importante economicamente para sofrer interferências de modo a poder suportar novas funcionalidades como *multicast* e QoS. Nos últimos anos, muitos pesquisadores redirecionaram seus esforços para as camadas de aplicação e transporte, para suporte de *multicast* e QoS. A maior parte dos outros tópicos interessantes de pesquisa da Internet, como segurança e sistemas P2P, envolve apenas a camada de aplicação. Há também uma grande concentração de pesquisas em redes sem fio e redes ad hoc, redes de sensores e redes por satélite. Essas redes podem ser consideradas ou como sistemas autônomos ou como sistemas de camada de enlace, que podem se desenvolver porque estão fora da camisa-de-força do IP.

Há muita gente animada com a possibilidade da utilização de sistemas P2P como plataforma para novas aplicações da Internet. Todavia, a utilização de recursos da Internet por sistemas P2P é altamente ineficiente. Uma das minhas preocupações é se a capacidade de transmissão e comutação do núcleo da Internet continuará a crescer mais rapidamente do que a demanda de tráfego na Internet, enquanto ela cresce para interconectar todos os tipos de equipamentos e suportar futuras aplicações habilitadas para P2P. Sem um superprovisionamento substancial de capacidade, garantir a estabilidade da rede na presença de congestão e de ataques mal-intencionados seria uma tarefa muito importante.

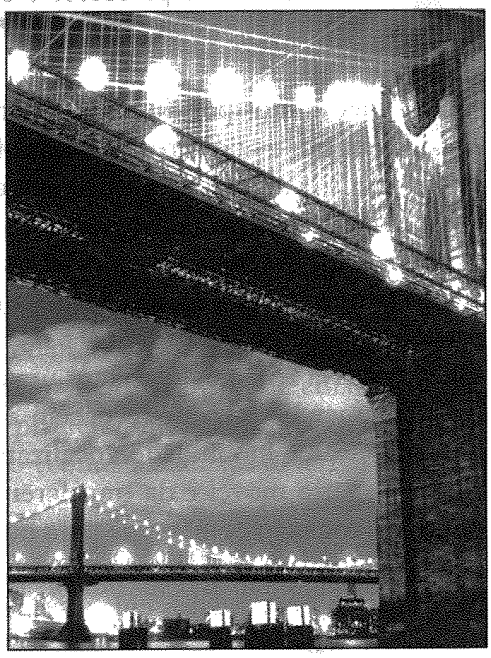
Qual parte de seu trabalho lhe apresenta mais desafios?

A parte do meu trabalho que me apresenta mais desafios é ensinar e motivar *todos* os estudantes que assistem às minhas aulas e *todos* os estudantes de doutorado que supervisiono, e não apenas os superbem-dotados. Os muito inteligentes e motivados podem exigir um pouco de supervisão, mas não muito mais do que isso. Muitas vezes aprendo mais com esses estudantes do que eles aprendem comigo. Ensinar e motivar os que não se destacam muito é um grande desafio.

Que impactos sobre o aprendizado o senhor acha que a tecnologia terá no futuro?

Eventualmente, quase todo o conhecimento humano estará acessível pela Internet, algo que será a mais poderosa ferramenta de aprendizado. Essa vasta base de conhecimento terá o potencial de nivelar o terreno para estudantes em todo o mundo. Por exemplo, estudantes motivados de qualquer país poderão acessar os melhores sites Web de aulas, conferências multimídia e material de ensino. Já foi dito que as bibliotecas digitais do IEEE e da ACM aceleraram o desenvolvimento de pesquisadores da ciência da computação na China. Com o tempo, a Internet transcenderá todas as barreiras geográficas do aprendizado.

Capítulo 6



Redes sem Fio e Redes Móveis

No mundo da telefonia pode-se dizer que os dez últimos anos foram a década da telefonia celular. O número de assinantes de telefones móveis no mundo inteiro aumentou de 34 milhões em 1993 para mais de um bilhão em 2003 e, agora, o número de assinantes da telefonia celular ultrapassa o número de linhas telefônicas convencionais [ITU Statistics, 2004]. As muitas vantagens dos telefones celulares são evidentes para todos — em qualquer lugar, a qualquer hora, acesso desimpedido à rede global de telefonia por meio de um equipamento leve e totalmente portátil. Com o advento de laptops, palmtops, PDAs e a promessa de acesso desimpedido à Internet global de qualquer lugar, a qualquer hora, será que estamos prestes a assistir a uma explosão semelhante da utilização de dispositivos sem fio para acesso à Internet?

Independentemente do crescimento futuro de equipamentos sem fio para Internet, já ficou claro que redes sem fio e os serviços móveis relacionados que elas possibilitam vieram para ficar. Do ponto de vista de rede, os desafios propostos por essas redes, particularmente nas camadas de enlace e de rede, são tão diferentes dos desafios das redes de computadores cabeadas que é necessário um capítulo individual (*este* capítulo) devotado ao estudo de redes sem fio e redes móveis.

Iniciaremos este capítulo com uma discussão sobre usuários móveis, enlaces e redes sem fio e sua relação com as redes maiores (normalmente cabeadas) às quais se conectam. Traçaremos uma distinção entre os desafios propostos pela natureza *sem fio* dos enlaces de comunicação nessas redes e pela *mobilidade* que esses enlaces sem fio habilitam. Fazer essa importante distinção — entre sem fio e mobilidade — nos permitirá isolar, identificar e dominar melhor os conceitos fundamentais em cada área. Note que, na realidade, há muitos ambientes de rede nos quais os nós da rede são sem fio, mas não são móveis (por exemplo, redes residenciais sem fio ou redes de escritórios compostas por estações de trabalho estacionárias e monitores de grandes dimensões), e formas limitadas de mobilidade que não requerem enlaces sem fio (por exemplo, um profissional que utiliza um laptop em casa, desliga o equipamento e o leva para seu escritório, onde o liga à rede cabeada da empresa em que trabalha). É claro que muitos dos ambientes sem fio mais interessantes são aqueles em que os usuários são sem fio *e também* móveis — por exemplo, um cenário no qual um usuário móvel (digamos, no banco traseiro de um carro) mantém uma chamada de voz sobre IP e várias conexões TCP ativas enquanto corre pela rodovia a 160 quilômetros por hora. É nesse ponto, em que o sem fio se cruza com a mobilidade, que encontraremos os desafios técnicos mais interessantes!

Começaremos por ilustrar, em primeiro lugar, o cenário no qual consideraremos comunicação e mobilidade sem fio — uma rede na qual usuários sem fio (e possivelmente móveis) estão conectados à infra-estrutura da rede maior por um enlace sem fio na borda de rede. Então, consideraremos as características desse enlace sem fio na Seção 6.2. Incluímos uma breve introdução ao Acesso Múltiplo por Divisão de Código (*Code Division Multiple Access* — CDMA), um protocolo de acesso ao meio compartilhado que é utilizado freqüentemente em redes sem fio. Na Seção 6.3 estudaremos com certa profundidade os aspectos da camada de enlace do padrão da LAN sem fio IEEE 802.11 (Wi-Fi); também falaremos um pouco sobre Bluetooth. Na Seção 6.4 daremos uma visão geral do acesso celular à Internet, incluindo as tecnologias celulares emergentes 3G, que fornecem acesso à Internet por voz e a alta velocidade. Na Seção 6.5, voltaremos nossa atenção à mobilidade, focalizando os problemas da localização de um usuário móvel, do roteamento para o usuário móvel e da transferência (handing off) do usuário móvel que passa dinamicamente de um ponto de conexão com a rede para outro. Estudaremos como esses serviços de mobilidade são implementados no padrão IP móvel e em GSM nas Seções 6.6 e 6.7, respectivamente. Finalmente, na Seção 6.8 consideraremos o impacto dos enlaces e da mobilidade sem fio sobre protocolos de camada de transporte e aplicações em rede.

6.1 Introdução

A Figura 6.1 mostra o cenário no qual consideraremos os tópicos de comunicação de dados e mobilidade sem fio. Começaremos mantendo nossa discussão dentro de um contexto geral o suficiente para abranger uma ampla faixa de redes, entre elas LANs sem fio como a IEEE 802.11 e redes celulares como uma rede 3G; passaremos para uma discussão mais detalhada de arquiteturas sem fio específicas em seções posteriores. Podemos identificar os seguintes elementos em uma rede sem fio:

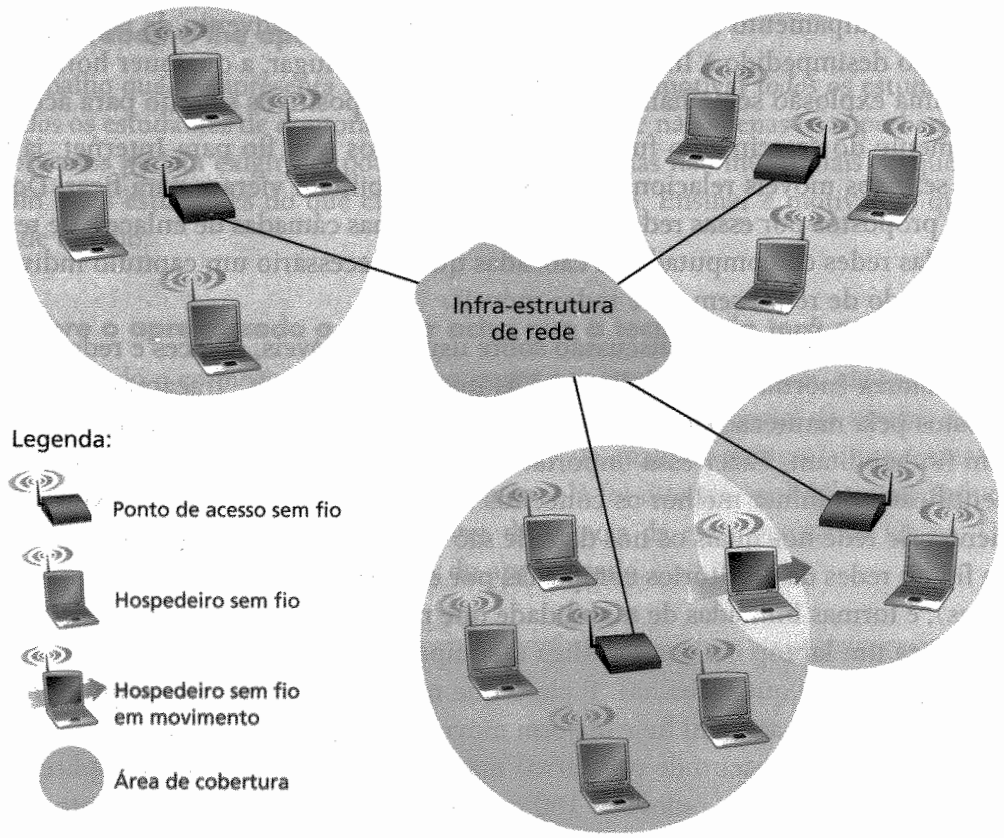


Figura 6.1 Elementos de uma rede sem fio

- *Hospedeiros sem fio.* Como no caso de redes cabeadas (ou com fio), hospedeiros são os equipamentos de sistemas finais que executam aplicações. Um **hospedeiro sem fio** pode ser um laptop, um palmtop, um PDA, um telefone ou um computador de mesa. Os próprios hospedeiros podem ser ou não ser móveis.
- *Enlaces sem fio.* Um hospedeiro se conecta a uma estação-base (definida mais adiante) ou a um outro hospedeiro sem fio por meio de um **enlace de comunicação sem fio**. Tecnologias de enlace sem fio diferentes têm taxas de transmissão diferentes e podem transmitir a distâncias diferentes. A Figura 6.2 mostra algumas das características fundamentais dos padrões de enlace sem fio mais populares. Abordaremos esses padrões mais adiante na primeira metade deste capítulo; consideraremos também outras características de enlaces sem fio (como suas taxas de erros de bits e as causas desses erros) na Seção 6.2.
- Na Figura 6.1 enlaces sem fio conectam hospedeiros localizados na borda da rede com a infra-estrutura da rede de maior porte. Não podemos nos esquecer de acrescentar que enlaces sem fio às vezes também são utilizados *dentro* de uma rede e para conectar roteadores, comutadores e outros equipamentos de rede. Contudo, neste capítulo, focalizaremos a utilização de comunicação sem fio nas bordas da rede, pois é aqui que estão ocorrendo muitos dos desafios técnicos mais interessantes e a maior parte do crescimento.
- *Estação-base.* A **estação-base** é uma parte fundamental da infra-estrutura de rede sem fio. Diferentemente dos hospedeiros e enlaces sem fio, uma estação-base não tem nenhuma contraparte óbvia em uma rede cabeada. Uma estação-base é responsável pelo envio e recebimento de dados (por exemplo, pacotes) de e para um hospedeiro sem fio que está associado com ela. Uma estação-base freqüentemente será responsável pela coordenação da transmissão de vários hospedeiros sem fio com os quais está associada. Quando dizemos que um hospedeiro sem fio está 'associado' com uma estação-base, isso quer dizer que (1) o hospedeiro está dentro do alcance de comunicação sem fio da estação-base e (2) o hospedeiro usa a estação-base para retransmitir dados entre ele (o hospedeiro) e a rede maior. **Torres celulares** em redes celulares e **pontos de acesso** em uma LAN sem fio 802.11 são exemplos de estações-base.

Na Figura 6.1, a estação-base está conectada à rede maior (isto é, à Internet, à rede empresarial ou residencial ou à rede telefônica) e, portanto, funciona como uma retransmissora de camada de enlace entre o hospedeiro sem fio e o resto do mundo com o qual o hospedeiro se comunica.

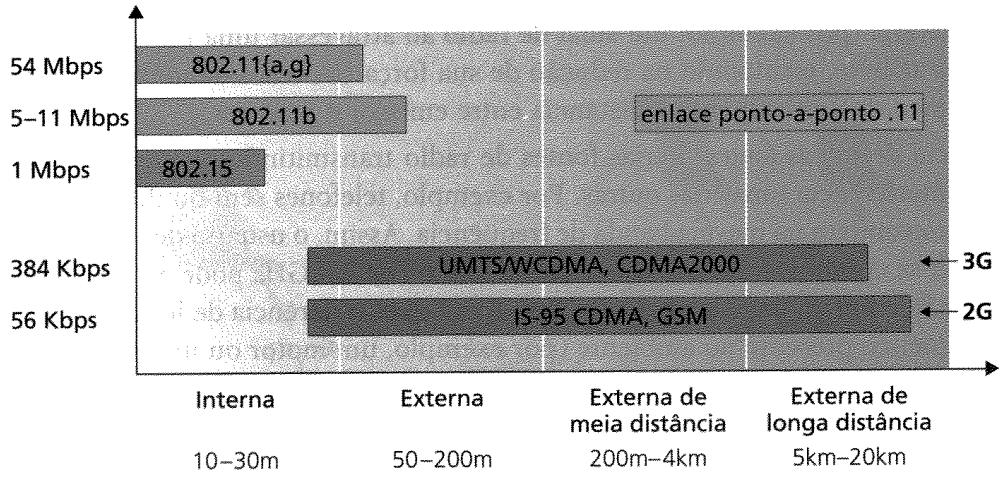


Figura 6.2 Características de enlaces de padrões selecionados de rede sem fio

Quando hospedeiros estão associados com uma estação-base, em geral diz-se que estão operando em **modo de infra-estrutura**, já que todos os serviços tradicionais de rede (por exemplo, atribuição de endereço e roteamento) são fornecidos pela rede com a qual estiverem conectados por meio da estação-base. Em **redes ad hoc**, hospedeiros sem fio não dispõem de nenhuma infra-estrutura desse tipo com a qual se conectar. Na ausência de tal infra-estrutura, os próprios hospedeiros devem prover serviços como roteamento, atribuição de endereço, tradução de endereços semelhante ao DNS e outros. Neste livro, concentraremos nossa atenção primordialmente em redes de modo de infra-estrutura.

Quando um hospedeiro móvel se desloca para fora da faixa de alcance de uma estação-base e entra na faixa de uma outra, ele muda seu ponto de conexão com a rede maior (isto é, muda a estação-base com a qual está associado) — um processo denominado **transferência** (handoff). Essa mobilidade dá origem a muitas questões desafiadoras. Se um hospedeiro pode se mover, como descobrir sua localização corrente na rede de modo que seja possível lhe transmitir dados? Como é realizado o endereçamento, dado que um hospedeiro pode estar em uma entre muitas localizações possíveis? Se o hospedeiro se movimentar *durante* uma conexão TCP ou ligação telefônica, como os dados serão roteados de modo que a conexão continue sem interrupção? Essas e muitas (mas muitas!) outras questões fazem das redes sem fio e móveis uma área muito interessante da pesquisa de redes.

- *Infra-estrutura de rede.* É a rede maior com a qual um hospedeiro sem fio pode querer se comunicar.

Agora vamos nos aprofundar um pouco mais nos desafios técnicos que surgem em redes sem fio e móveis. Começaremos considerando, em primeiro lugar, o enlace sem fio individual, adiando nossa discussão da mobilidade para outra parte deste capítulo.

6.2 Características de enlaces e redes sem fio

Vamos começar considerando uma rede simples cabeada, por exemplo, uma rede residencial com hospedeiros interconectados por um comutador Ethernet cabeado (ver Seção 5.6). Se substituíssemos a Ethernet cabeada por uma rede 802.11 sem fio, uma placa NIC sem fio substituiria as placas da Ethernet cabeada nos hospedeiros e um ponto de acesso substituiria o comutador Ethernet, mas, na camada de rede ou acima dela, praticamente nenhuma mudança seria necessária. Isso sugere que concentremos nossa atenção na camada de enlace ao procurarmos diferenças importantes entre redes com fio e sem fio. Realmente, podemos encontrar várias diferenças importantes entre um enlace com fio e um enlace sem fio:

- *Redução da força do sinal.* Radiações eletromagnéticas são atenuadas quando atravessam algum tipo de matéria (por exemplo, um sinal de rádio ao atravessar uma parede). O sinal se dispersará mesmo ao ar livre, resultando na redução de sua força (às vezes denominada **atenuação de percurso**) à medida que aumenta a distância entre emissor e receptor.
- *Interferência de outras fontes.* Várias fontes de rádio transmitindo na mesma banda de freqüência sofrerão interferência umas das outras. Por exemplo, telefones sem fio de 2,4 GHz e LANs sem fio 802.11b transmitem na mesma banda de freqüência. Assim, o usuário de uma LAN sem fio 802.11b que estiver se comunicando por um telefone sem fio de 2,4 GHz pode esperar que nem a rede nem o telefone funcionem particularmente bem. Além da interferência de fontes transmissoras, o ruído eletromagnético presente no ambiente (por exemplo, um motor ou um equipamento de microondas próximo) pode resultar em interferência.
- *Propagação multivias.* **Propagação multivias** (ou multicaminhos) ocorre quando porções da onda eletromagnética se refletem em objetos e no solo e tomam caminhos de comprimentos diferentes entre um emissor e um receptor. Isso resulta no embaralhamento do sinal recebido no destinatário. Objetos que se movimentam entre o emissor e o receptor podem fazer com que a propagação multivias mude ao longo do tempo.

História

Acesso público Wi-Fi: em breve facilmente a seu dispor?

Há apenas cinco anos, redes de computadores sem fio eram, de certa forma, uma curiosidade. Embora o licenciamento de espectro de rádio para sistemas 3G estivesse recebendo investimentos maciços (veja Histórico de Caso: Celular móvel 3G versus LANS sem fio), sistemas 3G estavam (e ainda estão) apenas nos estágios iniciais de disponibilização. Na época, alguns poucos usuários pioneiros estavam começando a experimentar a tecnologia de LAN sem fio IEEE 802.11, que acabara de ser padronizada. Mas que diferença cinco anos fazem! Hoje, muitas empresas, universidades e residências têm suas próprias LANs sem fio IEEE 802.11. Ainda mais notável é que o número de acessos públicos a redes sem fio — locais públicos onde usuários podem encontrar acesso sem fio 802.11 (hotspots) — está se expandindo rapidamente. O Gartner Group estima que havia 71 mil hotspots em 2003, um aumento de aproximadamente 50 vezes em relação a 2001. Nos Estados Unidos, cafés e lanchonetes como Starbucks e McDonalds oferecem acesso Wi-Fi em inúmeras localidades. Na cidade de Nova York, a Verizon Communications disponibilizou pontos de acesso Wi-Fi em mais de mil cabines de telefone público de sua propriedade e conectou essas cabines com a Internet (Verizon, 2004), oferecendo acesso Wi-Fi a transeuntes e estabelecimentos comerciais próximos. No início de 2004, a T-Mobile [T-Mobile, 2004] disponibilizou mais de quatro mil hotspots Wi-Fi públicos em lugares como aeroportos, restaurantes e livrarias. Em 2003, a Cometa, uma novata no setor, anunciou planos de estabelecer 20 mil hotspots Wi-Fi em 50 áreas metropolitanas até 2005. Com esse nível de atividade, o sonho do acesso à Internet global praticamente ubíquo, a qualquer hora e sem interferências pode estar mais perto de se concretizar do que pensamos!

A discussão anterior sugere que erros de bits serão mais comuns em enlaces sem fio do que em enlaces com fio. Por essa razão, talvez não seja nenhuma surpresa que protocolos de enlace sem fio (como o protocolo 802.11 que examinaremos na seção seguinte) empreguem não somente poderosos códigos de detecção de erros por CRC, mas também protocolos ARQ de nível de enlace que retransmitem quadros corrompidos.

Taxas de erros de bits mais altas e que variam ao longo do tempo não são as únicas diferenças entre um enlace com fio e um enlace sem fio. Lembre-se de que, no caso de enlaces *broadcast* cabeados, todos os nós recebem as transmissões de todos os outros nós. No caso de enlaces sem fio, a situação não é tão simples, como mostra a Figura 6.3.

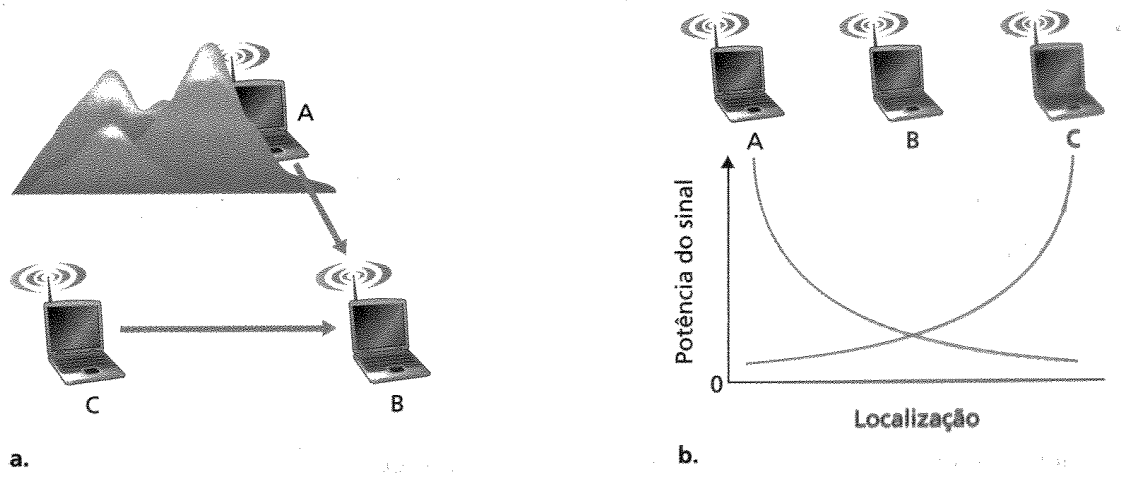


Figura 6.3 Problema do terminal oculto (a) e do desvanecimento (b)

Suponha que a estação A esteja transmitindo para a estação B. Suponha também que a estação C esteja transmitindo para a estação B. O denominado **problema do terminal oculto**, obstruções físicas presentes no ambiente (por exemplo, uma montanha ou um edifício), pode impedir que A e C escutem as transmissões de um e de outro, mesmo que as transmissões de A e C estejam interferindo no destino, B. Isso é mostrado na Figura 6.3(a). Um segundo cenário que resulta em colisões que não são detectadas no receptor é causado pelo **desvanecimento** da força de um sinal à medida que ele se propaga pelo meio sem fio. A Figura 6.3(b) ilustra o caso em que a localização de A e C é tal que as potências de seus sinais não são suficientes para que eles detectem as transmissões de um e de outro, mas, mesmo assim, *são* suficientemente fortes para interferir uma com a outra na estação B. Como veremos na Seção 6.3, o problema do terminal oculto e o desvanecimento tornam o acesso múltiplo em uma rede sem fio consideravelmente mais complexo do que em uma rede cabeada.

6.2.1 CDMA

Lembre-se de que dissemos, no Capítulo 5, que, quando hospedeiros se comunicam por um meio compartilhado, é preciso um protocolo para que os sinais enviados por vários emissores não interfiram nos receptores. No mesmo capítulo descrevemos três classes de protocolos de acesso ao meio: de partição de canal, de acesso aleatório e de revezamento. O acesso múltiplo por divisão de código (Code Division Multiple Access — CDMA) é ainda um quarto tipo de protocolo de acesso a um meio compartilhado, predominante em tecnologias celulares e de LAN sem fio. Por ser tão importante no mundo sem fio, examinaremos o CDMA rapidamente agora, antes de passar para tecnologias específicas de acesso sem fio nas seções subseqüentes.

Com um protocolo CDMA, cada bit que está sendo enviado é codificado pela multiplicação do bit por um sinal (o código) que muda a uma velocidade muito maior (conhecida como **velocidade de chipping**) do que a seqüência original de bits de dados. A Figura 6.4 mostra um cenário simples e idealizado de codificação/decodificação CDMA. Suponha que a velocidade com que bits de dados originais cheguem ao codificador CDMA defina a unidade de tempo, isto é, que cada bit original de dados a ser transmitido requeira um intervalo de tempo de um bit. Seja d_i o valor do bit de dados para o i-ésimo intervalo de bit. Por conveniência do cálculo matemático, representamos o bit de dados com valor 0 por -1. Cada intervalo de bit é ainda subdividido em M miniintervalos. Na Figura 6.4, $M = 8$, embora, na prática, M seja muito maior. O código CDMA usado pelo remetente consiste em uma seqüência de M valores, c_m, $m = 1, \ldots, M$, cada um assumindo um valor de $+1$ ou -1. No exemplo da Figura 6.4, o código CDMA de M bits que está sendo usado pelo remetente é $(1, 1, 1, -1, 1, -1, -1, -1)$.

Para ilustrar como o CDMA funciona, vamos focalizar o i-ésimo bit de dados, d_i. Para o m-ésimo mini-intervalo de tempo de transmissão de bits d_i, a saída do codificador CDMA, $Z_{i,m}$, é o valor de d_i multiplicado pelo m-ésimo bit do código CDMA escolhido, c_m:

$$Z_{i,m} = d_i \cdot c_m \tag{6.1}$$

Se o mundo fosse simples e não houvesse remetentes interferindo, o receptor receberia os bits codificados, $Z_{i,m}$, e recuperaria os bits de dados originais, d_i, calculando:

$$d_i = \frac{1}{M} \sum_{m=1}^{M} Z_{i,m} \cdot c_m \tag{6.2}$$

Talvez o leitor queira repassar os detalhes do exemplo da Figura 6.4 para verificar se os bits originais de dados são, de fato, corretamente recuperados no receptor usando a Equação 6.2.

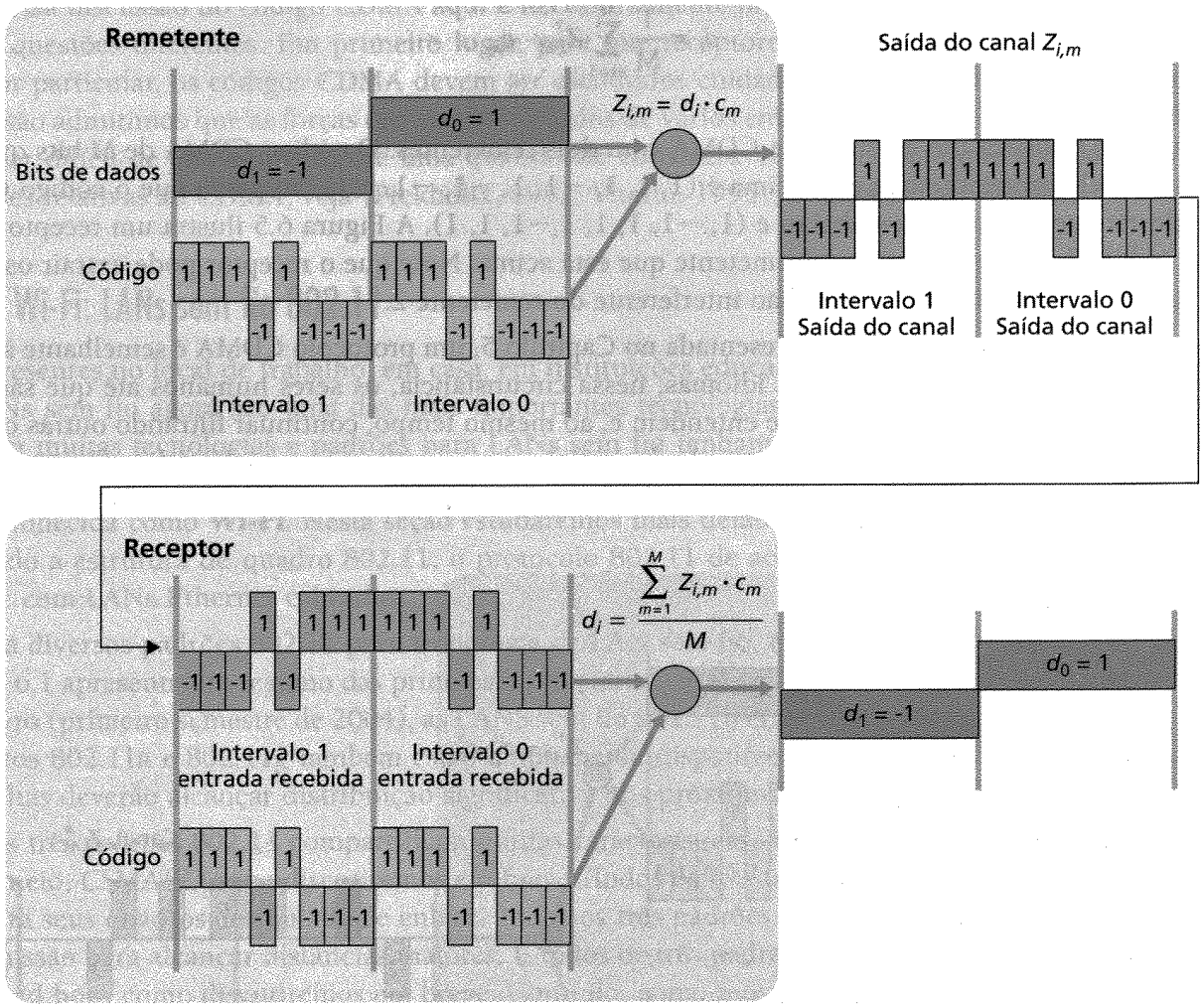


Figura 6.4 Um exemplo simples de CDMA: codificação no remetente, decodificação no receptor

No entanto, o mundo está longe de ser ideal e, como mencionamos antes, o CDMA deve funcionar na presença de remetentes que interferem e que estão codificando e transmitindo seus dados usando um código designado diferente. Mas, como um receptor CDMA pode recuperar bits de dados originais de um remetente quando esses bits de dados estão sendo embaralhados com bits que estão sendo transmitidos por outros remetentes? O CDMA funciona na presunção de que os sinais de bits interferentes que estão sendo transmitidos são aditivos. Isso significa, por exemplo, que, se três remetentes enviam um valor 1 e um quarto remetente envia um valor −1 durante o mesmo miniintervalo, então o sinal recebido em todos os receptores durante o miniintervalo é 2 (já que 1 + 1 + 1 − 1 = 2). Na presença de vários remetentes, o remetente s calcula suas transmissões codificadas, $Z^s_{i,m}$, exatamente como na Equação 6.1. O valor recebido no receptor durante o m-ésimo miniintervalo do i-ésimo intervalo de bit, contudo, é agora a *soma* dos bits transmitidos de todos os N remetentes durante o miniintervalo:

$$Z^*_{i,m} = \sum_{s=1}^{N} Z^s_{i,m}$$

Surpreendentemente, se os códigos dos remetentes forem escolhidos com cuidado, cada receptor pode recuperar os dados enviados por um dado remetente a partir do sinal agregado apenas usando o código do remetente, como na Equação 6.2:

$$d_i = \frac{1}{M}\sum_{m=1}^{M} Z_{i,m}^{*} \cdot c_m \tag{6.3}$$

A Figura 6.5 ilustra um exemplo de CDMA com dois remetentes. O código CDMA de M bits que está sendo usado pelo remetente que está acima é (1, 1, 1, −1, 1, −1, −1, −1), ao passo que o código CDMA usado pelo remetente que está embaixo é (1, −1, 1, 1, 1,−1, 1, 1). A Figura 6.5 ilustra um receptor recuperando os bits de dados originais do remetente que está acima. Note que o receptor pode extrair os dados do remetente 1, a despeito da transmissão interferente do remetente 2.

Voltando à analogia do coquetel apresentada no Capítulo 5, um protocolo CDMA é semelhante à situação em que os convidados falam vários idiomas; nessa circunstância, os seres humanos até que são bons para manter conversações no idioma que entendem e, ao mesmo tempo, continuar filtrando outras conversações. Vemos aqui que o CDMA é um protocolo de partição, pois reparte o espaço de código (e não o tempo ou a freqüência) e atribui a cada nó uma parcela dedicada do espaço de código.

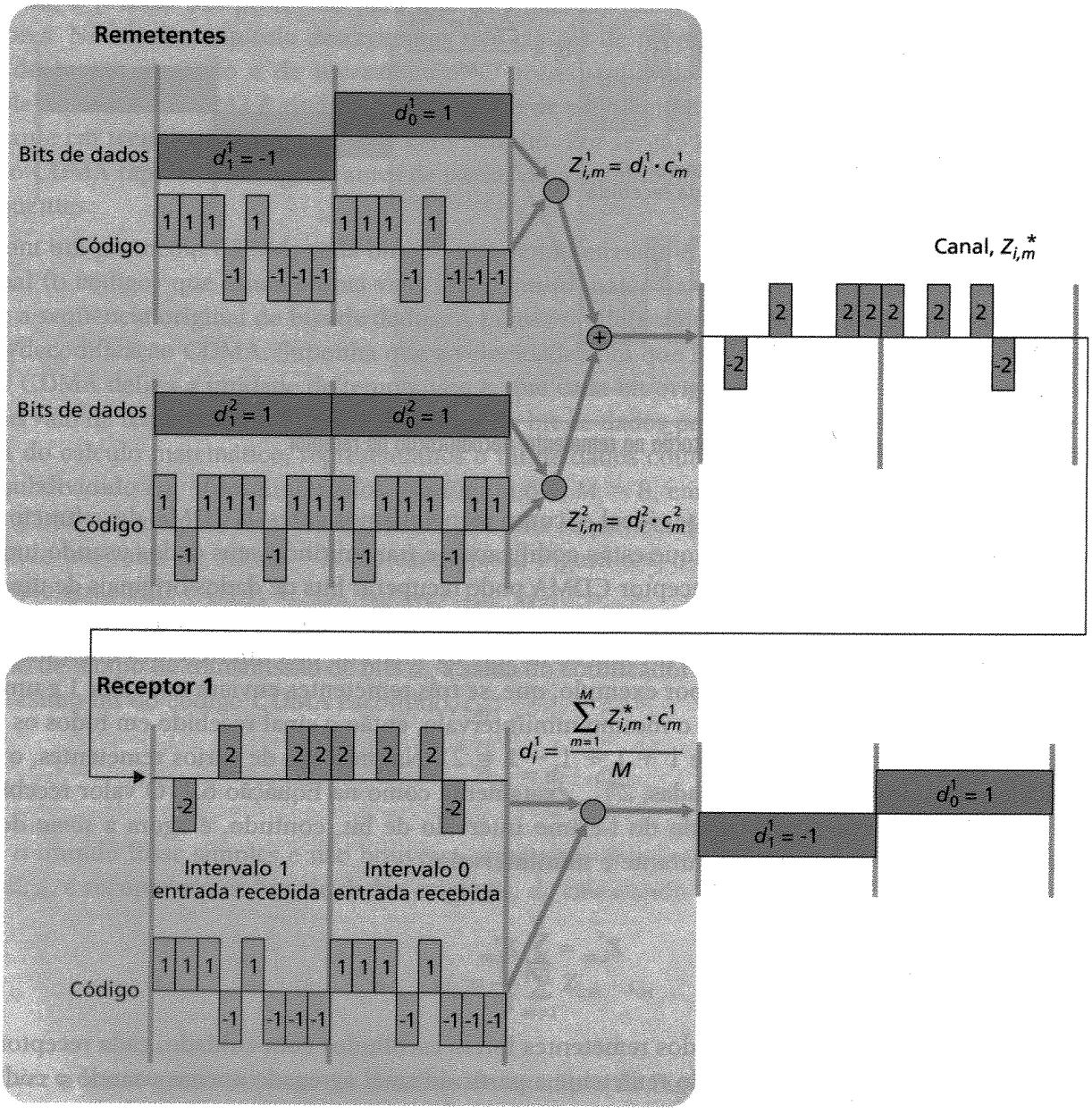


Figura 6.5 Um exemplo de CDMA com dois remetentes

Nossa discussão do código CDMA aqui é necessariamente breve; na prática devem ser abordadas inúmeras questões diferentes. Em primeiro lugar, para que receptores CDMA possam extrair o sinal de um emissor particular, os códigos CDMA devem ser escolhidos cuidadosamente. Em segundo lugar, em nossa discussão admitimos que as forças dos sinais recebidos de vários emissores são as mesmas; na realidade, isso pode ser difícil de conseguir. Existe uma quantidade considerável de literatura que aborda essas e outras questões relativas ao CDMA; veja [Pickholtz, 1982; Viterbi, 1995], se quiser mais detalhes.

6.3 Wi-Fi: LANs sem fio 802.11

Presentes no local de trabalho, em casa, em instituições educacionais, em cafés, aeroportos e esquinas, as LANs sem fio agora são uma das mais importantes tecnologias de rede de acesso na Internet de hoje. Embora muitas tecnologias e padrões para LANs sem fio tenham sido desenvolvidos na década de 1990, uma classe particular de padrões surgiu claramente como a vencedora: a **LAN sem fio IEEE 802.11**, também conhecida como **Wi-Fi**. Nesta seção estudaremos mais detalhadamente as LANs sem fio 802.11, examinando a estrutura do quadro 802.11, o protocolo 802.11 de acesso ao meio e a interconexão de LANs 802.11 com LANs Ethernet cabeadas.

Há diversos padrões 802.11 para tecnologia de LAN sem fio, entre eles 802.11b, 802.11a e 802.11g. A Tabela 6.1 apresenta um resumo das principais características desses padrões. Na época em que escrevíamos este livro (primeiro semestre de 2004), as LANS sem fio 802.11b eram de longe as predominantes. Contudo, produtos 802.11a e 802.11g também estão amplamente disponíveis, e essas LANs sem fio de velocidades mais altas deverão alcançar distribuição significativa nos próximos anos.

Os três padrões 802.11 compartilham muitas características. Todos usam o mesmo protocolo de acesso ao meio, CSMA/CA, que discutiremos em breve. Todos os três também usam a mesma estrutura de quadro para seus quadros de camada de enlace. Todos os três padrões têm a capacidade de reduzir sua taxa de transmissão para alcançar distâncias maiores. E todos os três padrões permitem 'modo de infra-estrutura' e 'modo ad hoc', como discutiremos em breve. Contudo, como mostra a Tabela 6.1, os três padrões apresentam algumas diferenças importantes na camada física.

A LAN sem fio 802.11b tem uma taxa de dados de 11 Mbps, que é mais do que suficiente para a maioria das redes residenciais com acesso à Internet por cabo de banda larga ou por DSL. LANs 802.11b operam na faixa de freqüência não licenciada de 2,4 a 2,485 GHz, competindo por espectro de freqüência com telefones e fornos de microondas de 2,4 GHz. LANs sem fio 802.11a podem funcionar a taxas de bits significativamente mais altas, porém em freqüências mais altas. Entretanto, porque operam a uma freqüência mais alta, a distância de transmissão dessas LANs é mais curta para um dado nível de potência e elas sofrem mais com a propagação multivias. LANs sem fio 802.11g, que operam na mesma faixa de freqüência mais baixa das LANS 802.11b, porém com as taxas de transmissão mais altas da 802.11a, devem permitir que os usuários tenham o melhor dos dois mundos.

Padrão	Faixa de freqüência	Taxa de dados
802.11b	2.4-2.485 GHz	até 11 Mbps
802.11a	5.1-5.8 GHz	até 54 Mbps
802.11g	2.4-2.485 GHz	até 54 Mbps

Tabela 6.1 Resumo de padrões IEEE 802.11

6.3.1 A arquitetura 802.11

A Figura 6.6 ilustra os principais componentes da arquitetura de LAN sem fio 802.11. O bloco construtivo fundamental da arquitetura 802.11 é o **conjunto básico de serviço** (*basic service set* — BSS). Um BSS contém uma ou mais estações sem fio e uma **estação-base central**, conhecida como um **ponto de acesso** (*access point* — AP) na terminologia da 802.11. A Figura 6.6 mostra o AP em cada um dos dois BSSs conectando-se a um dispositivo de interconexão (tal como um hub, um comutador ou um roteador), que, por sua vez, leva à Internet. Em uma rede residencial típica, há apenas um AP e um roteador (quase sempre acompanhado de um modem a cabo ou ADSL, formando um só pacote) que conecta o BSS à Internet.

Como acontece com dispositivos Ethernet, cada estação sem fio 802.11 tem um endereço MAC de 6 bytes que é armazenado no suporte lógico inalterável (firmware) do adaptador da estação (isto é, na placa de interface de rede 802.11). Cada AP também tem um endereço MAC para sua interface sem fio. Como na Ethernet, esses endereços MAC são administrados pelo IEEE e são (em teoria) globalmente exclusivos.

Como observamos na Seção 6.1, LANs sem fio que disponibilizam APs normalmente são denominadas **LANS sem fio de infra-estrutura** e, nesse contexto, 'infra-estrutura' quer dizer os APs juntamente com a infra-estrutura de Ethernet cabeada que interconecta os APs e um roteador. A Figura 6.7 mostra que estações IEEE 802.11 também podem se agrupar e formar uma rede ad hoc — rede sem nenhum controle central e sem nenhuma conexão com o 'mundo externo'. Nesse caso, a rede é formada conforme a necessidade, por equipamentos móveis que, por acaso, estão próximos uns dos outros, têm necessidade de se comunicar e não dispõem de infra-estrutura de rede no lugar em que se encontram. Uma rede ad hoc pode ser formada quando pessoas que portam laptops se reúnem (por exemplo, em uma sala de conferências, um trem ou um carro) e querem trocar dados na ausência de um AP centralizado. As redes ad hoc estão despertando um interesse extraordinário com a contínua proliferação de equipamentos portáteis que podem se comunicar. Porém, nesta seção, concentraremos nossa atenção em LANS sem fio de infra-estrutura.

Canais e associação

Em 802.11, cada estação sem fio precisa se associar com um AP antes de poder enviar ou receber quadros 802.11 contendo dados de camada de rede. Embora todos os padrões 802.11 usem associação, discutiremos esse tópico especificamente no contexto da IEEE 802.11b.

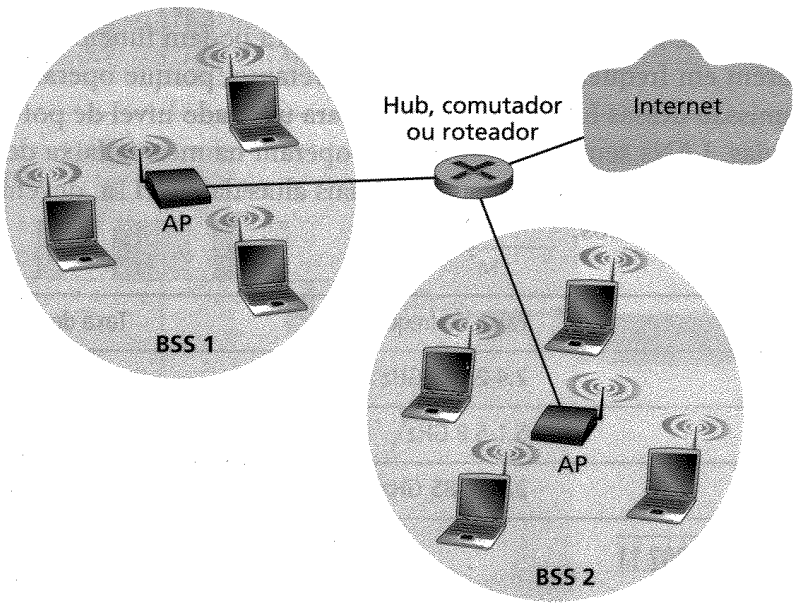


Figura 6.6 Arquitetura da LAN IEEE 802.11

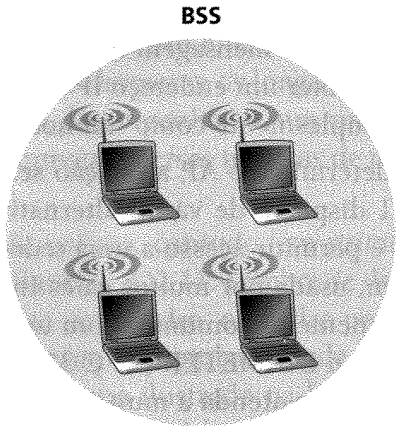


Figura 6.7 Uma rede ad hoc IEEE 802.11

Ao instalar um AP, um administrador de rede designa ao ponto de acesso um **Identificador de Conjunto de Serviços** (*Service Set Identifier* — SSID) composto de uma ou duas palavras. (O comando 'veja redes disponíveis' no Microsoft Windows XP, por exemplo, apresenta uma lista que mostra o SSID de cada AP ordenado por faixa.) O administrador também deve designar um número de canal ao AP. Para entender números de canal, lembre-se de que as redes 802.11b operam na faixa de freqüência de 2,4 GHz a 2,485 GHz. Dentro dessa faixa de 85 MHz, o padrão 802.11b define 11 canais que se sobrepõem parcialmente. Não há sobreposição entre quaisquer dois canais se, e somente se, eles estiverem separados por quatro ou mais canais. Em particular, o conjunto dos canais 1, 6 e 11 é o único conjunto de três canais não sobrepostos. Isso significa que um administrador poderia criar uma LAN sem fio com uma taxa máxima de transmissão agregada de 33 Mbps instalando três APs 802.11b na mesma localização física, designando os canais 1, 6 e 11 aos APs e interconectando cada um desses APs com um comutador.

Agora que já entendemos o básico sobre canais 802.11, vamos descrever uma situação interessante (e que não é completamente fora do comum) — uma **selva de Wi-Fis** (Wi-Fi jungle). Uma **selva de Wi-Fis** é qualquer localização física na qual uma estação sem fio recebe um sinal suficientemente forte de dois ou mais APs. Por exemplo, em muitos cafés da cidade de Nova York, uma estação sem fio pode captar um sinal de numerosos APs próximos. Um desses APs pode ser o AP gerenciado pelo café, enquanto os outros podem estar localizados em apartamentos vizinhos. Cada um desses pontos de acesso provavelmente estaria localizado em uma sub-rede diferente e o canal que utilizam teria sido designado independentemente.

Agora suponha que você entra nessa selva de Wi-Fis com seu computador portátil, em busca de acesso à Internet e de um cafezinho com biscoitos. Suponha que há cinco APS na selva. Para conseguir acesso à Internet, sua estação sem fio terá de se juntar a exatamente uma das sub-redes e, portanto, precisará se **associar** com exatamente um dos APs. Associar significa que a estação sem fio cria um fio virtual entre ela mesma e o AP. Especificamente, somente o AP associado enviará quadros de dados (isto é, quadros contendo dados, tal como um datagrama) à sua estação sem fio e sua estação sem fio enviará quadros de dados à Internet somente por meio do AP associado. Mas como sua estação sem fio se associa com um determinado AP? E, o que é mais fundamental, como sua estação sem fio sabe quais APS estão dentro da selva, se é que há algum?

O padrão 802.11b requer que um AP envie periodicamente **quadros de sinalização**, cada qual incluindo o SSID e o endereço MAC do AP. Sua estação sem fio, sabendo que os APs estão enviando quadros de sinalização, faz uma varredura dos 11 canais em busca de quadros de sinalização de quaisquer APS que possam estar por lá (alguns dos quais podem estar transmitindo no mesmo canal — afinal, estamos na selva!). Ao tomar conhecimento dos APs disponíveis por meio dos quadros de sinalização, você (ou seu hospedeiro sem fio) seleciona um desses pontos de acesso para se associar. Após selecionar o AP, seu hospedeiro sem fio e o AP escolhido dialogam usando o protocolo de associação 802.11. Se tudo correr bem nesse diálogo, sua estação sem fio fica associada com o AP selecionado. Durante a fase de associação, sua estação sem fio

está implicitamente se juntando à sub-rede à qual pertence o AP selecionado. Logo após a fase de associação, a estação sem fio normalmente enviará uma mensagem de descoberta DHCP (veja Seção 5.4.3) à sub-rede por meio do AP associado, de modo a obter um endereço IP na sub-rede do AP. Nesse ponto, o restante da Internet agora vê seu computador simplesmente como um hospedeiro na sub-rede do AP.

Para criar uma associação com um determinado AP, a estação sem fio possivelmente terá de se autenticar perante o AP. LANs sem fio 802.11 dispõem de várias alternativas para autenticação e acesso. Uma abordagem, usada por muitas empresas, é permitir acesso a uma rede sem fio com base no endereço MAC de uma estação. Uma segunda abordagem, usada por muitos cafés Internet, emprega nomes de usuários e senhas. Em ambos os casos, o AP normalmente se comunica com um servidor de autenticação usando um protocolo como o RADIUS [RFC 2138] ou o DIAMETER [RFC 3588]. Separar o servidor de autenticação do AP permite que um servidor de autenticação atenda a muitos APs, centralizando as decisões de autenticação e acesso (quase sempre delicadas) em um único servidor e mantendo baixos os custos e a complexidade do AP. Veremos, na Seção 8.8.4, que o novo protocolo IEEE 802.11i, que define aspectos de segurança da família de protocolos 802.11, adota exatamente essa abordagem.

6.3.2 O protocolo MAC 802.11

Uma vez associada com um AP, uma estação sem fio pode começar a enviar e receber quadros de dados de e para o ponto de acesso. Porém, como várias estações podem querer transmitir quadros de dados ao mesmo tempo sobre o mesmo canal, é preciso um protocolo de acesso múltiplo para coordenar as transmissões. Aqui, **estação** significa uma estação sem fio ou um AP. Como discutimos no Capítulo 5 e na Seção 6.2.1, em termos gerais há quatro classes de protocolos de acesso múltiplo: partição de canal, acesso aleatório, revezamento e CDMA. Inspirados pelo enorme sucesso da Ethernet e seu protocolo de acesso aleatório, os projetistas do 802.11 escolheram um protocolo de acesso aleatório para as LANs sem fio 802.11. Esse protocolo de acesso aleatório é denominado **CSMA com prevenção de colisão** ou, mais sucintamente, **CSMA/CA**. Do mesmo modo que o CSMA/CD da Ethernet, o 'CSMA' de CSMA/CA quer dizer 'acesso múltiplo por detecção de portadora', o que significa que cada estação sonda o canal antes de transmitir e abstém-se de transmitir quando percebe que o canal está ocupado. Embora tanto a Ethernet quanto o 802.11 usem acesso aleatório por detecção de portadora, os dois protocolos MAC apresentam diferenças importantes. Em primeiro lugar, em vez de usar detecção de colisão, o 802.11 usa técnicas de prevenção de colisão. Em segundo lugar, devido às taxas relativamente altas de erros de bits em canais sem fio, o 802.11 (diferentemente da Ethernet) usa um esquema de reconhecimento/retransmissão (ARQ) de camada de enlace. Mais adiante descreveremos os esquemas usados pelo 802.11 para prevenção de colisão e reconhecimento na camada de enlace.

Lembre-se de que nas seções 5.3 e 5.5 dissemos que, com o algoritmo de detecção de colisão, uma estação Ethernet ouve o canal à medida que transmite. Se, enquanto estiver transmitindo, a estação detectar que uma outra estação também está transmitindo, ela abortará sua transmissão e tentará transmitir novamente após uma pequena unidade de tempo aleatória. Diferentemente do protocolo Ethernet 802.3, o protocolo MAC 802.11 *não* implementa detecção de colisão. Isso se deve a duas razões importantes:

- A capacidade de detectar colisões exige as capacidades de enviar (o próprio sinal da estação) e de receber (para determinar se uma outra estação está transmitindo) ao mesmo tempo. Como a potência do sinal recebido normalmente é muito pequena em comparação com a potência do sinal transmitido no adaptador 802.11, é caro construir um hardware que possa detectar colisões.
- Mais importante, mesmo que o adaptador pudesse transmitir e ouvir ao mesmo tempo (e, presumivelmente, abortar transmissões quando percebesse um canal ocupado), ainda assim ele não seria capaz de detectar todas as colisões devido ao problema do terminal escondido e do desvanecimento, como discutimos na Seção 6.2.

Como LANs 802.11 sem fio não usam detecção de colisão, uma vez que uma estação comece a transmitir um quadro, *ela o transmite integralmente*; isto é, tão logo uma estação inicie, não há volta. Como é de esperar, transmitir quadros inteiros (particularmente os longos) quando existe grande possibilidade de colisão pode degradar significativamente o desempenho de um protocolo de acesso múltiplo. Para reduzir a probabilidade de colisões, o 802.11 emprega diversas técnicas de prevenção de colisão, que discutiremos em breve.

Antes de considerar prevenção de colisão, contudo, em primeiro lugar devemos examinar o esquema de **reconhecimento na camada de enlace** do 802.11. Lembre-se de que dissemos, na Seção 6.2, que, quando uma estação em uma LAN sem fio envia um quadro, o quadro pode não chegar intacto à estação de destino por uma variedade de razões. Para lidar com essa probabilidade não desprezível de falha, o MAC 802.11 usa reconhecimentos de camada de enlace. Como ilustrado na Figura 6.8, quando a estação de destino recebe um quadro que passou na verificação de CRC, ela espera um curto período de tempo, conhecido como **Espaçamento Curto Interquadros** (*Short Inter-Frame Spacing* — SIFS), e então devolve um quadro de reconhecimento. Se a estação transmissora não receber um reconhecimento dentro de um dado período de tempo, ela admitirá que ocorreu um erro e retransmitirá o quadro usando novamente o protocolo CSMA/CA para acessar o canal. Se a estação transmissora não receber um reconhecimento após um certo número fixo de retransmissões, desistirá e descartará o quadro.

Agora que já discutimos como o 802.11 usa reconhecimentos de camada de enlace, estamos prontos para descrever o protocolo CSMA/CA 802.11. Suponha que uma estação qualquer (pode ser uma estação sem fio ou um AP) tenha um quadro para transmitir.

1. Se inicialmente a estação perceber que o canal está ocioso, ela transmitirá seu quadro após um curto período de tempo conhecido como **Espaçamento Interquadros Distribuído** (*Distributed Inter-Frame Space* — DIFS); ver Figura 6.8.

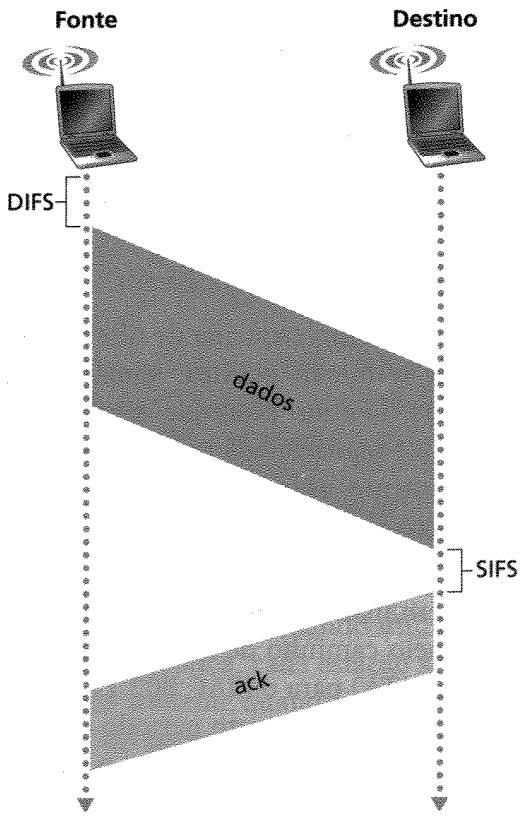


Figura 6.8 802.11 usa reconhecimentos de camada de enlace

2. Caso contrário, a estação escolherá um valor aleatório de backoff e fará a contagem regressiva a partir desse valor quando perceber que o canal estiver ocioso. Se a estação perceber que o canal está ocupado, o valor do contador permanecerá estacionário.
3. Quando o contador chegar ao zero (note que isso pode ocorrer somente quando a estação percebe que o canal está ocioso), a estação transmitirá o quadro inteiro e então ficará esperando um reconhecimento.
4. Se receber um reconhecimento, a estação transmissora saberá que esse quadro foi corretamente recebido na estação de destino. Se a estação tiver outro quadro para transmitir, iniciará o protocolo CSMA/CA na etapa 2. Se não receber um reconhecimento, a estação transmissora entrará novamente na fase de backoff na etapa 2 e escolherá um valor aleatório dentro de um intervalo maior.

O leitor observador talvez tenha notado que, na etapa 2, a estação escolhe um valor de backoff aleatório e inicia uma contagem regressiva atrasando, efetivamente, sua transmissão, mesmo que perceba que o canal está ocioso. Contudo, sob o protocolo de acesso múltiplo CSMA/CD (Seção 5.5.2), uma estação começa a transmitir tão logo percebe que o canal está ocioso. Por que o CSMA/CD e o CDMA/CA adotam essas abordagens diferentes aqui?

Para responder a essa pergunta, vamos considerar um cenário com duas estações em que cada uma tem um quadro a transmitir, mas nenhuma transmite imediatamente porque percebe que uma terceira estação já está transmitindo. Com o CSMA/CD da Ethernet, cada uma das duas estações transmitiria tão logo detectasse que a terceira estação terminou de transmitir. Isso causaria uma colisão, o que não é uma questão séria em CSMA/CD, já que ambas as estações abortariam suas transmissões e assim evitariam a transmissão inútil do restante de um quadro que tivesse sofrido uma colisão. Entretanto, com 802.11 a situação é bem diferente. Como o 802.11 não detecta uma colisão nem aborta transmissão, um quadro que sofra uma colisão será transmitido integralmente. Assim, a meta do 802.11 é evitar colisões sempre que possível. Com esse protocolo, se duas estações perceberem que o canal está ocupado, ambas entrarão imediatamente em backoff aleatório e, esperamos, escolherão valores diferentes de backoff. Se esses valores forem, de fato, diferentes, assim que o canal ficar ocioso, uma das duas começará a transmitir antes da outra e (se as duas estações não estiverem ocultas uma da outra) a 'estação perdedora' ouvirá o sinal da 'estação vencedora', interromperá seu contador e não transmitirá até que a estação vencedora tenha concluído sua transmissão. Desse modo é evitada uma colisão dispendiosa. É claro que ainda podem ocorrer colisões com 802.11 nesse cenário: as duas estações podem estar ocultas uma da outra ou podem escolher valores idênticos de backoff aleatório.

Terminais ocultos: RTS e CTS

O protocolo 802.11 MAC também inclui um esquema de reserva inteligente (mas opcional) que ajuda a evitar colisões mesmo na presença de terminais ocultos. Vamos estudar esse esquema no contexto da Figura 6.9, que mostra duas estações sem fio e um ponto de acesso. Ambas as estações sem fio estão dentro da faixa do AP (cuja área de cobertura é representada por um círculo sombreado) e ambas se associaram com o AP. Contudo, devido ao desvanecimento, as faixas de sinal de estações sem fio estão limitadas ao interior dos círculos sombreados mostrados na Figura 6.9. Assim, cada uma das estações sem fio está oculta da outra, embora nenhuma esteja oculta do AP.

Agora vamos considerar por que terminais ocultos podem ser problemáticos. Suponha que a estação H1 está transmitindo um quadro e, a meio caminho de sua transmissão, a camada de rede na estação H2 passa um quadro (que denominaremos DATA) para o MAC 802.11. Como a estação H2 não está ouvindo a transmissão de H1, ela primeiramente esperará um curto período de tempo aleatório e então transmitirá o quadro DATA, resultando em uma colisão. Conseqüentemente, o canal será desperdiçado durante todo o período da transmissão de H1, bem como durante a transmissão de H2.

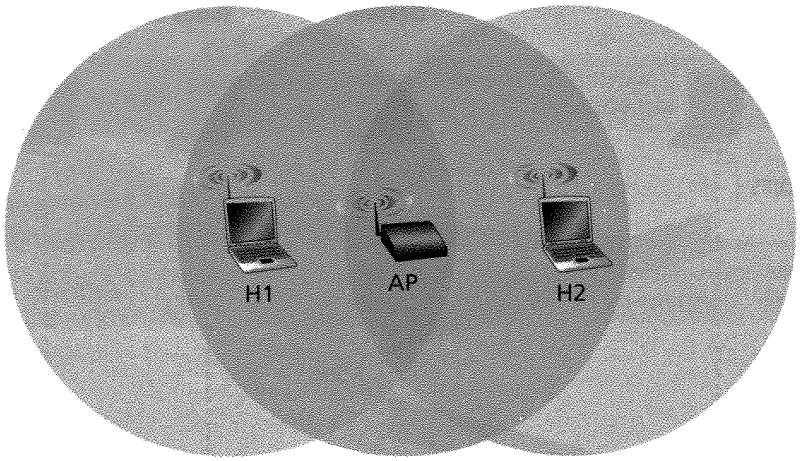


Figura 6.9 Exemplo de terminal oculto: H1 está oculto de H2, e vice-versa

Para evitar esse problema, o protocolo IEEE 802.11 permite que uma estação utilize um quadro de controle **RTS** curto (*Request to Send* — solicitação de envio) e um quadro de controle **CTS** curto (*Clear to Send* — pronto para envio) para *reservar* acesso ao canal. Quando um remetente quer enviar um quadro DATA, ele pode enviar primeiramente um quadro RTS ao AP indicando o tempo total requerido para transmitir o quadro DATA e o quadro de reconhecimento (ACK). Quando o AP recebe o quadro RTS, responde fazendo a transmissão *broadcast* de um quadro CTS. Esse quadro CTS tem duas finalidades: dá ao remetente uma permissão explícita para enviar e também instrui as outras estações a não enviar durante o tempo reservado.

Assim, na Figura 6.10, antes de transmitir um quadro DATA, H1 primeiramente faz uma transmissão *broadcast* de um quadro RTS, que é ouvida por todas as estações que estiverem dentro do seu círculo de alcance, incluindo o AP. O AP então responde com um quadro CTS, que é ouvido por todas as estações dentro de sua faixa de alcance, incluindo H1 e H2. Como ouviu o CTS, a estação H2 deixa de transmitir durante o tempo especificado no quadro CTS. Os quadros RTS, CTS, DATA e ACK são mostrados na Figura 6.10.

A utilização dos quadros RTS e CTS pode melhorar o desempenho de dois modos importantes:

- O problema da estação oculta é atenuado, visto que um quadro DATA longo é transmitido somente após o canal ter sido reservado.
- Como os quadros RTS e CTS são curtos, uma colisão que envolva um quadro RTS ou CTS terá apenas a duração dos quadros RTS ou CTS curtos. Uma vez que os quadros RTS e CTS sejam corretamente transmitidos, os quadros DATA e ACK subseqüentes deverão ser transmitidos sem colisões.

Aconselhamos o leitor a verificar o applet 802.11 no site Web deste livro. Esse applet interativo ilustra o protocolo CSMA/CA, incluindo a seqüência de troca RTS/CTS.

Embora a troca RTS/CTS possa ajudar a reduzir colisões, também introduz atraso e consome recursos do canal. Por essa razão, a troca RTS/CTS é utilizada (quando utilizada) apenas para reservar o canal para a transmissão de um quadro DATA longo. Na prática, cada estação sem fio pode estabelecer um patamar RTS tal que a seqüência RTS/CTS seja utilizada somente quando o quadro for mais longo do que o patamar. Para muitas estações sem fio, o valor default do patamar RTS é maior do que o comprimento máximo do quadro, de modo que a seqüência RTS/CTS é omitida para todos os quadros DATA enviados.

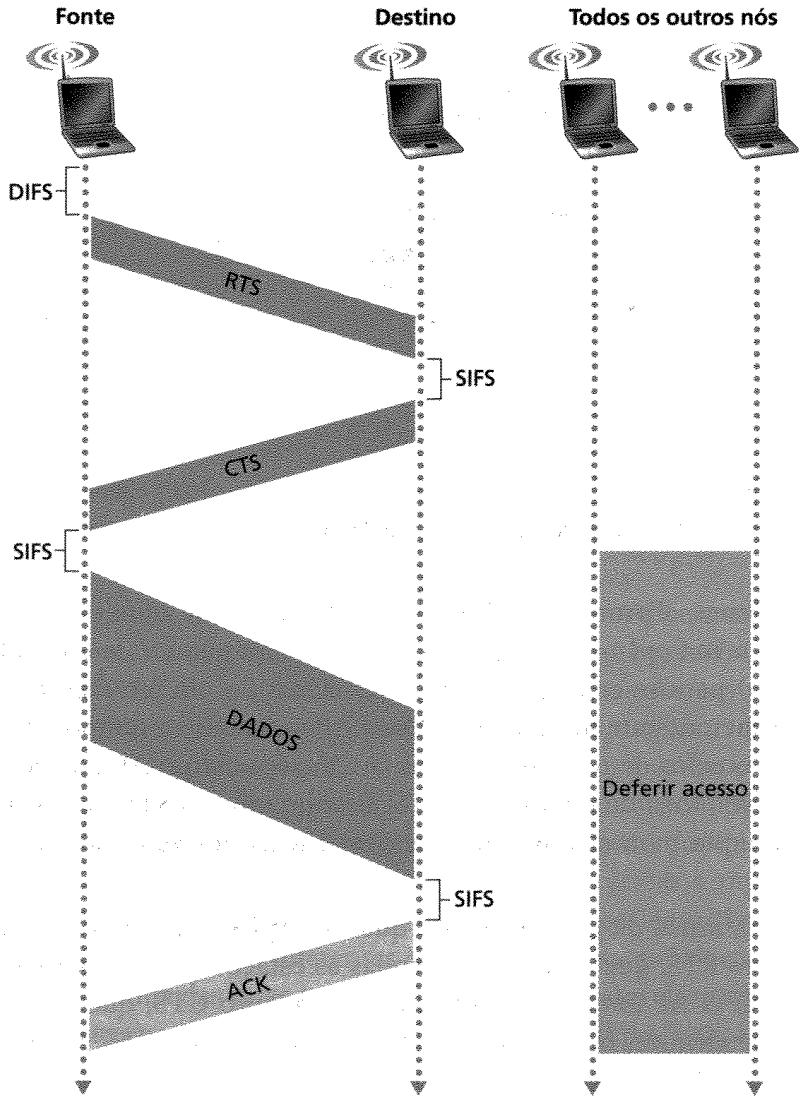


Figura 6.10 Prevenção de colisão usando os quadros RTS e CTS

Usando o 802.11 como enlace ponto-a-ponto

Até aqui nossa discussão focalizou a utilização do 802.11 em um cenário de múltiplo acesso. Devemos mencionar que, se dois nós tiverem, cada um, uma antena direcional, eles poderão dirigir suas antenas um para o outro e executar o protocolo 802.11 sobre o que é, essencialmente, um enlace ponto-a-ponto. Dado o baixo custo comercial do hardware 802.11, a utilização de antenas direcionais e uma maior potência de transmissão permitem que o 802.11 seja utilizado como um meio barato de prover conexões sem fio ponto-a-ponto por dezenas de quilômetros. [Bhagwat, 2003] descreve uma dessas redes sem fio multi-saltos que funciona nas planícies rurais do rio Ganges, na Índia, e que contém enlaces 802.11 ponto-a-ponto.

6.3.3 O quadro IEEE 802.11

Embora o quadro 802.11 tenha muitas semelhanças com um quadro Ethernet, ele também contém vários campos que são específicos para sua utilização para enlaces sem fio. O quadro 802.11 é mostrado na Figura 6.11. Os números acima de cada um dos campos no quadro representam os comprimentos dos campos em *bytes*; os números acima de cada um dos subcampos no campo de controle do quadro representam os comprimentos dos subcampos em *bits*. Agora vamos examinar os campos no quadro, bem como alguns dos subcampos mais importantes no campo de controle do quadro.

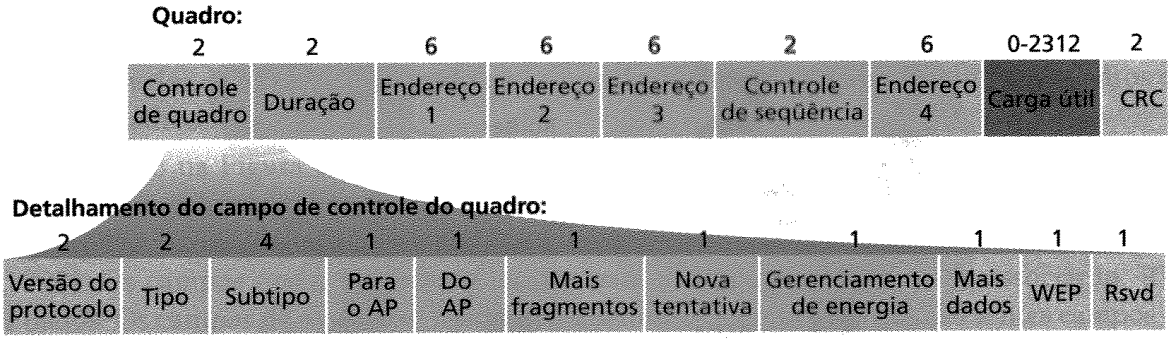


Figura 6.11 O quadro 802.11

Campos de carga útil e de CRC

No coração do quadro está a carga útil, que consiste, tipicamente, em um datagrama IP ou em um pacote ARP. Embora o comprimento permitido do campo seja 2.312 bytes, normalmente ele é menor do que 1.500 bytes, contendo um datagrama IP ou um pacote ARP. Como um quadro Ethernet, um quadro 802.11 inclui uma verificação de redundância cíclica (CRC), de modo que o receptor possa detectar erros de bits no quadro recebido. Como já vimos, erros de bits são muito mais comuns em LANs sem fio do que em LANs cabeadas, portanto, aqui, CRC é ainda mais útil.

Campos de endereço

Talvez a diferença mais marcante no quadro 802.11 é que ele tem *quatro* campos de endereço e cada um pode conter um endereço MAC de 6 bytes. Mas por que quatro campos de endereço? Um campo de fonte MAC e um campo de destino MAC não são suficientes como são na Ethernet? Acontece que aqueles três campos de endereço são necessários para finalidades de interconexão em rede — especificamente, para mover o datagrama de camada de enlace de uma estação sem fio, passando por um AP, até uma interface de roteador. O quarto campo de endereço é usado em redes ad hoc, mas não em redes de infra-estrutura. Visto que estamos considerando apenas redes de infra-estrutura, vamos concentrar nossa atenção nos três primeiros campos de endereço. O padrão 802.11 define esses campos como se segue:

- Endereço 2 é o endereço MAC da estação que transmite o quadro. Assim, se uma estação sem fio transmitir o quadro, o endereço MAC daquela estação será inserido no campo de endereço 2. De modo semelhante, se um AP transmitir o quadro, o endereço MAC do AP será inserido no campo de endereço 2.
- Endereço 1 é o endereço MAC da estação sem fio que deve receber o quadro. Assim, se uma estação móvel sem fio transmitir o quadro, o endereço 1 conterá o endereço MAC do AP de destino. De modo semelhante, se um AP transmitir o quadro, o endereço 1 conterá o endereço MAC da estação sem fio destinatária.
- Para entender o endereço 3, lembre-se de que o BSS (que consiste no AP e estações sem fio) é parte de uma sub-rede, e que essa sub-rede se conecta com outras sub-redes, via alguma interface de roteador.

Para compreender melhor a finalidade do endereço 3, vamos examinar um exemplo de interconexão em rede no contexto da Figura 6.12. Nessa figura há dois APs, cada um responsável por um certo número de estações sem fio. Cada um dos APs tem uma conexão direta com um roteador, que, por sua vez, se conecta com a Internet global. Devemos ter sempre em mente que um AP é um dispositivo de camada de enlace e, portanto, não 'fala' IP nem entende endereços IP. Agora, considere mover um datagrama da interface de roteador R1 até a estação sem fio H1. O roteador não está ciente de que há um AP entre ele e H1; da perspectiva do roteador, H1 é apenas um hospedeiro em uma das sub-redes às quais ele (o roteador) está conectado.

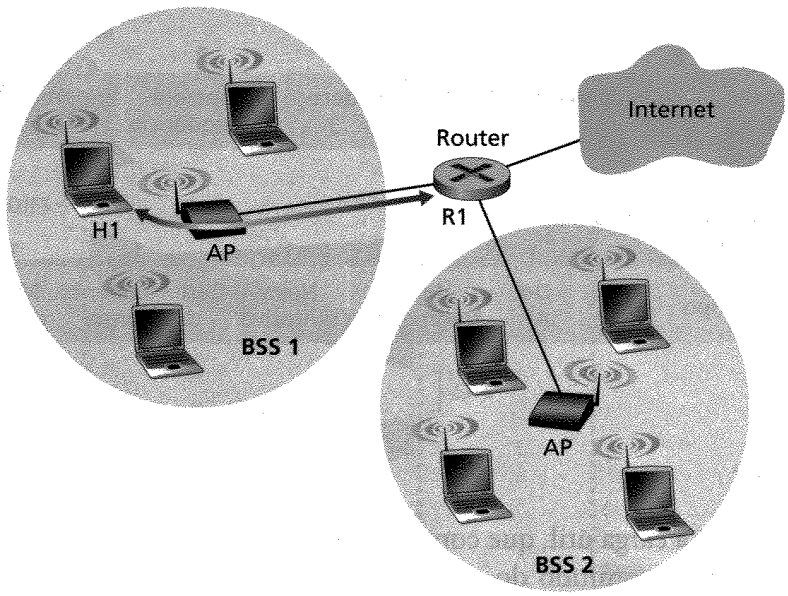


Figura 6.12 A utilização de campos de endereço em quadros 802.11: movendo um quadro entre H1 e R1

- O roteador, que conhece o endereço IP de H1 (pelo endereço de destino do datagrama), utiliza ARP para determinar o endereço MAC de H1, exatamente como aconteceria em uma LAN Ethernet comum. Após obter o endereço MAC de H1, a interface de roteador R1 encapsula o datagrama em um quadro Ethernet. O campo de endereço de fonte desse quadro contém o endereço MAC de R1 e o campo de endereço de destino contém o endereço MAC de H1.
- Quando o quadro Ethernet chega ao AP, este converte o quadro Ethernet 802.3 para um quadro 802.11 antes de transmiti-lo para o canal sem fio. O AP preenche o endereço 1 e o endereço 2 com o endereço MAC de H1 e seu próprio endereço MAC, respectivamente, como descrito acima. Para o endereço 3, o AP insere o endereço MAC de R1. Dessa maneira, H1 pode determinar (a partir do endereço 3) o endereço MAC da interface de roteador que enviou o datagrama para a sub-rede.

Agora considere o que acontece quando a estação sem fio H1 responde movendo um datagrama de H1 para R1.

- H1 cria um quadro 802.11, preenchendo os campos de endereço 1 e 2 com o endereço MAC do AP e com o endereço MAC de H1, respectivamente, como descrito acima. Para o endereço 3, H1 insere o endereço MAC de R1.
- Quando recebe o quadro 802.11, o AP o converte para um quadro Ethernet. O campo de endereço de fonte para esse quadro é o endereço MAC de H1 e o campo de endereço de destino é o endereço MAC de R1. Assim, o endereço 3 permite que o AP determine o endereço de destino MAC apropriado ao construir o quadro Ethernet.

Em resumo, o endereço 3 desempenha um papel crucial na interconexão do BSS com uma LAN cabeada.

Campos de número de seqüência, duração e controle de quadro

Lembre-se de que, em 802.11, sempre que uma estação recebe corretamente um quadro de uma outra estação, devolve um reconhecimento. Como reconhecimentos podem ser perdidos, a estação emissora pode enviar várias cópias de um dado quadro. Como vimos na nossa discussão sobre o protocolo rdt2.1 (Seção 3.4.1), a utilização de números de seqüência permite que o receptor distinga entre um quadro recém-transmitido e a retransmissão de um quadro anterior. Assim, o campo de número de seqüência no quadro 802.11 cumpre aqui, na camada de enlace, exatamente a mesma finalidade que cumpria na camada de transporte do Capítulo 3.

Não se esqueça de que o protocolo 802.11 permite que uma estação transmissora reserve o canal durante um período que inclui o tempo para transmitir seu quadro de dados e o tempo para transmitir um reconhecimento. Esse valor de duração é incluído no campo de duração do quadro (tanto para quadros de dados quanto para os quadros RTS e CTS).

Como mostrado na Figura 6.11, o campo de controle de quadro inclui muitos subcampos. Diremos apenas umas poucas palavras sobre alguns dos subcampos mais importantes; se o leitor quiser uma discussão mais completa, aconselhamos que consulte a especificação 802.11 [Held, 2001; Crow, 1997; IEEE 802.11, 1999]. Os campos *tipo* e *subtipo* são usados para distinguir os quadros de associação, RTS, CTS, ACK e de dados. Os campos de e para são usados para definir os significados dos diferentes campos de endereço. (Esses significados mudam dependendo da utilização dos modos ad hoc ou de infra-estrutura e, no caso do modo de infra-estrutura, mudam dependendo de o emissor do quadro ser uma estação sem fio ou um AP.) Finalmente, o campo WEP (*wireless equivalent privacy*) indica se está sendo ou não utilizada criptografia. (A WEP é discutida no Capítulo 8.)

6.3.4 Mobilidade na mesma sub-rede IP

Para ampliar a faixa física de uma LAN sem fio, empresas e universidades freqüentemente distribuirão vários BSSs dentro da mesma sub-rede IP. Isso naturalmente levanta a questão da mobilidade entre os BSSs — como estações sem fio passam ininterrupta e imperceptivelmente de um BSS para outro enquanto mantêm sessões TCP em curso? Como veremos nesta subseção, a mobilidade pode ser manipulada de uma maneira relativamente direta quando os BSSs são parte de uma sub-rede. Quando estações se movimentam entre sub-redes, são necessários protocolos de gerenciamento de mobilidade mais sofisticados, tais como os que estudaremos nas seções 6.5 e 6.6.

Agora vamos examinar um exemplo específico de mobilidade entre BSSs na mesma sub-rede. A Figura 6.13 mostra dois BSSs interconectados e um hospedeiro, H1, que se move entre BSS1 e BSS2. Como nesse exemplo o dispositivo de interconexão entre os dois BSSs *não* é um roteador, todas as estações nos dois BSSs, incluindo os APs, pertencem à mesma sub-rede IP. Assim, quando H1 se move de BSS1 para BSS2, ele pode manter seu endereço IP e todas as suas conexões TCP em curso. Se o dispositivo de interconexão fosse um roteador, então H1 teria de trocar seu endereço IP e ou descartar suas conexões TCP em curso, ou utilizar um protocolo de mobilidade de camada de rede, tal como o IP móvel, como discutido na Seção 6.6.

Mas o que acontece especificamente quando H1 passa de BSS1 para BSS2? À medida que se afasta de AP1, o H1 detecta um enfraquecimento do sinal de AP1 e começa a fazer uma varredura em busca de um sinal mais forte. H1 recebe quadros de sinalização de AP2 (que em muitos ambientes empresariais e universitários terão o mesmo SSID do AP1). Então, H1 se desassocia de AP1 e se associa com AP2, mantendo, ao mesmo tempo, seu endereço IP e suas sessões TCP em curso.

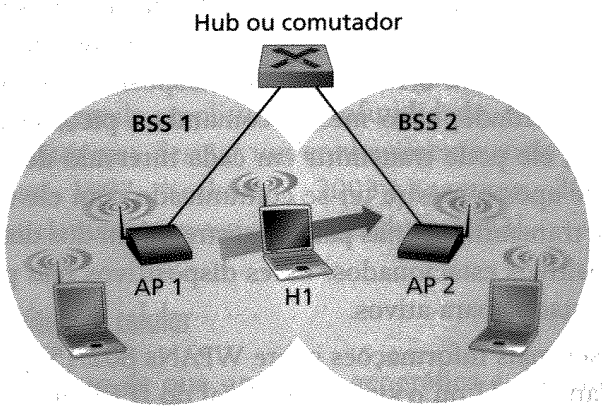


Figura 6.13 Mobilidade na mesma sub-rede

Tudo isso funciona muito bem se o dispositivo de interconexão for um hub. Mas, se for um comutador — como acontece freqüentemente —, então será preciso tomar um cuidado especial. O leitor talvez se lembre de que, no Capítulo 5, dissemos que comutadores observam os endereços e constroem automaticamente suas tabelas de repasse. Essa característica de auto-aprendizagem funciona bem para movimentações ocasionais (por exemplo, quando um profissional é transferido de um departamento para outro); contudo, comutadores não são projetados para suportar usuários com alto grau de mobilidade, que querem manter conexões TCP enquanto se movimentam entre BSSs. Para avaliar este problema aqui, lembre-se de que, antes da movimentação, o comutador tem um registro em sua tabela de repasse que vincula o endereço MAC de H1 com a interface de saída do comutador por meio da qual o H1 pode ser alcançado. Se H1 estiver inicialmente em BSS1, então um datagrama destinado a H1 será direcionado a ele via AP1. Contudo, tão logo H1 se associe com BSS2, seus quadros deverão ser direcionados para AP2. Uma solução (na verdade um tanto forçada) é o AP2 enviar ao comutador um quadro Ethernet *broadcast* com o endereço de fonte de H1 logo após a nova associação. Quando o comutador recebe o quadro, atualiza sua tabela de repasse, permitindo que H1 seja alcançado via AP2. O grupo de padrões 802.11 está desenvolvendo um protocolo inter-AP para manipular essas e outras questões associadas.

6.3.5 802.15 e Bluetooth

Como observado na Figura 6.2, o padrão IEEE 802.11 Wi-Fi visa à comunicação entre dispositivos separados por até 100 metros, enquanto o acesso à rede por sistemas de telefonia celular abrange a faixa de dezenas de quilômetros. Antes de entrar nos detalhes dessas redes celulares sem fio de maiores distâncias, vamos considerar brevemente o padrão 802.15, denominado **padrão de rede de área pessoal sem fio** (*wireless personal area network* — WPAN), um primo do 802.11 na família IEEE 802, cuja finalidade é prover conectividade entre dispositivos pessoais separados por até 10 metros, aproximadamente. O 802.15 é essencialmente uma tecnologia de 'substituição de cabo' de baixa potência, curto alcance e baixa velocidade para interconectar notebooks, equipamentos periféricos, telefones celulares e PDAs, ao passo que o 802.11 é uma tecnologia 'de acesso' de potência mais alta, alcance médio e velocidade mais alta.

Uma rede IEEE 802.15 funciona sobre uma área pequena, com baixa potência e a baixo custo. As camadas de enlace e física da rede 802.15 são baseadas na especificação original do **Bluetooth** para redes pessoais [Held, 2001; Bisdikian, 2001]. Redes 802.15 operam em faixa de rádio não licenciada de 2,4 GHz em modo TDM, com intervalos de tempo (time slots) de 625 microssegundos. Durante cada intervalo de tempo, um emissor transmite por um entre 79 canais, sendo que, de intervalo para intervalo, o canal muda de uma maneira conhecida, porém pseudo-aleatória. Essa forma de saltar de canal em canal (hopping), conhecida como ***frequency-hopping spread spectrum*** (FHSS), espalha transmissões sobre o espectro de freqüência ao longo do tempo. O 802.15 pode prover velocidades de dados de até 721 kbps.

Redes 802.15 são redes ad hoc: não é preciso nenhuma infra-estrutura de rede (por exemplo, um ponto de acesso) para interconectar dispositivos 802.15. Assim, esses dispositivos devem se organizar por si sós. Dispositivos 802.15 são primeiramente organizados em uma **picorrede** (piconet: pequena rede) de até oito dispositivos ativos, como mostra a Figura 6.14. Um desses dispositivos é designado como o mestre, e os outros agem como escravos. Na verdade, o nó mestre comanda a picorrede como um rei — seu relógio determina o tempo na picorrede, ele pode transmitir em cada intervalo de tempo de número ímpar e um escravo pode transmitir somente após o mestre ter-se comunicado com ele no intervalo de tempo anterior e, mesmo assim, o escravo pode transmitir apenas para o mestre. Além dos dispositivos escravos, a rede também pode conter até 255 dispositivos estacionados. Esses dispositivos não podem se comunicar até que o nó mestre tenha mudado seus estados para ativos.

Se o leitor interessado quiser mais informações sobre WPANs 802.15, pode consultar as referências do Bluetooth [Held, 2001; Bisdikian, 2001] ou o site Web oficial do IEEE 802.15 [IEEE 802.15, 2004].

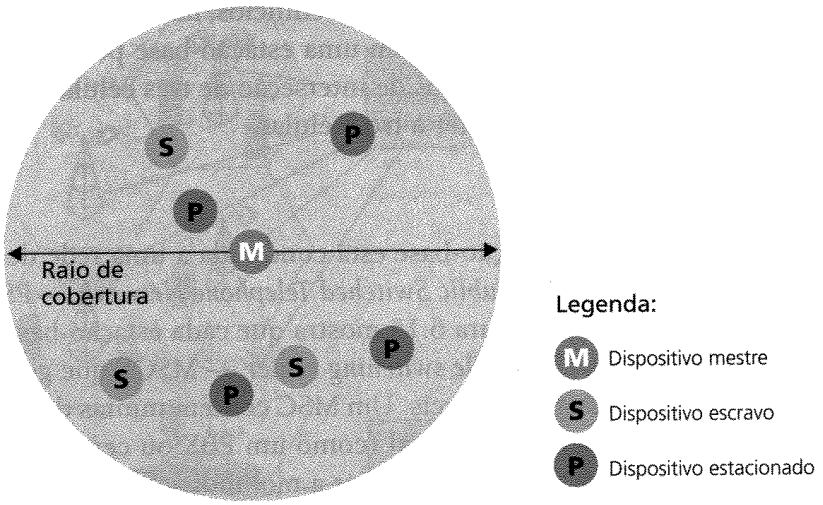


Figura 6.14 Uma picorrede 802.15

6.4 Acesso celular à Internet

Na seção anterior vimos como um hospedeiro da Internet pode acessar a Internet quando estiver dentro de um hotspot Wi-Fi, isto é, quando estiver nas vizinhanças de um ponto de acesso 802.11. Mas a área de cobertura da maioria dos hotspots Wi-Fi é pequena, entre 10 e 100 metros de diâmetro. O que fazer, então, quando precisamos desesperadamente de um acesso sem fio à Internet e não podemos acessar nenhum hotspot Wi-Fi?

Dado que a telefonia celular agora está presente por toda a parte, em muitas áreas no mundo inteiro, uma estratégia natural é estender redes celulares de modo que suportem não somente telefonia de voz, mas também acesso sem fio à Internet. Idealmente, esse acesso à Internet teria uma velocidade razoavelmente alta e proveria mobilidade imperceptível e ininterrupta, permitindo que usuários mantivessem suas sessões TCP enquanto em trânsito, por exemplo, em um ônibus ou em um trem. Com taxas de bits suficientemente altas entre a fonte e o usuário e vice-versa, o usuário poderia até mesmo manter sessões de videoconferência enquanto em trânsito. Esse cenário não é assim tão implausível. Quando escrevíamos este livro (início de 2004), muitas provedoras de telefonia celular ofereciam a seus assinantes um serviço de acesso celular à Internet por menos de 100 dólares mensais com taxas de bits entre a provedora e o usuário e vice-versa na faixa de algumas centenas de kilobits por segundo.

Nesta seção, damos uma breve visão geral das tecnologias de acesso celular à Internet já existentes e emergentes. Mais uma vez, aqui focalizaremos o primeiro salto sem fio entre o telefone celular e a infraestrutura de rede de telefonia cabeada; na Seção 6.7 consideraremos como chamadas são roteadas para um usuário que está se movimentando entre estações-base. Nossa breve discussão proverá, necessariamente, somente uma descrição simplificada e de alto nível de tecnologias celulares. É claro que a questão das comunicações celulares modernas é de grande amplitude e profundidade, e muitas universidades oferecem diversos cursos sobre o assunto. O leitor que desejar um conhecimento mais profundo da questão deve consultar [Goodman, 1997; Scourias, 1997; Korhonen, 2003; Kaaranen, 2001; Lin, 2001], bem como as referências particularmente excelentes e exaustivas em [Mouly, 1992].

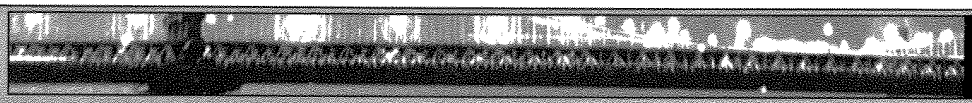
6.4.1 Uma visão geral da arquitetura celular

O termo *celular* refere-se ao fato de que uma área geográfica é dividida em várias áreas de cobertura geográfica, conhecidas como **células**, como ilustrado no lado esquerdo da Figura 6.15. Cada célula contém uma estação-base que transmite sinais para, e recebe sinais de, estações móveis dentro de sua célula. A área de cobertura de uma célula depende de muitos fatores, incluindo potência de transmissão da estação-base,

potência de transmissão da estação móvel, obstáculos, como edifícios, na célula, e altura das antenas da estação-base. Embora a Figura 6.15 mostre cada célula com uma estação-base posicionada no seu meio, hoje, muitos sistemas posicionam estações-base em pontos de interseção de três células, de modo que uma única estação-base com antenas direcionais possa atender a três células.

Arquitetura básica da rede

Como ilustrado na Figura 6.15, cada estação-base está conectada a uma rede de longa distância — tal como uma rede telefônica pública comutada (*Public Switched Telephone Network* — PSTN) por meio de uma infra-estrutura cabeada. Especificamente, a Figura 6.15 mostra que cada estação-base está conectada a uma **central de comutação de unidade móvel** (*mobile switching center* — MSC), que gerencia o estabelecimento e o término de chamadas de e para usuários móveis. Um MSC contém muitas das funcionalidades encontradas em uma central de comutação telefônica normal (como um PBX ou central telefônica), mas ampliada com a funcionalidade adicional requerida para gerenciar a mobilidade de seus usuários.



História

Celular móvel 3G versus LANS sem fio

Muitas operadoras de telefonia celular móvel estão disponibilizando sistemas celulares móveis 3G, com taxas de dados de 2 Mbps em ambiente fechado e 384 kbps ao ar livre. Os sistemas 3G estão sendo disponibilizados em faixas licenciadas de radiofreqüência, sendo que o preço que algumas operadoras pagam aos governos pela licença chega a 2 mil dólares por assinante. Os sistemas 3G permitirão que os usuários acessem a Internet a partir de localizações remotas externas enquanto em trânsito, de um modo semelhante ao acesso a telefones celulares existente hoje. Por exemplo, a tecnologia 3G permitirá que um usuário acesse informações sobre estradas enquanto dirige, ou informações sobre a programação de cinema enquanto estiver tomando sol na praia. Não obstante, hoje muitos especialistas estão começando a questionar se a tecnologia 3G terá sucesso, dados o seu custo e a concorrência da tecnologia de LAN sem fio [Weinstein, 2002]. Em particular, esses especialistas argumentam que:

- A infra-estrutura emergente de LAN sem fio logo se tornará quase ubíqua. As LANs sem fio IEEE 802.11, que operam a taxas de 11 Mbps e mais altas, estão sendo amplamente disponibilizadas, como vimos em um quadro anterior (Acesso público Wi-Fi). Dentro de pouco tempo, quase todos os computadores portáteis e PDAS já virão da fábrica equipados com placas de LAN 802.11. Além disso, as novidades em equipamentos de Internet — como câmeras sem fio e molduras para fotografias — também usarão pequenas placas de LAN sem fio de baixa potência.
- Grande parte do tráfego de dados sem fio terá origem ou destino em ambientes locais. Admitindo-se que o tráfego sem fio que se origina/termina em centros comerciais, edifícios de escritórios e assim por diante seja carregado pelas LANs sem fio de baixo custo, sobrará relativamente pouco tráfego para os caros sistemas 3G.
- Estações-base de LANs sem fio também podem manipular equipamentos de telefonia móvel, o que permite baixo custo de acesso a dados para equipamentos celulares móveis, bem como equipamentos de LAN sem fio. Assim, um telefone celular que não tenha uma placa de LAN sem fio, mas esteja presente em um ambiente local, poderá evitar os sistemas 3G das operadoras para acessar a Internet.

É claro que muitos outros especialistas acham que a tecnologia 3G não somente será o maior sucesso, mas também revolucionará drasticamente o modo como trabalhamos e vivemos. É claro que ambas, Wi-Fi e 3G, podem se tornar tecnologias sem fio dominantes, com equipamentos sem fio em trânsito que selecionam automaticamente a tecnologia de acesso que provê o melhor serviço em sua localização física corrente (veja a discussão de acesso sem fio 4G nesta seção).

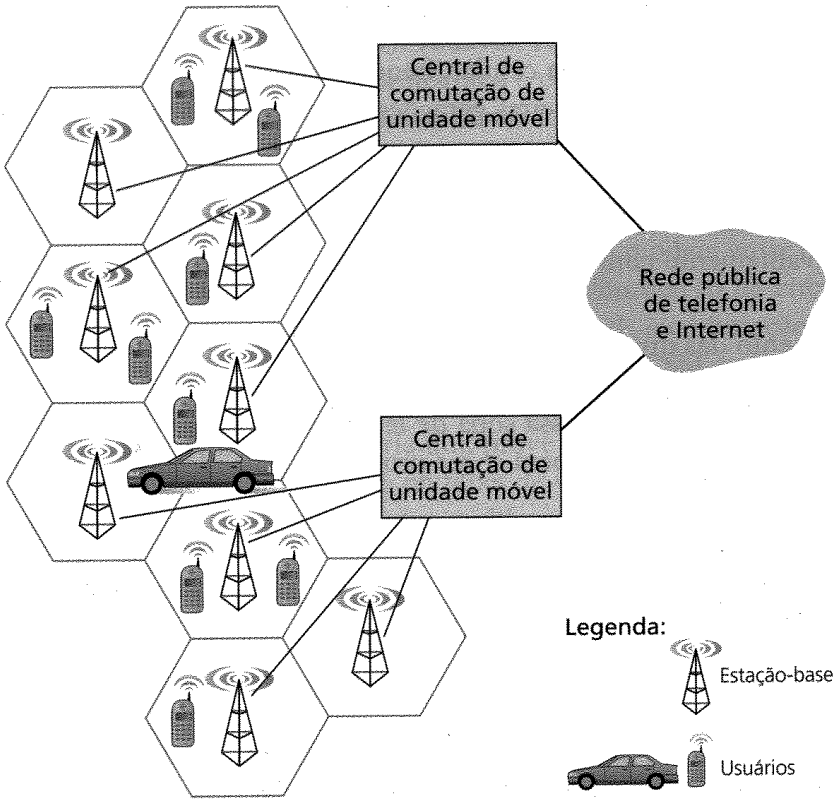


Figura 6.15 Componentes de uma arquitetura de rede celular

Técnicas de acesso à interface ar

Normalmente ocorrem muitas chamadas simultâneas em uma dada célula. Essas chamadas precisam compartilhar a porção do espectro de rádio alocada à provedora de serviço celular. Hoje, a maioria dos sistemas celulares usa uma de duas abordagens gerais para compartilhar o espectro de rádio:

- *Uma combinação de multiplexação por divisão de freqüência (FDM) e multiplexação por divisão de tempo (TDM).* Lembre-se de que, no Capítulo 1, dissemos que, com FDM puro, o canal é dividido em várias bandas de freqüência e cada banda é dedicada a uma chamada. Nesse mesmo capítulo dissemos que, com TDM puro, o tempo é dividido em quadros, cada quadro é subdividido em compartimentos (slots) e cada chamada utiliza um compartimento particular designado quadro rotativo. Em sistemas combinados FDM/TDM o canal é dividido em várias sub-bandas de freqüência; dentro de cada sub-banda o tempo é dividido em quadros e compartimentos. Assim, para um sistema combinado FDM/TDM, se o canal for dividido em F sub-bandas e o tempo for dividido em T compartimentos, então o canal poderá suportar F-T chamadas simultâneas.
- *Acesso múltiplo por divisão de código (CDMA).* Lembre-se de que, na Seção 6.2.1, vimos que o CDMA não divide nem freqüência, nem tempo. Em vez disso, todos os usuários compartilham a mesma radiofreqüência ao mesmo tempo. A cada usuário em uma célula é atribuída uma seqüência distinta de bits, denominada uma seqüência de chipping. Como vimos na Seção 6.2.1, quando o emissor e o receptor usam a mesma seqüência de chipping, o receptor pode recuperar a transmissão do remetente entre as transmissões simultâneas de outros remetentes. Uma importante vantagem do CDMA é que ele elimina a necessidade de alocação de freqüência. Quando é usado um sistema FDM/TDM, os receptores são sensíveis à interferência de outros sinais na mesma banda de freqüência. Assim, uma dada freqüência pode ser reutilizada em um sistema FDM/TDM somente em células localizadas a uma distância suficiente que impeça tal interferência. Essa **reutilização de freqüência** não é uma preocupação importante no projeto de sistemas CDMA.

6.4.2 Padrões e tecnologias celulares: um breve levantamento

Quando alguém fala sobre a tecnologia celular, freqüentemente classifica essa tecnologia como pertencente a uma de diversas 'gerações'. As primeiras gerações foram projetadas primordialmente para tráfego de voz; os sistemas celulares mais recentes suportam acesso à Internet, bem como voz. Como este livro trata de redes de computadores e não de telefonia de voz, é claro que estamos mais interessados nas gerações recentes de sistemas celulares. Mas, como as gerações recentes evoluíram diretamente das gerações anteriores, iniciamos nosso levantamento com uma breve discussão de sistemas sem fio de primeira e de segunda geração.

Durante o levantamento das diversas gerações, faremos uma breve revisão da sopa de letrinhas do jargão das várias tecnologias. Você, leitor, não deve se sentir obrigado a absorver ou memorizar totalmente todos os termos e acrônimos presentes nessa sopa de letrinhas. O propósito desse levantamento é colocar em perspectiva as várias gerações sem fio, bem como prover um guia rápido de referência para os vários termos e acrônimos.

Sistemas de primeira geração (1G) eram análogos aos sistemas FDMA projetados somente para comunicação por voz. Esses sistemas 1G agora estão quase extintos, tendo sido substituídos pelos sistemas digitais 2G.

Segunda geração (2G)

Sistemas de segunda geração, embora digitais, também foram projetados para comunicação por voz. Mas, como os atuais sistemas 2,5G e 3G projetados para manipular comunicação de dados surgiram do 2G, é importante fazer um breve comentário sobre o 2G. Um telefone celular 2G converte um sinal analógico de voz para formato digital antes de modular e então transmitir o sinal pelo ar. A tecnologia digital do 2G tem várias vantagens sobre a tecnologia analógica do 1G, entre elas maior capacidade de serviço dentro de uma célula, melhor segurança para reduzir fraudes e serviços mais avançados, tais como identificação (ID) de chamada e serviço de mensagens. A maioria das provedoras de telefonia celular de hoje usa tecnologia 2G. Há vários padrões e tecnologias 2G amplamente disponibilizados, entre eles:

- *TDMA Interim Standard 136* (***IS-136*** *— Padrão Provisório TDMA IS-136*). É um sistema combinado FDM/TDM que evoluiu da tecnologia FDMA 1G. É amplamente disponibilizado na América do Norte.
- *Global System for Mobile Communications* (***GSM*** *— Sistema Global para Comunicações Móveis*). Na década de 1980, os europeus reconheceram a necessidade de um sistema digital pan-europeu que substituísse seus sistemas 1G incompatíveis, provendo mobilidade ininterrupta e imperceptível entre países, bem como características e capacidades que não eram possíveis com sistemas analógicos. Essa necessidade levou ao padrão GSM para comunicações celulares. Os europeus disponibilizaram tecnologia GSM com grande sucesso no início da década de 1990. Então, o GSM se espalhou para a Ásia e América do Norte e, agora, é o padrão de comunicações celulares mais amplamente disponibilizado. O padrão GSM para sistemas celulares 2G usa FDM/TDM combinados para interface ar. Sistemas GSM consistem em bandas de freqüência de 200 kHz e cada banda suporta oito chamadas TDM. O GSM codifica voz humana a 13 kbps e 12,2 kbps.
- *CDMA* ***IS-95***. Diferentemente do IS-136 e do GSM, que usam FDM/TDM, o CDMA IS-95 usa acesso múltiplo por divisão de código (ver Seção 6.2.1). A empresa Qualcomm demonstrou a viabilidade do CDMA para telefonia celular no final da década de 1980; desde então, foram disponibilizados muitos sistemas IS-95, particularmente na América do Norte e Coréia.

Transição da segunda geração para a terceira geração (2,5G)

Sistemas 2G como IS-95, GSM e IS-136 são otimizados para serviço de voz e não se adaptam particularmente bem à comunicação de dados. Na década de 1990, instituições padronizadoras reconheceram a

necessidade de uma tecnologia celular 3G que fosse apropriada tanto para comunicação de voz como para comunicação de dados (incluindo acesso à Internet). Todavia, como a ampla disponibilização de tecnologia 3G leva muitos anos, as empresas desenvolveram protocolos e padrões provisórios que permitem transmissão de dados pela infra-estrutura 2G existente. Esses sistemas foram apelidados coletivamente de 'sistemas celulares 2,5G'. Entre eles estão:

- ***General Packet Radio Service*** (GPRS — Serviço Geral de Rádio por Pacote). O GPRS evoluiu do GSM. Para serviço de dados, o GSM emula efetivamente um modem entre o equipamento do usuário e a rede de dados destinatária — isto é, o GSM usa comutação de circuitos para seus dados, bem como tráfego de voz. Como aprendemos no Capítulo 1, a comutação de circuitos é altamente ineficiente para rajadas de dados. Além do mais, o padrão GSM suporta taxas de dados somente até 9,6 kbps, o que é intoleravelmente lento para praticamente quase tudo, exceto texto comum. O GPRS é uma solução provisória que provê serviço de dados mais eficiente, por pacotes, a velocidades mais altas (tipicamente na faixa de 40 kbps a 60 kbps). O serviço GPRS é fornecido por uma rede GSM subjacente. Todavia, diferentemente do GSM tradicional, uma estação GPRS móvel pode usar mais de um intervalo de tempo dentro de um dado canal, por demanda. Com o GPRS, uma certa quantidade de intervalos é reservada para comunicação de dados e alocada dinamicamente às estações móveis como uma função de suas demandas instantâneas.
- ***Enhanced Data Rates for Global Evolution*** (*EDGE* — Melhores Taxas de Dados para Evolução Global). O principal objetivo do EDGE é aumentar as capacidades de taxa de dados de uma rede GSM/GPRS, isto é, explorar melhor o canal GSM de 200 kHz com seus quadros TDMA de oito compartimentos. Isso é feito, primordialmente, substituindo o esquema de modulação do GSM por um esquema mais potente. Em teoria, o EDGE pode prover 384 kbps aos usuários para comunicação de dados. Uma revisão excelente do EDGE é apresentada em [Ericsson, 2004].
- ***CDMA 2000, Phase 1*** (CDMA 2000, fase 1). Essa tecnologia 2,5G evoluiu do IS-95 e pode prover serviços de pacote de dados até 144,4 kbps e prepara o terreno para a disponibilização 3G do CDMA 2000, Phase 2.

Terceira geração (3G)

Sistemas celulares 3G devem prover serviço de telefonia bem como de comunicação de dados a velocidades significativamente mais altas do que seus contrapartes 2G. Em particular, sistemas 3G devem prover, obrigatoriamente:

- 144 kbps em velocidades de trânsito.
- 384 kbps para uso estacionário em ambiente externo ou velocidades de quem anda a pé.
- 2 Mbps em ambiente interno.

Há dois importantes padrões competindo na arena do 3G:

- ***Universal Mobile Telecommunications Service*** (***UTMS*** — Serviço Universal de Telecomunicação Móvel). O UMTS é uma evolução do GSM para suportar capacidades 3G. A arquitetura de rede UMTS toma muitas coisas emprestadas da arquitetura de rede GSM já estabelecida. Todavia, o acesso UMTS por rádio é significativamente diferente do esquema FDMA/TDMA usado em GSM. Especificamente, UMTS usa uma técnica CDMA denominada Direct Sequence Wideband CDMA (DS-WCDMA) (CDMA de banda larga de seqüência direta). Visto que o UMTS tem suas raízes no GSM, não é surpresa que ele esteja sendo disponibilizado amplamente na Europa.
- ***CDMA-2000***. CDMA-2000 é uma evolução do sistema IS-95 2G e é reversivelmente compatível com IS-95. Como seu nome dá a entender, ele também usa CDMA como parte de sua interface ar. O CDMA-2000 está sendo disponibilizado na América do Norte e em partes da Ásia.

Quarta geração (4G)

Agora que já examinamos ambas as tecnologias de LAN sem fio e a gama completa das tecnologias celulares de acesso, vamos voltar um pouco e refletir sobre o que nós, como usuários, gostaríamos de ter, idealmente, em matéria de acesso sem fio à Internet. Eis aqui uma lista de desejos que podem vir à nossa mente:

- Gostaríamos que o acesso sem fio à Internet fosse ubíquo. Seja em casa, no escritório, em um carro, em um café ou na praia, gostaríamos de poder acessar a Internet.
- Gostaríamos de poder acessar a Internet à velocidade mais alta possível, como uma função de nossa localização física e da velocidade com a qual estamos nos movendo. Por exemplo, se estivermos em uma esquina onde há acessos disponíveis 802.11b de 11 Mbps e 3G de 384 kbps, gostaríamos que nosso sistema selecionasse automaticamente o sistema 802.11b, que oferece a taxa de bits mais alta naquele instante e naquele lugar.
- Enquanto passeamos por esse ambiente heterogêneo, gostaríamos de ser comutados automática e transparentemente de uma tecnologia de acesso para outra (por exemplo, de 802.11 para 3G), dependendo da disponibilidade, sem nenhuma intervenção do usuário.
- É claro que, enquanto passeamos para lá e para cá, queremos manter nossas conexões TCP em curso. Além do mais, gostaríamos que o sistema soubesse onde estamos de modo que novas chamadas continuem a nos alcançar enquanto nos deslocamos.
- Gostaríamos que o sistema suportasse voz e vídeo em tempo real sobre IP, de modo que todos pudéssemos usar relógios do Dick Tracy e fazer videoconferência com nossos amigos e colegas, não importando onde estivéssemos.

E, é claro, gostaríamos que isso fosse gratuito (ou, mais realisticamente, ao menos a baixo custo). Embora tudo isso talvez pareça um nirvana sem fio, a boa notícia é que grande parte dos componentes dessa tecnologia crítica já está disponível. O que realmente falta agora é integrar protocolos e tecnologias para transformar essa lista de desejos em realidade. Entre esses componentes estão tecnologias de acesso como 802.11 e 3G, bem como protocolos de gerenciamento de mobilidade (veja seções 6.5, 6.6 e 6.7), criptografia e protocolos de autenticação (veja Capítulo 8) e protocolos de redes multimídia (tal como SIP para voz sobre IP, que será discutido no Capítulo 7).

6.5 Gerenciamento da mobilidade: princípios

Agora que já estudamos a natureza *sem fio* dos enlaces de comunicação em uma rede sem fio, é hora de voltarmos nossa atenção à *mobilidade* que esses enlaces sem fio possibilitam. No sentido mais amplo, um nó móvel é aquele que muda seu ponto de conexão com a rede ao longo do tempo. Como o termo *mobilidade* adquiriu muitos significados no mundo da computação e também no mundo da telefonia, será proveitoso, antes de mais nada, considerar diversas dimensões da mobilidade com um certo detalhe.

- *Do ponto de vista da camada de rede, até que ponto um usuário é móvel?* Um usuário fisicamente móvel apresentará à camada de rede conjuntos de desafios muito diferentes dependendo de como ele se movimenta entre pontos de conexão com a rede. Em uma extremidade do espectro na Figura 6.16, um usuário pode carregar consigo um laptop equipado com uma placa de interface de rede sem fio dentro de um edifício. Como vimos na Seção 6.3.4, esse usuário *não* é móvel da perspectiva da camada de rede. Além do mais, se o usuário se associar com o mesmo ponto de acesso independentemente de localização, então ele não será móvel nem mesmo do ponto de vista da camada de enlace.

  Na outra extremidade do espectro, considere o usuário que está dentro de um BMW, correndo pela estrada a 150 quilômetros por hora, passando por várias redes de acesso sem fio e querendo manter uma conexão TCP ininterrupta com uma aplicação remota durante a viagem. Esse usuário é

definitivamente móvel! Entre esses extremos está um usuário que leva seu laptop de uma localização (por exemplo, escritório ou quarto de dormir) a outra (por exemplo, lanchonete, biblioteca) e quer se conectar à rede na nova localização. Esse usuário também é móvel (embora menos do que o motorista do BMW!), mas não precisa manter uma conexão ativa enquanto se movimenta entre pontos de conexão com a rede. A Figura 6.16 ilustra esse espectro de mobilidade do usuário da perspectiva da camada de rede.

Qual é a importância de o endereço do nó móvel permanecer sempre o mesmo? Com telefonia móvel, o número de seu telefone — essencialmente o endereço de camada de rede do seu aparelho — permanece o mesmo quando você transita entre uma provedora de rede de telefonia móvel e outra. Um laptop também terá de manter o mesmo endereço IP enquanto se movimenta entre redes IP?

A resposta a essa pergunta dependerá muito da aplicação que está sendo executada. Para o motorista da BMW que quer manter uma conexão TCP ininterrupta com uma aplicação remota enquanto voa pela estrada, seria conveniente manter o mesmo endereço IP. Lembre-se de que dissemos, no Capítulo 3, que uma aplicação de Internet precisa conhecer o endereço IP e o número de porta da entidade remota com a qual está se comunicando. Se uma entidade móvel puder manter seu endereço IP enquanto estiver em trânsito, a mobilidade torna-se invisível do ponto de vista da aplicação. Essa transparência é de grande valor — uma aplicação não precisa se preocupar com uma potencial mudança de endereço IP e o mesmo código de aplicação servirá igualmente a conexões móveis e não móveis. Na próxima seção veremos que o IP móvel provê essa transparência, permitindo que um nó móvel mantenha seu endereço IP permanente enquanto se movimenta entre redes.

Por outro lado, um usuário móvel menos sofisticado poderia querer simplesmente desligar seu laptop no escritório, levá-lo para casa, ligá-lo novamente e continuar trabalhando em casa. Se o laptop funciona primordialmente como um cliente em aplicações cliente-servidor (por exemplo, enviar/receber e-mails, navegar pela Web, usar Telnet com um hospedeiro remoto), o endereço IP particular utilizado pelo laptop não é muito importante. Em particular, é possível passar muito bem com um endereço temporariamente alocado ao laptop pelo ISP ao qual a residência está ligada. Na Seção 5.4.3 vimos que o DHCP já provê essa funcionalidade.

Qual é a infra-estrutura cabeada de suporte disponível? Em todos os quatro cenários descritos, admitimos implicitamente que existe uma infra-estrutura fixa à qual o usuário móvel pode se conectar, por exemplo, a rede ISP à qual a residência está ligada, a rede de acesso sem fio no local de trabalho ou as redes de acesso sem fio ao longo da rodovia. E se tal infra-estrutura não existir? Se dois usuários estiverem a uma distância que permita comunicação, eles poderão estabelecer uma conexão de rede na ausência de qualquer outra infra-estrutura de camada de rede? As redes ad hoc provêem exatamente essas capacidades. Essa área, que está em rápido desenvolvimento e representa a pesquisa de ponta em redes móveis, está além do escopo deste livro. [Perkins, 2000] e as páginas Web do grupo de trabalho Mobile Ad Hoc Network (manet) da IETF apresentam um tratamento completo do assunto.

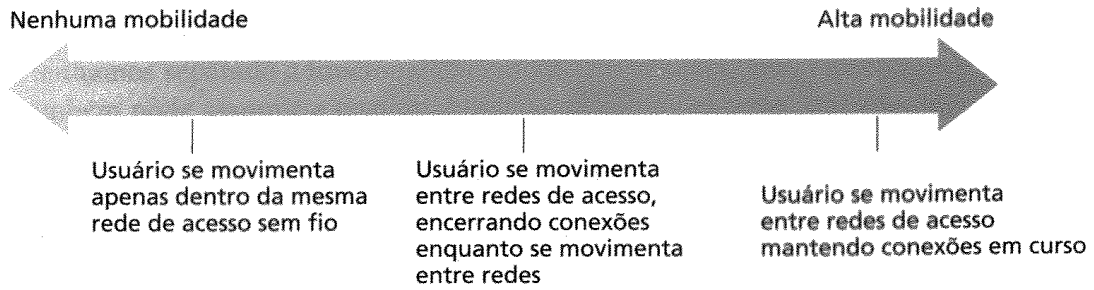


Figura 6.16 Vários graus de mobilidade do ponto de vista da camada de rede

Para ilustrar as questões que envolvem permitir que um usuário mantenha conexões em curso enquanto se movimenta entre redes, vamos considerar uma analogia com seres humanos. Um adulto de 20 e poucos anos que sai da casa de seus pais torna-se móvel, pois passa a morar em uma série de quartos e/ou apartamentos e está sempre mudando de endereço. Se uma velha amiga quiser entrar em contato com ele, como conseguirá encontrar o endereço de seu amigo móvel? Uma maneira comum de fazer isso é entrar em contato com a família, já que um jovem móvel costuma informar seus novos endereços à família (nem que seja apenas para que seus pais possam lhe enviar dinheiro para ajudar a pagar o aluguel!). A residência da família, com seu endereço permanente, torna-se aquele único lugar a que outros podem se dirigir como uma primeira etapa para estabelecer comunicação com o jovem móvel. As comunicações posteriores da velha amiga podem ser indiretas (por exemplo, pelo envio de uma carta primeiramente à casa dos pais, que então encaminharão a carta ao jovem móvel) ou diretas (por exemplo, a velha amiga utiliza o endereço informado pelos pais para enviar uma carta diretamente a seu amigo móvel).

Em um ambiente de rede, a residência permanente de um nó móvel (tal como um laptop ou um PDA) é conhecida como **rede nativa** (home network) e a entidade dentro dessa rede que executa o gerenciamento de funções de mobilidade em nome do nó móvel (e que serão discutidas mais adiante) é conhecida como **agente nativo** (home agent). A rede na qual o nó móvel está residindo atualmente é conhecida como **rede externa** (foreign network) ou **rede visitada** (visited network), e a entidade dentro da rede externa que auxilia o nó móvel no gerenciamento das funções de mobilidade discutidas adiante é conhecida como **agente externo** (foreign agent). No caso de profissionais móveis, suas redes nativas possivelmente seriam as redes das empresas em que trabalham, enquanto a rede visitada poderia ser a rede de um colega que estão visitando. Um **correspondente** é a entidade que quer se comunicar com o nó móvel. A Figura 6.17 ilustra esses conceitos, bem como conceitos de endereçamento considerados mais adiante. Observe que, na Figura 6.17, os agentes são colocados junto aos roteadores (por exemplo, como processos que funcionam em roteadores), mas, alternativamente, poderiam estar funcionando em outros hospedeiros ou servidores na rede.

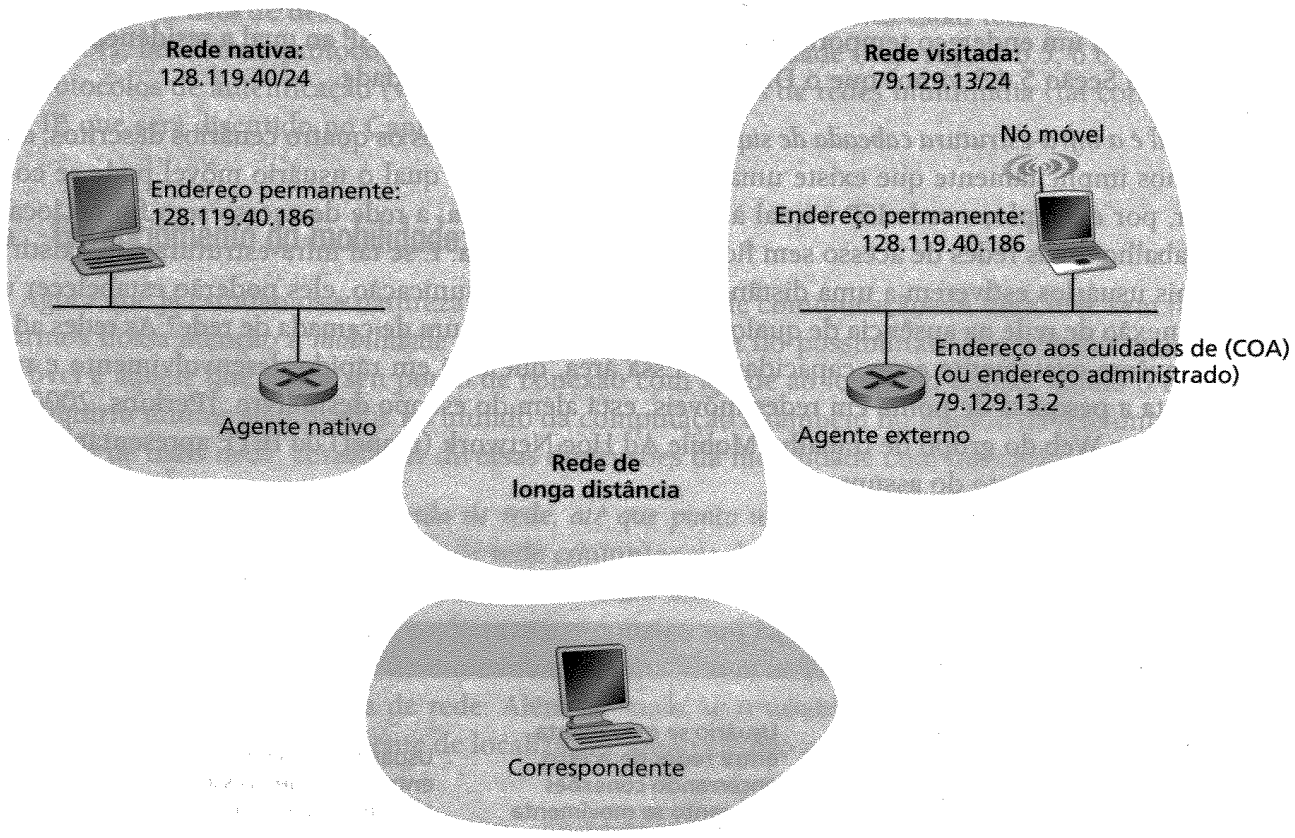


Figura 6.17 Elementos iniciais de uma arquitetura de rede móvel

6.5.1 Endereçamento

Já observamos que, para que a mobilidade do usuário seja transparente para aplicações de rede, é desejável que um nó móvel mantenha seu endereço quando transita de uma rede para outra. Quando um nó móvel residir em uma rede externa, todo o tráfego enviado ao endereço permanente do nó agora precisará ser roteado para a rede externa. Como isso pode ser feito? Uma opção é a rede externa anunciar para todas as outras redes que o nó móvel agora reside em sua rede, o que poderia ser feito mediante a costumeira troca de informações de roteamento interdomínios e intradomínios e exigiria poucas mudanças na infra-estrutura de roteamento existente. A rede externa poderia simplesmente anunciar a seus vizinhos que tem uma rota altamente específica para o endereço permanente do nó móvel (isto é, essencialmente, informar a outras redes que ela tem o caminho correto para rotear datagramas para o endereço permanente do nó móvel; ver Seção 4.4). Esses vizinhos então propagariam essa informação de roteamento por toda a rede como parte do procedimento normal de atualização de informações de roteamento e tabelas de repasse. Quando o nó móvel sair de uma rede externa e se juntar a uma outra, a nova rede externa anunciaria uma nova rota, altamente específica, até o nó móvel e a antiga rede externa retiraria suas informações referentes ao nó móvel.

Esse procedimento resolve dois problemas de uma vez só e o faz sem promover mudanças significativas na infra-estrutura da camada de rede. Outras redes conhecem a localização do nó móvel e é fácil rotear datagramas para o nó móvel, visto que as tabelas de repasse dirigirão datagramas à rede externa. Uma desvantagem significativa, contudo, é a da escalabilidade. Se a responsabilidade pelo gerenciamento da mobilidade tivesse de recair sobre os roteadores da rede, eles teriam de manter registros em tabelas de repasse para potencialmente milhões de nós móveis e atualizar esses registros à medida que os nós se movimentassem. Algumas desvantagens adicionais serão exploradas nos problemas ao final deste capítulo.

Uma abordagem alternativa (que tem sido adotada na prática) é passar a funcionalidade de mobilidade do núcleo da rede para a borda da rede — um tema recorrente em nosso estudo da arquitetura da Internet. Um modo natural de fazer isso é por meio da rede nativa do nó móvel. De maneira muito semelhante ao modo como os pais daquele jovem de 20 e poucos anos monitoram a localização de seu filho, o agente nativo na rede nativa do nó móvel pode monitorar a rede externa na qual o nó móvel reside. Certamente será preciso um protocolo (ou um agente externo representando o nó móvel) entre o nó móvel e o agente nativo para atualizar a localização do nó móvel.

Agora vamos considerar o agente externo com mais detalhes. A abordagem conceitualmente mais simples, mostrada na Figura 6.17, é localizar agentes externos nos roteadores de borda na rede externa. Um dos papéis do agente externo é criar o denominado **endereço aos cuidados de** (*care-of-address* — **COA**) **ou endereço administrado** para o nó móvel, sendo que a parte da rede do endereço COA combinaria com a parte da rede do endereço da rede externa. Assim, há dois endereços associados com um nó móvel, seu **endereço permanente** (análogo ao endereço da família do nosso jovem móvel) e seu endereço COA, às vezes denominado **endereço externo** (análogo ao endereço da casa onde nosso jovem móvel está residindo atualmente). No exemplo da Figura 6.17, o endereço permanente do nó móvel é 128.119.40.186. Quando está visitando a rede 79.129.13/24, o nó móvel tem um COA 79.129.13.2. Um segundo papel desempenhado pelo agente externo é informar ao agente nativo que o nó móvel está residindo em sua rede (do agente externo) e tem o dado endereço COA. Veremos, em breve, que o COA pode ser utilizado para 'rerrotear' datagramas para o nó móvel via seu agente externo.

Embora tenhamos separado a funcionalidade do nó móvel e o agente externo, vale a pena observar que o nó móvel também pode assumir a responsabilidade do agente externo. Por exemplo, o nó móvel poderia obter um COA na rede externa (por exemplo, utilizando um protocolo como o DHCP) e ele mesmo informar seu COA ao agente nativo.

6.5.2 Roteamento para um nó móvel

Agora já vimos como um nó móvel obtém um COA e como é possível informar esse endereço ao agente nativo. Mas conseguir que o agente nativo conheça o COA resolve apenas parte do problema. Como datagramas devem ser endereçados e repassados para o nó móvel? Visto que somente o agente nativo (e não os roteadores no âmbito total da rede) conhece a localização do nó móvel, já não será mais suficiente simplesmente endereçar um datagrama para o endereço permanente do nó móvel e enviá-lo para a infra-estrutura de camada de rede. É preciso fazer algo mais. Duas abordagens podem ser identificadas, as quais denominaremos roteamento indireto e roteamento direto.

Roteamento indireto para um nó móvel

Vamos considerar primeiramente um correspondente que quer enviar um datagrama a um nó móvel. Na abordagem de **roteamento indireto** o correspondente simplesmente endereça o datagrama ao endereço permanente do nó móvel, envia o datagrama para a rede e fica alegremente inconsciente quanto ao nó móvel residir em sua rede nativa ou estar visitando uma rede externa; assim, a mobilidade é completamente transparente para o correspondente. Esses datagramas são primeiramente roteados, como sempre, para a rede local do nó móvel. Isso é ilustrado na etapa 1 da Figura 6.18.

Agora vamos voltar nossa atenção ao agente nativo. Além de ser responsável por interagir com um agente externo para monitorar o COA do nó móvel, o agente nativo tem uma outra função muito importante. Sua segunda tarefa é ficar à espreita de datagramas que chegam e são endereçados a nós cuja rede nativa é a rede do agente nativo, mas que estão correntemente residindo em uma rede externa. O agente nativo intercepta esses datagramas e então os repassa a um nó móvel por um processo de duas etapas. Primeiramente o datagrama é repassado para o agente externo usando o COA do nó móvel (etapa 2 na Figura 6.18) e então é repassado do agente externo para o nó móvel (etapa 3 na Figura 6.18).

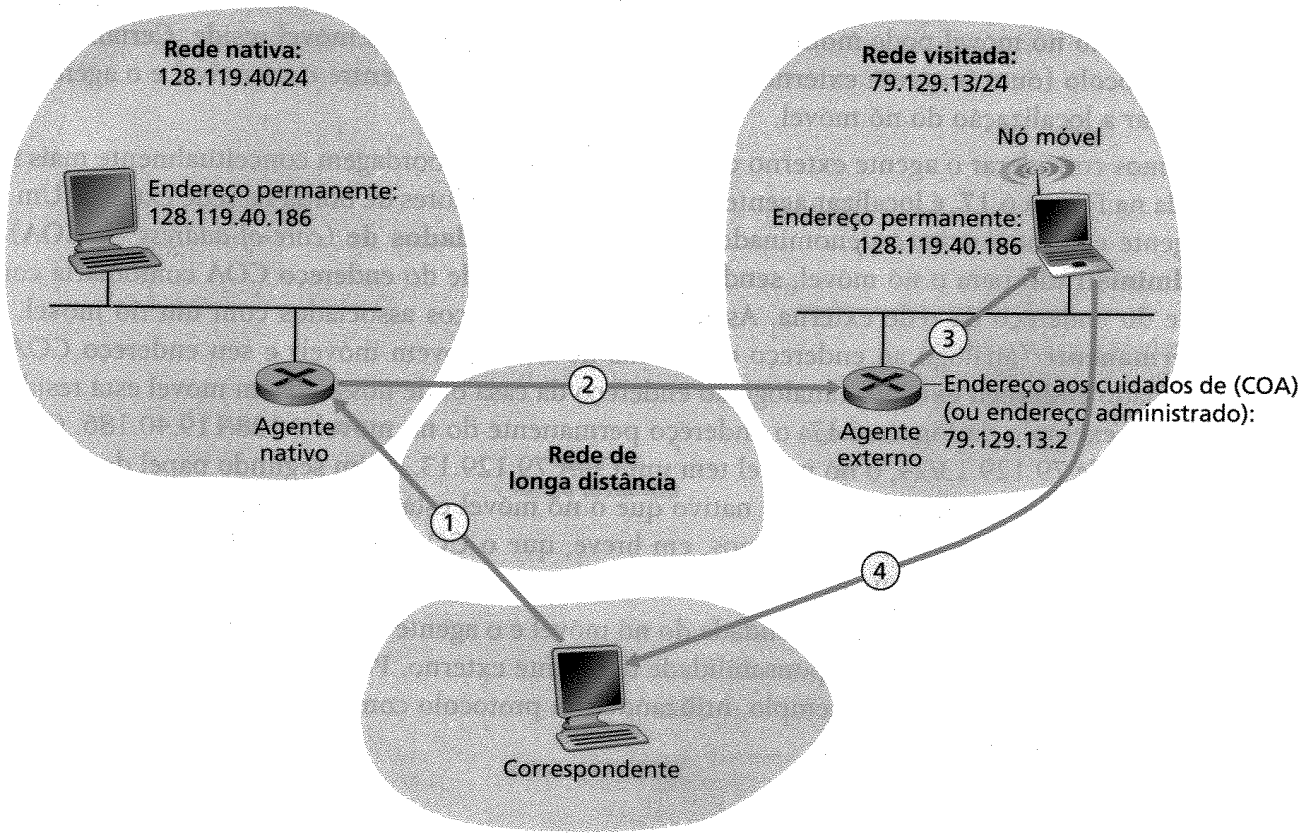


Figura 6.18 Repasse indireto para um nó móvel

É instrutivo considerar esse rerroteamento com mais detalhes. O agente nativo precisará endereçar o datagrama usando o COA do nó móvel, de modo que a camada de rede roteará o datagrama para a rede externa. Por outro lado, é desejável deixar intacto o datagrama do correspondente, visto que a aplicação que recebe o datagrama deve ficar inconsciente de que o datagrama foi repassado por meio do agente nativo. Ambas as metas podem ser cumpridas fazendo com que o agente nativo **encapsule** o datagrama original completo do correspondente dentro de um novo (e maior) datagrama. Esse datagrama maior é endereçado e entregue ao COA do nó móvel. O agente externo 'proprietário' do COA receberá e desencapsulará o datagrama, isto é, removerá o datagrama original do correspondente de dentro daquele datagrama maior de encapsulamento e repassará o datagrama original (etapa 3 na Figura 6.18) para o nó móvel. A Figura 6.19 mostra um datagrama original de um correspondente sendo enviado para a rede local, um datagrama encapsulado sendo enviado ao agente externo e o datagrama original sendo entregue ao nó móvel. O leitor observador notará que o encapsulamento/desencapsulamento descrito aqui é idêntico à noção de implementação de túnel discutida no Capítulo 4 no contexto do IP *multicast* e do IPv6.

A seguir, vamos considerar como um nó móvel envia datagramas a um correspondente. Isso é muito simples, pois o nó móvel pode endereçar seu datagrama *diretamente* ao correspondente (usando seu próprio endereço permanente como o endereço de origem e o endereço do correspondente como o endereço de destino). Visto que o nó móvel conhece o endereço do correspondente, não há necessidade de rotear o datagrama de volta por meio do agente nativo. Isso é mostrado como etapa 4 na Figura 6.18.

Vamos resumir o que discutimos sobre roteamento indireto relacionando as novas funcionalidades de camada de rede exigidas para suportar mobilidade.

- *Um protocolo nó móvel–agente externo.* O nó móvel fará seu registro junto ao agente externo ao se conectar à rede externa. De modo semelhante, um nó móvel cancelará seu registro junto ao agente externo quando sair da rede externa.
- *Um protocolo de registro agente externo–agente nativo.* O agente externo registrará o COA do nó móvel junto ao agente nativo. Um agente externo não precisa cancelar explicitamente um COA quando um nó móvel sai da rede porque o registro subseqüente de um novo COA se encarregará disso quando o nó móvel passar para uma nova rede.

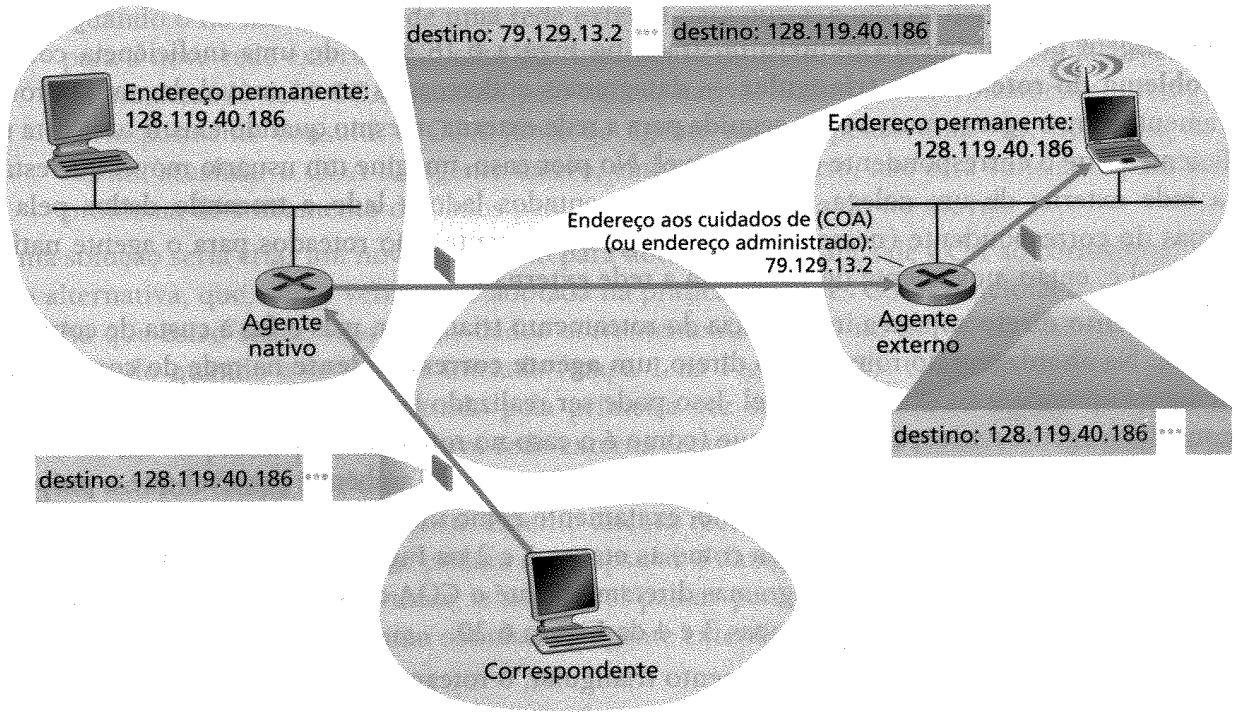


Figura 6.19 Encapsulamento e desencapsulamento

- *Um protocolo de encapsulamento de datagrama para o agente nativo.* Encapsulamento e repasse do datagrama original do correspondente dentro de um datagrama endereçado ao COA.
- *Um protocolo de desencapsulamento de datagrama para o agente externo.* Extração do datagrama original do correspondente de dentro do datagrama de encapsulamento e repasse do datagrama original ao nó móvel.

A discussão anterior provê todas as peças — agentes externos, agente nativo e roteamento indireto — de que um nó móvel necessita para manter uma conexão em curso enquanto transita entre redes. Como um exemplo de como essas peças se encaixam, admita que o nó móvel está ligado à rede externa A, registrou um COA junto ao agente local na rede A e está recebendo datagramas que estão sendo roteados indiretamente por meio de seu agente nativo. O nó móvel agora passa para a rede externa B e se registra junto ao agente externo na rede B, que informa ao agente nativo o novo COA do nó móvel. Desse ponto em diante, o agente nativo rerroteará datagramas para a rede externa B. No que diz respeito ao correspondente, a mobilidade é transparente — datagramas são roteados por meio do mesmo agente nativo antes e depois de o nó móvel mudar de rede. No que diz respeito ao agente nativo, não há nenhuma disrupção no fluxo de datagramas — datagramas que chegam são primeiramente repassados para a rede externa A; após a mudança no COA, datagramas são repassados para a rede externa B. Mas o nó móvel verá um fluxo ininterrupto de datagramas ao se movimentar entre redes? Contanto que seja pequeno o tempo transcorrido entre o desligamento do nó móvel da rede A (ponto em que ele não pode mais receber datagramas via A) e sua conexão com a rede B (ponto em que registrará um novo COA junto ao agente nativo da rede), poucos datagramas serão perdidos. Lembre-se de que dissemos, no Capítulo 3, que conexões fim-a-fim podem sofrer perda de datagramas devido a congestionamento na rede. Por conseguinte, a perda ocasional de datagramas dentro de uma conexão quando um nó se move entre redes não é, de maneira alguma, um problema catastrófico. Se for preciso uma comunicação livre de perdas, mecanismos de camadas superiores recuperarão a perda de dados, quer essas perdas resultem do congestionamento da rede ou da mobilidade do usuário.

Uma abordagem indireta de roteamento é utilizada no padrão IP móvel [RFC 3220], como discutiremos na Seção 6.6.

Roteamento direto para um nó móvel

A abordagem do roteamento indireto ilustrada na Figura 6.18 sofre de uma ineficiência conhecida como **problema do roteamento triangular** — datagramas endereçados ao nó móvel devem ser roteados primeiramente para o agente nativo e em seguida para a rede externa, mesmo quando existir uma rota muito mais eficiente entre o correspondente e o nó móvel. No pior caso, imagine um usuário móvel que está visitando a rede externa de um colega. Os dois estão sentados lado a lado e trocando dados pela rede. Datagramas do correspondente (nesse caso, do colega do visitante) são roteados para o agente nativo do usuário móvel e, então, novamente de volta para a rede externa!

O **roteamento direto** supera a ineficiência do roteamento triangular, mas o faz à custa de complexidade adicional. Na abordagem do roteamento direto, um **agente correspondente** na rede do correspondente primeiramente aprende o COA do nó móvel. Isso pode ser realizado fazendo com que o agente correspondente consulte o agente nativo, admitindo que (como é o caso no roteamento indireto) o nó móvel tem um valor atualizado para seu COA registrado junto a seu agente nativo. Também é possível que o próprio correspondente execute a função do agente externo, exatamente como um nó móvel poderia executar a função do agente externo. Essa situação é ilustrada como as etapas 1 e 2 na Figura 6.20. O agente correspondente então implementa um túnel para os datagramas diretamente até o COA do nó móvel, de um modo análogo ao túnel executado pelo agente nativo, etapas 3 e 4 da Figura 6.20.

Conquanto supere o problema do roteamento triangular, o roteamento direto introduz dois importantes desafios adicionais:

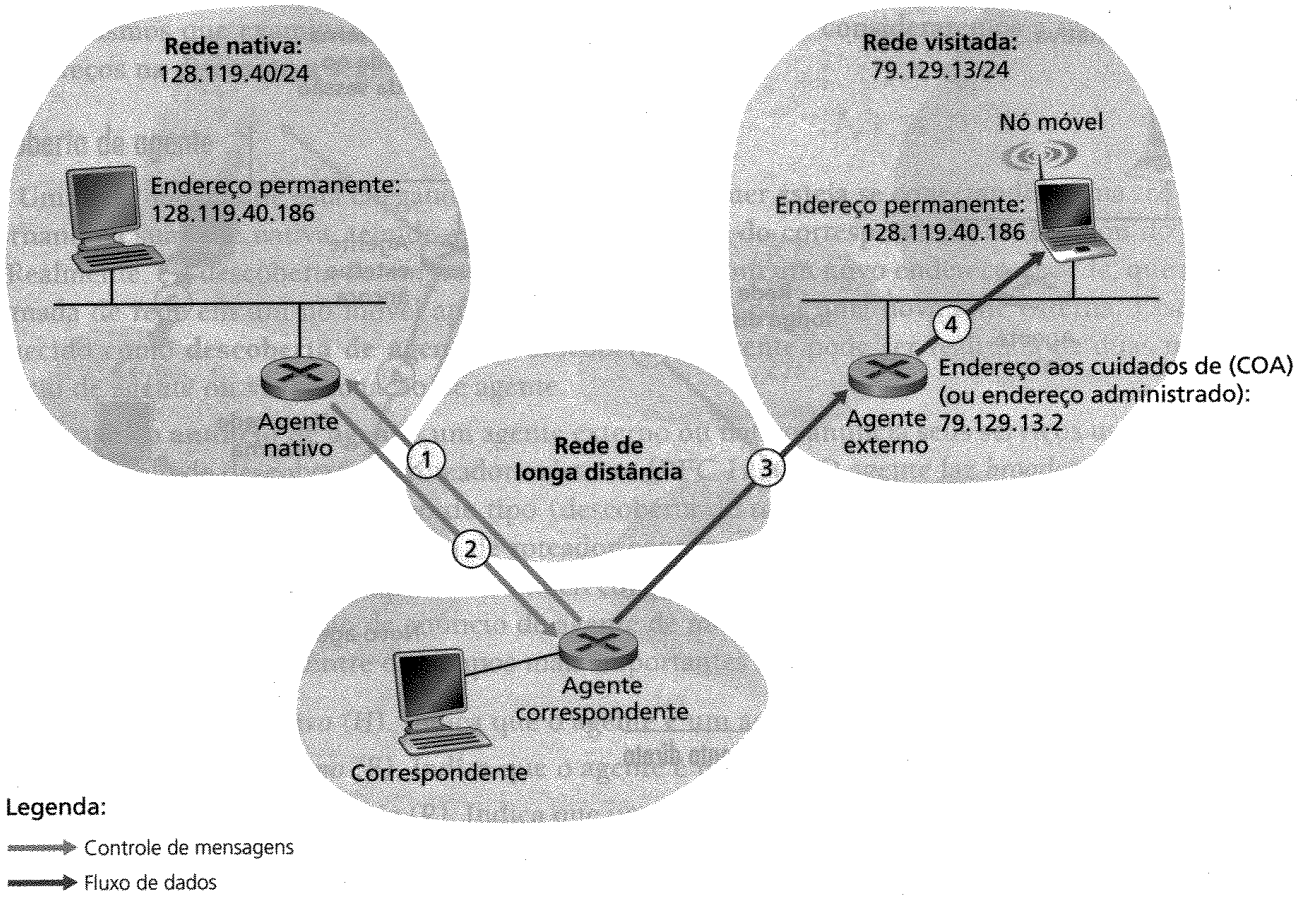


Figura 6.20 Roteamento direto para um usuário móvel

É preciso um **protocolo de localização de usuário móvel** para o agente correspondente consultar o agente nativo de modo a obter o COA do nó móvel (etapas 1 e 2 da Figura 6.20).

Quando o nó móvel passa de uma rede externa para outra, como os dados são repassados, agora, para a nova rede externa? No caso do roteamento indireto, esse problema era facilmente resolvido atualizando-se o COA mantido pelo agente nativo. Todavia, com roteamento direto, o agente correspondente consulta o COA junto ao agente nativo apenas uma vez, no início da sessão. Assim, atualizar o COA do agente nativo, embora necessário, não será suficiente para resolver o problema do roteamento de dados para a nova rede externa do nó móvel.

Uma solução seria criar um novo protocolo para notificar a mudança de COA ao correspondente. Uma solução alternativa, que, como veremos, é adotada na prática em redes GSM, funciona da seguinte maneira: suponha que estão sendo repassados dados correntemente para o nó móvel na rede externa onde ele estava localizado quando a sessão foi iniciada (etapa 1 na Figura 6.21). O agente externo naquela rede externa onde o nó móvel foi encontrado pela primeira vez será denominado **agente externo âncora**. Quando o nó móvel passar para uma nova rede externa (etapa 2 na Figura 6.21), ele se registrará junto ao novo agente externo (etapa 3) e esse novo agente externo fornecerá ao agente externo âncora o novo COA do nó móvel (etapa 4). Quando o agente externo âncora receber um datagrama encapsulado para um nó móvel que já deixou a rede, ele então poderá reencapsular o datagrama e repassá-lo para o nó móvel (etapa 5) usando o novo COA. Se, mais tarde, o nó móvel passar para mais uma outra rede externa, o agente externo nessa nova rede visitada então contatará o agente externo âncora para estabelecer repasse para essa nova rede externa.

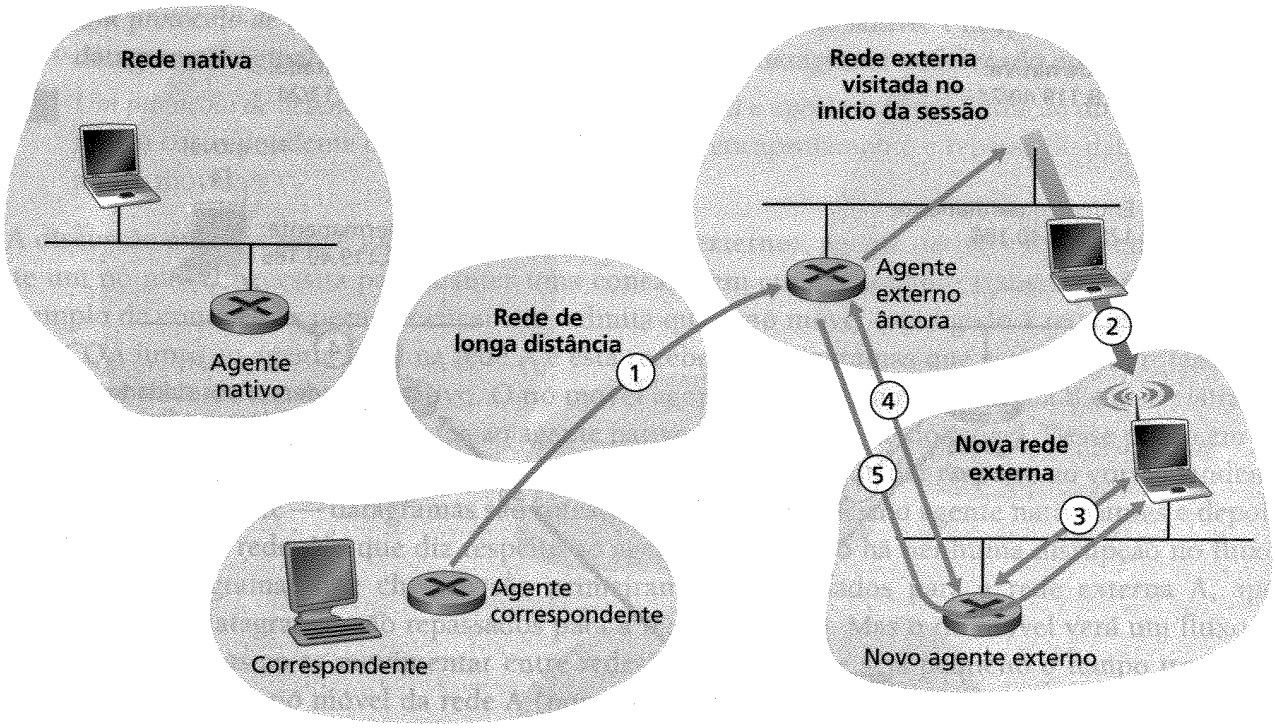


Figura 6.21 Transferência móvel entre redes com roteamento direto

6.6 IP móvel

A arquitetura e os protocolos da Internet para suporte de mobilidade, conhecidos coletivamente como IP móvel, estão definidos primariamente no RFC 3220. O IP móvel é um protocolo flexível que suporta muitos modos de operação diferentes, por exemplo, operação com ou sem um agente externo, várias maneiras para agentes e nós móveis descobrirem uns aos outros, utilização de um único COA ou de vários COAs e diversas formas de encapsulamento. Por isso, o IP móvel é um protocolo complexo cuja descrição detalhada exigiria um livro inteiro; um desses livros é [Perkins, 1998b]. Aqui, nossa modesta meta é prover uma visão geral dos aspectos mais importantes do IP móvel e ilustrar sua utilização em alguns cenários comuns.

A arquitetura do IP móvel contém muitos dos elementos que acabamos de considerar, incluindo os conceitos de agentes nativos, agentes externos, endereços administrados e encapsulamento/desencapsulamento. O padrão corrente [RFC 3220] especifica a utilização de roteamento indireto para o nó móvel.

O padrão IP móvel consiste em três partes principais:

- *Descoberta de agente.* O IP móvel define os protocolos utilizados por um agente nativo ou por um agente externo para anunciar seus serviços a nós móveis e protocolos para que os nós móveis solicitem os serviços de um agente externo ou nativo.
- *Registro junto ao agente nativo.* O IP móvel define os protocolos usados pelo nó móvel e/ou agente externo para registrar e anular os registros de COAs junto ao agente local de um nó móvel.
- *Roteamento indireto de datagramas.* O padrão também define a maneira pela qual datagramas são repassados para nós móveis por um agente nativo, incluindo regras para repassar datagramas, regras para manipular condições de erro e diversas formas de encapsulamento [RFC 2003, RFC 2004].

Considerações de segurança têm destaque em todo o padrão IP móvel. Por exemplo, a autenticação de um nó móvel é claramente necessária para impedir que um usuário mal-intencionado registre junto ao agente nativo um falso endereço administrado, o que poderia fazer com que datagramas endereçados a um ende-

reço IP sejam direcionados ao usuário mal-intencionado. O IP móvel consegue segurança usando muitos dos mecanismos que examinaremos no Capítulo 8, portanto, não consideraremos a questão da segurança de endereços na discussão a seguir.

Descoberta de agente

Um nó IP móvel que está chegando a uma nova rede, quer esteja se conectando a uma rede externa ou retornando à sua rede nativa, tem de aprender a identidade do correspondente externo ou do agente nativo. Realmente, é a descoberta de um novo agente externo, com um novo endereço de rede, que permite que a camada de rede em um nó móvel aprenda que ele passou para uma nova rede externa. Esse processo é conhecido como **descoberta de agente**. A descoberta de agente pode ser realizada de duas maneiras: via anúncio de agente ou via solicitação de agente.

No caso do **anúncio de agente**, um agente externo ou nativo anuncia seus serviços usando uma extensão do protocolo de descoberta de roteador existente [RFC 1256]. O agente faz *broadcast* periódico de uma mensagem ICMP tendo 9 no campo de tipo (descoberta de roteador) em todos os enlaces aos quais está conectado. A mensagem de descoberta de roteador contém o endereço IP do roteador (isto é, o agente), o que permite que um nó móvel aprenda o endereço IP do agente. A mensagem de descoberta de roteador também contém uma extensão de anúncio de agente de mobilidade que contém informações adicionais de que o nó móvel necessita. Entre os campos mais importantes na extensão estão os seguintes:

- *Bit do agente nativo (H).* Indica que o agente é um agente nativo para a rede na qual reside.
- *Bit de agente externo (F).* Indica que o agente é um agente externo para a rede na qual reside.
- *Bit de registro obrigatório (R).* Indica que um usuário móvel nessa rede *deve* se registrar junto a um agente externo. Em particular, um usuário móvel não pode obter um endereço administrado na rede externa (por exemplo, usando DHCP) e assumir a funcionalidade de agente externo para si mesmo sem se registrar junto ao agente externo.
- *Bits de encapsulamento M, G.* Indicam se será utilizada uma forma de encapsulamento que não seja o encapsulamento IP em IP.
- *Campos de endereços administrados (COA).* Uma lista de um ou mais endereços administrados fornecida pelo agente externo. No exemplo logo a seguir, o COA estará associado com o agente externo, que receberá datagramas enviados ao COA e então os repassará para o nó móvel adequado. O usuário móvel selecionará um desses endereços como seu COA ao se registrar junto a seu agente nativo.

A Figura 6.22 ilustra alguns dos principais campos na mensagem de anúncio de agente.

Com **solicitação de agente**, um nó móvel que quer conhecer agentes sem esperar para receber um anúncio de agente pode emitir uma mensagem de solicitação de agente em *broadcast*, que é, simplesmente, uma mensagem ICMP cujo valor de tipo é 10. Ao receber a solicitação, um agente emite um anúncio *unicast* de agente diretamente ao nó móvel, que, então, procederá como se tivesse recebido um anúncio não solicitado.

Registro junto ao agente nativo

Tão logo um nó móvel IP tenha recebido um COA, esse endereço deve ser registrado junto ao agente nativo, o que pode ser feito via o agente externo (que, então, registra o COA junto ao agente nativo) ou diretamente pelo próprio nó móvel IP. A seguir, consideramos o primeiro caso. Há quatro etapas envolvidas.

1. Após o recebimento de um anúncio de agente externo, um nó móvel envia uma mensagem de registro IP móvel ao agente externo. A mensagem de registro é carregada dentro de um datagrama UDP e enviada à porta 434. A mensagem leva um COA anunciado pelo agente externo, o endereço do agente nativo (HA), o endereço permanente do nó móvel (MA), o tempo de vida de registro requerido e uma identificação de registro de 64 bits. O tempo de vida de registro requerido é o

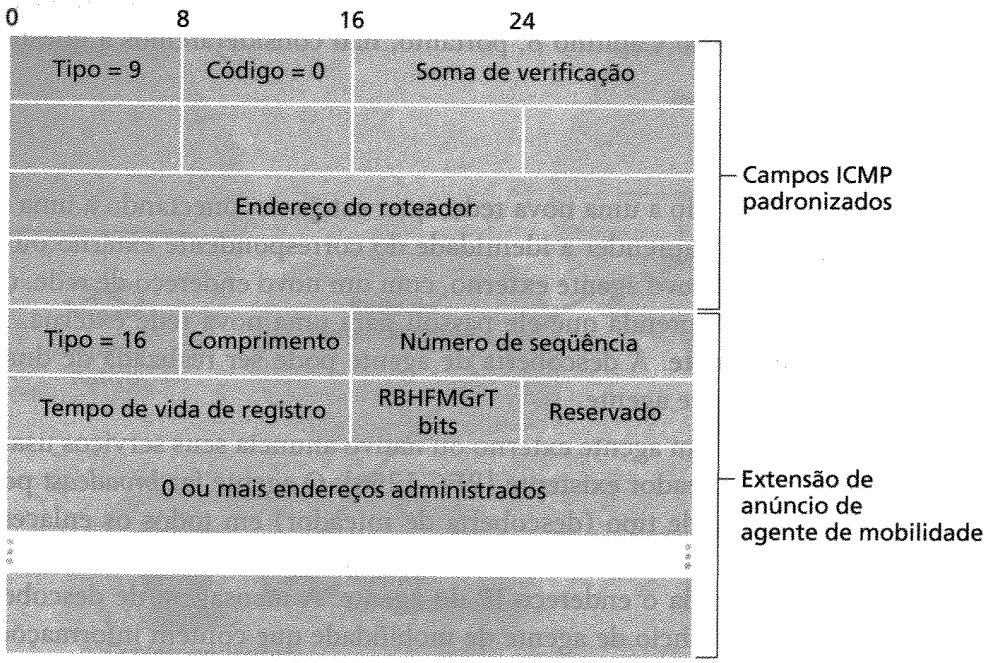


Figura 6.22 Mensagem ICMP de descoberta de roteador com extensão de anúncio de agente de mobilidade

número de segundos durante os quais o registro será válido. Se o registro não for renovado no agente nativo dentro do tempo de vida especificado, ele se tornará inválido. O identificador de registro age como um número de seqüência e serve para combinar uma resposta de registro recebida com uma solicitação de registro, como discutiremos mais adiante.

2. O agente externo recebe a mensagem de registro e registra o endereço IP permanente do nó móvel. Agora o agente externo sabe que deve procurar datagramas que contenham um datagrama encapsulado cujo endereço de destino combine com o endereço permanente do nó móvel. O agente externo então envia uma mensagem de registro IP móvel (novamente dentro de um datagrama UDP) à porta 434 do agente nativo. A mensagem contém o COA, o HA, o MA, o formato de encapsulamento requisitado, o tempo de vida de registro requisitado e a identificação do registro.
3. O agente nativo recebe a requisição de registro e verifica sua autenticidade e correção. O agente nativo vincula o endereço IP permanente do nó móvel ao COA; no futuro, datagramas que chegarem ao agente nativo e endereçados ao nó móvel serão encapsulados e enviados por túnel até o COA. O agente nativo envia uma resposta de registro IP móvel contendo o HA, o MA, o tempo de vida de registro vigente e a identificação de registro da solicitação que está sendo atendida com essa resposta.
4. O agente externo recebe a resposta de registro e então a repassa ao nó móvel.

Nesse ponto o registro está concluído e o nó móvel pode receber datagramas enviados a seu endereço permanente. A Figura 6.23 ilustra essas etapas. Note que o agente nativo especifica um tempo de vida menor do que o tempo de vida requisitado pelo nó móvel.

Um agente externo não precisa anular explicitamente um registro de COA quando um nó móvel sai da rede. Isso ocorrerá automaticamente quando o nó móvel passar para uma nova rede (seja uma outra rede externa, seja sua rede nativa) e registrar um novo COA.

O padrão IP móvel permite muitos cenários e capacidades adicionais além das que acabamos de descrever. O leitor interessado deve consultar [Perkins, 1998b; RFC 3220].

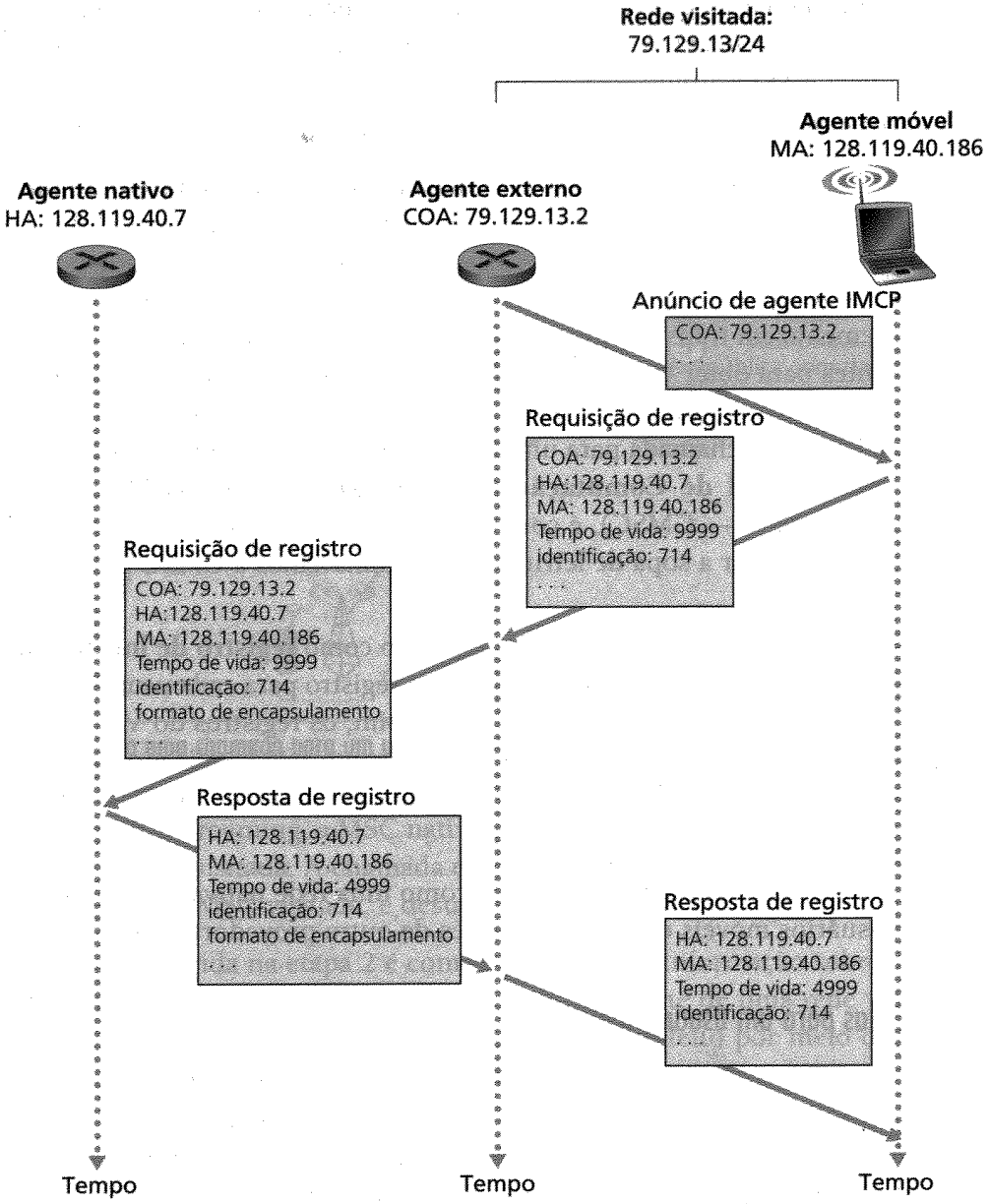


Figura 6.23 Anúncio de agente e registro IP móvel

6.7 Gerenciamento de mobilidade em redes celulares

Agora que acabamos de examinar como a mobilidade é gerenciada em redes IP, vamos voltar nossa atenção a redes cujo histórico de suporte à mobilidade é ainda mais longo — redes de telefonia celular. Enquanto na Seção 6.4 focalizamos o enlace sem fio do primeiro salto em redes celulares, aqui focalizaremos a mobilidade utilizando a arquitetura GSM de rede celular [Goodman, 1997; Mouly, 1992; Scourias, 1997; Kaaranen 2001; Korhonen, 2002] como objeto de nosso estudo, visto que é uma tecnologia madura e amplamente disponibilizada. Como no caso do IP móvel, veremos que vários dos princípios fundamentais que identificamos da Seção 6.5 estão incorporados à arquitetura de rede do GSM.

Do mesmo modo que o IP móvel, o GSM adota uma abordagem de roteamento indireto (veja Seção 6.5.2), primeiramente roteando a chamada do correspondente para a rede nativa do nó móvel e daí para a rede visitada. Em terminologia GSM a rede nativa do nó móvel é denominada **rede pública terrestre móvel nativa (PLMN nativa).** Visto que o acrônimo PLMN é um pouco complicado, e sempre firmes na nossa decisão de evitar uma sopa de letrinhas, denominaremos a rede PLMN nativa GSM simplesmente **rede nativa**.

A rede nativa é a provedora celular da qual o usuário móvel é assinante (isto é, a provedora que cobra mensalmente do usuário pelo serviço celular). A PLMN visitada, que denominaremos simplesmente **rede visitada**, é a rede na qual o nó móvel está residindo correntemente.

Como no caso do IP móvel, as responsabilidades das redes nativas e visitadas são bastante diferentes.

- A rede nativa mantém um banco de dados conhecido como **registro nativo de localização** (*home location register* — HLR), que contém o número permanente do telefone celular e as informações do perfil do assinante para cada um de seus assinantes. O importante é que o HLR também contém informações sobre as localizações correntes desses assinantes. Isto é, se um usuário móvel estiver correntemente em trânsito pela rede celular de uma outra provedora, o HLR conterá informações suficientes para obter (por meio de um processo que descreveremos em breve) um endereço na rede visitada para o qual deverá ser roteada uma chamada ao usuário móvel. Como veremos, quando é feita uma chamada para um usuário móvel, um comutador especial na rede nativa, conhecido como **Central de Comutação para Portal de Serviços Móveis** (*Gateway Mobile Services Switching Center* — GMSC), é contatado por um correspondente. Mais uma vez, conforme nossa decisão de evitar a sopa de letrinhas, denominaremos a GMSC por um termo mais descritivo, uma **MSC nativa**.
- A rede visitada mantém um banco de dados conhecido como **registro de localização de visitantes** (*visitor location register* — VLR). O VLR contém um registro para cada usuário móvel que está *correntemente* na porção da rede atendida pelo VLR. Assim, os registros do VLR vêm e vão à medida que usuários entram e saem da rede. Um VLR usualmente está localizado juntamente com a central de comutação móvel (MSC) que coordena o estabelecimento de uma chamada de e para a rede visitada.

Na prática, a rede celular de uma provedora servirá como uma rede nativa para seus assinantes e como uma rede visitada para usuários móveis que são assinantes de outra provedora de serviços celulares.

6.7.1 Roteando chamadas para um usuário móvel

Agora estamos prontos para descrever como é estabelecida uma chamada para um usuário GSM em uma rede visitada. Consideraremos um exemplo simples, representado na Figura 6.24; cenários mais complexos são descritos em [Mouly, 1992]. As etapas, como ilustradas na Figura 6.24, são as seguintes:

1. O correspondente disca o número do telefone do usuário móvel. Esse número, em si, não se refere a uma determinada linha telefônica ou localização (afinal, o número do telefone é fixo, mas o usuário é móvel!). Os dígitos iniciais do número são suficientes para identificar globalmente a rede nativa do usuário móvel. A chamada é roteada desde o correspondente, passa através da rede telefônica pública comutada e chega até a MSC na rede nativa do usuário móvel. Esse é o primeiro trecho da chamada.
2. A MSC nativa recebe a chamada e interroga o HLR para determinar a localização do usuário móvel. No caso mais simples, o HRL retorna o **número roaming da estação móvel** (*mobile station roaming number* — MSRN), que denominaremos **número de roaming**. Note que esse número é diferente do número permanente do telefone móvel, que é associado com a rede nativa do usuário móvel. O número de roaming é efêmero: é designado temporariamente a um usuário móvel quando ele entra em uma rede visitada. O número de roaming desempenha um papel semelhante ao do endereço administrado em IP móvel e, tal como o COA, é invisível para o correspondente e para o usuário móvel. Se o HLR não tiver o número de roaming, ele retornará o endereço do VLR na rede visitada. Nesse caso (que não é mostrado na Figura 6.24), a MSC nativa precisará consultar o VLR para obter o número de roaming do nó móvel. Mas, antes de mais nada, como o HLR obtém o número de roaming ou o endereço VLR? O que acontece a esses valores quando o usuário móvel passa para uma outra rede visitada? Em breve consideraremos essas perguntas importantes.

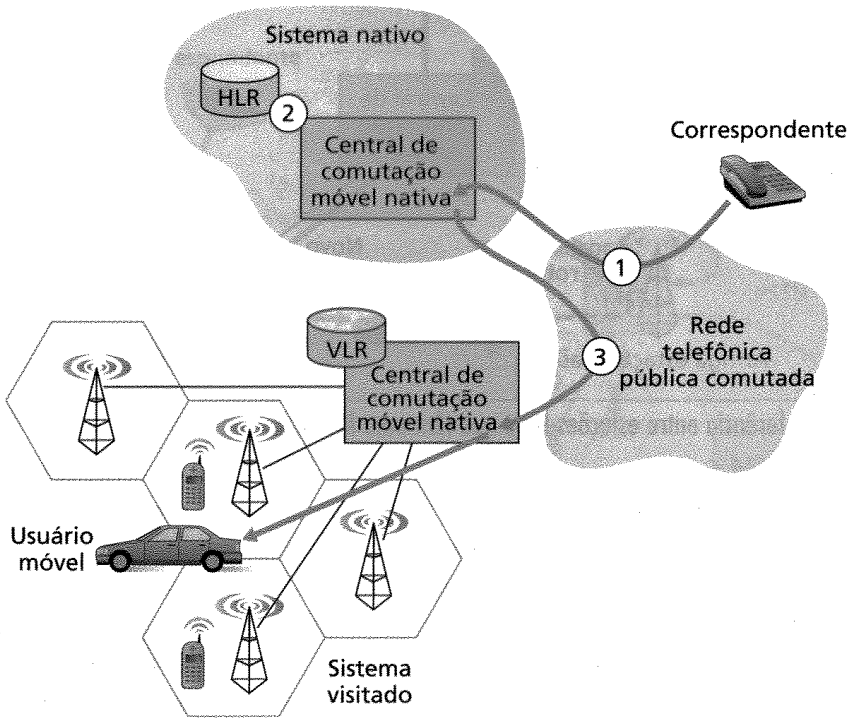


Figura 6.24 Estabelecendo uma chamada para um usuário móvel: roteamento indireto

3. Dado o número de roaming, a MSC nativa estabelece o segundo trecho da chamada através da rede até a MSC na rede visitada. A chamada está concluída — foi roteada do correspondente até a MSC nativa e daí para a MSC visitada, e desta até a estação-base que atende o usuário móvel.

Uma questão não resolvida na etapa 2 é como o HLR obtém informação sobre a localização do usuário móvel. Quando um telefone móvel é ligado ou entra em uma parte de uma rede visitada que é coberta por um novo VLR, ele deve se registrar junto à rede visitada. Isso é feito por meio da troca de mensagens de sinalização entre o usuário móvel e o VLR. O VLR visitado, por sua vez, envia uma mensagem de requisição de atualização de localização ao HLR do usuário móvel. Essa mensagem informa o HLR do número de roaming no qual o usuário móvel pode ser contatado ou o endereço do VLR (que então pode ser consultado mais tarde para obter o número do usuário móvel). Como parte dessa troca, o VLR também obtém do HRL informações sobre o assinante do usuário móvel e determina quais serviços (se houver algum) devem ser prestados ao usuário móvel pela rede visitada.

6.7.2 Transferências em GSM (handoffs)

Uma **transferência** (handoff) ocorre quando uma estação móvel muda sua associação de uma estação-base para outra durante uma chamada. Como mostra a Figura 6.25, uma chamada de telefone móvel é roteada inicialmente (antes da transferência) para o usuário móvel por meio de uma estação-base (a qual denominaremos estação-base antiga) e, após a transferência, por meio de uma outra estação-base (a qual denominaremos nova estação). Note que uma transferência entre estações-base resulta não somente em transferência/recepção de/para um telefone móvel e uma nova estação-base, mas também no *rerroteamento* da chamada em curso de um ponto de comutação dentro da rede para a nova estação-base. Vamos admitir inicialmente que as estações-base antiga e nova compartilham a mesma MSC e que o rerroteamento ocorre nessa MSC.

Há diversas razões possíveis para ocorrer transferência, incluindo (1) o sinal entre a estação-base corrente e o usuário móvel pode ter-se deteriorado a tal ponto que a chamada corre perigo de ser descartada e (2) uma célula pode ter ficado sobrecarregada, manipulando um grande número de chamadas. Esse congestionamento pode ser aliviado transferindo usuários móveis para células próximas menos congestionadas.

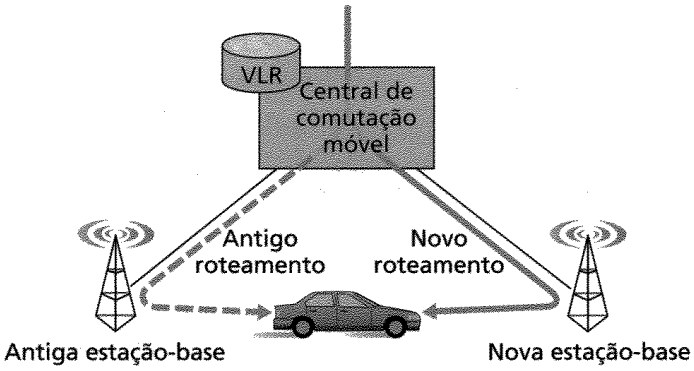


Figura 6.25 Cenário de transferência entre estações-base que têm uma MSC em comum

Enquanto está associado com uma estação-base, um usuário móvel mede periodicamente a potência de um sinal de sinalização emitido por sua estação-base corrente, bem como de sinais de sinalização emitidos por estações-base próximas que ele pode 'ouvir'. Essas medições são passadas uma ou duas vezes por segundo para a estação-base corrente do usuário móvel. A transferência em GSM é iniciada pela estação-base antiga com base nessas medições, nas cargas correntes de usuários móveis em células próximas e outros fatores [Mouly, 1992]. O padrão GSM não determina o algoritmo específico a ser utilizado por uma estação-base para decidir se realiza ou não uma transferência.

A Figura 6.26 ilustra as etapas envolvidas quando uma estação-base decide transferir um usuário móvel:

1. A antiga estação-base (BS) informa à MSC visitada que deve ser feita uma transferência e a BS (ou possível conjunto de BSs) para a qual o usuário móvel deve ser transferido.
2. A MSC visitada inicia o estabelecimento do caminho até a nova BS, alocando os recursos necessários para carregar a chamada rerroteada e sinalizando à nova BS que uma transferência está prestes a ocorrer.
3. A nova BS reserva e ativa um canal de rádio para ser utilizado pelo usuário móvel.
4. A nova BS devolve um sinal à MSC visitada e à antiga BS, indicando que foi estabelecido um caminho entre a MSC visitada e a nova BS e que o usuário móvel deve ser informado da transferência iminente. A nova BS provê todas as informações de que o usuário móvel necessitará para se associar com a nova BS.
5. O usuário móvel é informado de que deve realizar uma transferência. Note que, até esse ponto, o usuário móvel está alegremente inconsciente de que a rede estava preparando o terreno para uma transferência (por exemplo, reservando um canal na nova BS e alocando um caminho entre a MSC visitada e a nova BS).

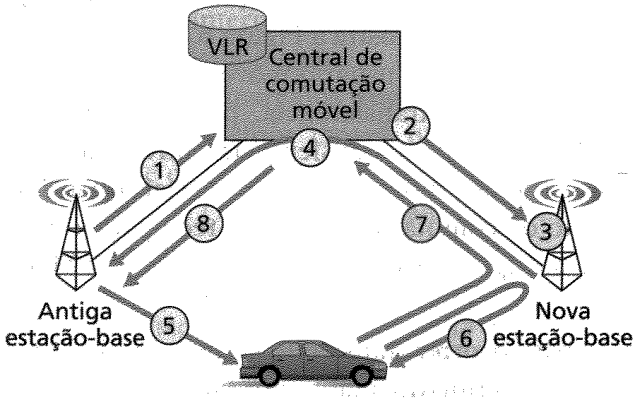


Figura 6.26 Etapas da execução de uma transferência entre estações-base que têm uma MSC em comum

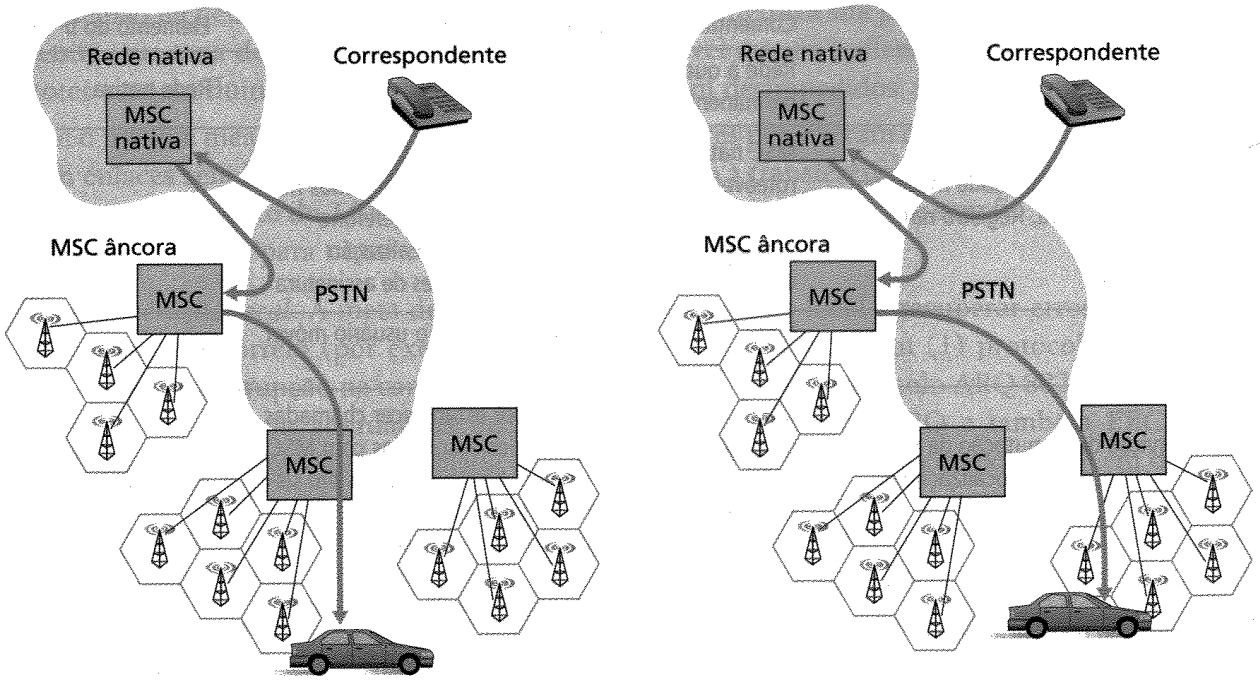


Figura 6.27 Rerroteamento via a MSC âncora

6. O usuário móvel e a nova BS trocam uma ou mais mensagens para ativar totalmente o novo canal na nova BS.
7. O móvel envia à nova BS uma mensagem de conclusão de transferência que é transmitida até a MSC visitada. Então, a MSC visitada rerroteia a chamada em curso para o usuário móvel, via a nova BS.
8. Os recursos reservados ao longo do caminho até a antiga BS são liberados.

Vamos concluir nossa discussão sobre transferência considerando o que acontece quando o usuário móvel passa para uma BS que está associada com uma MSC *diferente* da antiga BS e o que acontece quando essa transferência inter-MSCs ocorre mais de uma vez. Como mostra a Figura 6.27, o GSM define a noção de uma **MSC âncora**. A MSC âncora é a MSC visitada pelo usuário móvel logo no início de uma chamada; assim, a MSC âncora não muda durante a chamada. Por toda a duração da chamada e independentemente do número de transferências inter-MSCs realizadas pelo usuário móvel, a chamada é roteada desde a MSC nativa até a MSC âncora e, então, da MSC âncora até a MSC visitada, onde o usuário móvel está correntemente localizado. Quando um usuário móvel passa da área de cobertura de uma MSC para a área de cobertura de uma outra MSC, a chamada em curso é rerroteada desde a MSC âncora até a nova MSC visitada, que contém a nova estação-base. Assim, durante o tempo todo há, no máximo, três MSCs (a MSC local, a MSC âncora e a MSC visitada) entre o correspondente e o usuário móvel. A Figura 6.27 ilustra o roteamento de uma chamada entre as MSCs visitadas por um usuário móvel.

Em vez de manter um único salto desde a MSC âncora até a MSC corrente, uma abordagem alternativa seria simplesmente encadear as MSCs visitadas pelo móvel, fazendo com que uma MSC antiga transmitisse a chamada em curso até a nova MSC cada vez que o móvel passasse para uma nova MSC. Esse encadeamento de MSC pode realmente ocorrer em redes celulares IS-41, com uma etapa opcional de minimização de caminho para remover MSCs entre a MSC âncora e a nova MSC visitada [Lin, 2001].

Vamos encerrar nossa discussão sobre o gerenciamento da mobilidade GSM fazendo uma comparação entre gerenciamento de mobilidade em GSM e em IP Móvel. A comparação apresentada na Tabela 6.2 indica que, embora redes IP e celulares sejam fundamentalmente diferentes sob muitos aspectos, compartilham um número surpreendente de elementos funcionais comuns e abordagens gerais para o gerenciamento da mobilidade.

Elemento do GSM	Comentário sobre o elemento do GSM	Elemento do IP móvel
Sistema nativo	Rede à qual pertence o número de telefone permanente do usuário	Rede nativa
Central de comutação de unidade móvel ou simplesmente MSC nativa, Registro nativo de localização (HLR)	MSC nativa: ponto de contato para obter endereço roteável de usuário móvel. HLR: banco de dados no sistema nativo que contém número de telefone permanente, informações de perfil, localização corrente de usuário móvel, informações de assinatura	Agente nativo
Sistema visitado	Rede, exceto o sistema nativo, onde o usuário móvel está residindo correntemente	Rede visitada
Central de serviços de comutação de unidade móvel visitada, Registro de Localização de Visitante (VLR)	MSC visitada: responsável por estabelecer chamadas de/para nós móveis em células associadas com MSC. VLR: registro temporário em banco de dados em sistema visitado, contendo informações de assinatura para cada usuário móvel visitado	Agente externo
Número de roaming de estação móvel (MSRN) ou simplesmente número de roaming	Endereço roteável para segmento de chamada telefônica entre MSC nativa e MSC visitada, que não é visível nem para o usuário móvel nem para o correspondente	Endereço administrado

Tabela 6.2 Semelhanças entre a mobilidade em IP móvel e em GSM

6.8 Sem fio e mobilidade: impacto sobre protocolos de camadas superiores

Neste capítulo, vimos que redes sem fio são significativamente diferentes de suas contrapartes cabeadas tanto na camada de enlace (como resultado de características de canais sem fio como desvanecimento, propagação multivias e terminais ocultos) quanto na camada de rede (como resultado de usuários móveis que mudam seus pontos de conexão com a rede). Mas há diferenças importantes nas camadas de transporte e de aplicação? É tentador pensar que essas diferenças seriam minúsculas, visto que a camada de rede provê o mesmo modelo de serviço de entrega de melhor esforço às camadas superiores tanto em redes cabeadas quanto em redes sem fio.

De modo semelhante, se protocolos como TCP ou UDP são usados para prover serviços de camada de transporte a aplicações tanto em redes cabeadas como em redes sem fio, então a camada de aplicação também deve permanecer sem mudança. Nossa intuição está certa em um sentido — TCP e UDP podem operar, e realmente operam, em redes com enlaces sem fio. Por outro lado, protocolos de transporte em geral e o TCP em particular às vezes podem ter desempenhos muito diferentes em redes cabeadas e em redes sem fio, e é neste particular, em termos de desempenho, que as diferenças se manifestam. Vamos ver por quê.

Lembre-se de que o TCP retransmite um segmento que é perdido ou corrompido no caminho entre remetente e destinatário. No caso de usuários móveis, a perda pode resultar de congestionamento de rede (transbordamento de buffer de roteador) ou de transferência (por exemplo, de atrasos no rerroteamento de segmentos para um novo ponto de conexão do usuário à rede). Em todos os casos, o ACK receptor-emissor do TCP indica somente que um segmento não foi recebido intacto; o remetente não sabe se o segmento foi perdido devido a congestionamento, durante transferência, ou devido a erros de bits detectados. Em todos os casos, a resposta do remetente é a mesma — retransmitir o segmento. A resposta do controle de congestionamento do TCP *também* é a mesma em todos os casos — o TCP reduz sua janela de congestionamento, como discutimos na Seção 3.7. Reduzindo incondicionalmente sua janela de congestionamento, o TCP admite implicitamente que a perda de segmento resulta de congestionamento e não de corrupção ou transferência de usuário. Vimos na Seção 6.2 que erros de bits são muito mais comuns em redes sem fio do que em redes cabeadas. Quando ocorrem esses erros de bits ou quando ocorre perda na transferência de usuário,

na realidade não há nenhuma razão para que o remetente TCP reduza sua janela de congestionamento (reduzindo, assim, sua taxa de envio). Na verdade, é bem possível que os buffers de roteadores estejam vazios e que pacotes estejam fluindo ao longo de caminhos fim-a-fim desimpedidos, sem congestionamento.

Entre o início e meados da década de 1990, pesquisadores perceberam que, dadas as altas taxas de erros de bits em enlaces sem fio e a possibilidade de perdas devido à transferência de usuários, a resposta do controle de congestionamento do TCP poderia ser problemática em um ambiente sem fio. Há duas classes gerais de abordagens possíveis para tratar esse problema [Balakrishnan, 1995]:

- *Recuperação local.* A meta de abordagens de recuperação local é recuperar erros de bits quando e onde eles ocorrem (por exemplo, no enlace sem fio). Isso inclui (1) protocolos que recuperam perdas ou corrupção na camada de enlace (por exemplo, o protocolo ARQ 802.11, que estudamos na Seção 6.3, ou abordagens mais sofisticadas que utilizam ARQ e também FEC [Ayanoglu, 1995]); (2) protocolos de camada de transporte que dividem uma conexão TCP em dois segmentos, um da fonte até o enlace sem fio e um do enlace sem fio até o destino [Bakre, 1995; Brown, 1997]; e (3) protocolos de camada de enlace cientes do TCP [Balakrishnan, 1995; Liu, 2002].
- *Remetente TCP ciente de enlaces sem fio.* Em abordagens de recuperação local, o remetente TCP fica alegremente inconsciente de que seus segmentos estão atravessando um enlace sem fio. Uma abordagem alternativa é o remetente e o receptor ficarem cientes da existência de um enlace sem fio, para distinguir entre perdas por congestionamento que ocorrem na rede cabeada e corrupção/perdas que ocorrem no enlace sem fio e invocar controle de congestionamento somente em resposta a perdas por congestionamento na rede cabeada. [Balakrishnan, 1995] investiga vários tipos de TCP, admitindo que sistemas finais possam fazer essa distinção. [Wei, 2004] investiga técnicas para distinguir entre perdas nos segmentos cabeados e sem fio de um caminho fim-a-fim.

Aqui, nosso tratamento do TCP em enlaces sem fio foi necessariamente breve. Aconselhamos o leitor a consultar as referências se quiser mais detalhes sobre essa área de pesquisa.

Agora que já consideramos protocolos de camada de transporte, vamos considerar em seguida o efeito do sem fio e da mobilidade sobre protocolos de camada de aplicação. Uma consideração importante aqui é que enlaces sem fio muitas vezes têm larguras de banda relativamente baixas, como vimos na Figura 6.2. Conseqüentemente, aplicações que operam por enlaces sem fio, particularmente por enlaces celulares sem fio, devem tratar a largura de banda como uma mercadoria escassa. Por exemplo, um servidor Web que serve conteúdo a um browser Web que está rodando em um telefone 3G provavelmente não conseguirá prover o mesmo conteúdo rico em imagens que oferece a um browser que está rodando sobre uma conexão cabeada. Embora enlaces sem fio proponham desafios na camada de aplicação, a mobilidade que eles criam também torna possível um rico conjunto de aplicações dependentes de localização e cientes de contexto [Chen, 2000]. Em termos mais gerais, redes sem fio e redes móveis desempenharão um papel fundamental na concretização dos ambientes de computação ubíquos do futuro [Weiser, 1991]. É justo dizer que vimos somente a ponta do iceberg quando se trata do impacto de redes sem fio e móveis sobre aplicações em rede e seus protocolos!

6.9 Resumo

Redes sem fio e móveis revolucionaram a telefonia e também estão causando um impacto cada vez mais profundo no mundo das redes de computadores. Com o acesso à infra-estrutura da rede global que oferecem — desimpedido, a qualquer hora, em qualquer lugar — estão não somente aumentando a ubiqüidade do acesso a redes, mas também habilitando um novo conjunto muito interessante de serviços dependentes de localização. Dada a crescente importância das redes sem fio e móveis, este capítulo focalizou os princípios, as tecnologias de enlace e as arquiteturas de rede para suportar comunicações sem fio e móveis.

Iniciamos este capítulo com uma introdução às redes sem fio e móveis, traçando uma importante distinção entre os desafios propostos pela natureza *sem fio* dos enlaces de comunicação desse tipo de rede, e pela *mobilidade* que esses enlaces permitem. Isso nos permitiu isolar, identificar e dominar melhor os conceitos fundamentais em cada área. Focalizamos primeiramente a comunicação sem fio, considerando as características de um enlace sem fio na Seção 6.2. Nas seções 6.3 e 6.4 examinamos os aspectos de camada de enlace do padrão IEEE 802.11 para LANs sem fio (Wi-Fi) e do acesso celular à Internet. Então, voltamos nossa atenção para a questão da mobilidade. Na Seção 6.5 identificamos diversas formas de mobilidade e verificamos que há pontos ao longo desse espectro que propõem desafios diferentes e admitem soluções diferentes. Consideramos os problemas de localização e roteamento para um usuário móvel, bem como abordagens para transferir o usuário móvel que passa dinamicamente de um ponto de conexão com a rede para outro. Examinamos como essas questões foram abordadas no padrão IP móvel e em GSM nas seções 6.6 e 6.7, respectivamente. Finalmente, na Seção 6.8 consideramos o impacto causado por enlaces sem fio e pela mobilidade sobre protocolos de camada de transporte e aplicações em rede.

Embora tenhamos dedicado um capítulo inteiro ao estudo de redes sem fio e redes móveis, seria preciso todo um livro (ou mais) para explorar completamente esse campo tão animador e que está se expandindo tão rapidamente. Aconselhamos o leitor a se aprofundar mais nesse campo consultando as muitas referências fornecidas neste capítulo.

Exercícios de fixação

Capítulo 6 Questões de revisão

1. Descreva o papel dos quadros de sinalização em 802.11.
2. Discuta os métodos de autenticação de usuários disponíveis para redes 802.11.
3. Verdadeiro ou falso: antes de uma estação 802.11 transmitir um quadro de dados, ela deve primeiramente enviar um quadro RTS e receber um quadro CTS correspondente.
4. Por que são usados reconhecimentos em 802.11, mas não em Ethernet cabeada?
5. Falso ou verdadeiro: Ethernet e 802.11 usam a mesma estrutura de quadro.
6. Descreva como funciona o patamar RTS.
7. Suponha que os quadros RTS e CTS IEEE 802.11 fossem tão longos quanto os quadros padronizados DATA e ACK. Haveria alguma vantagem em usar os quadros CTS e RTS? Justifique sua resposta.
8. A Seção 6.3.4 discute mobilidade 802.11, na qual uma estação sem fio passa de um BSS para outro dentro da mesma sub-rede. Quando os APs estão interconectados com um comutador, um AP pode precisar enviar um quadro com um endereço MAC fingido para fazer com que o comutador transmita quadros adequadamente. Por quê?
9. Aprendemos na Seção 6.3.2 que há dois padrões 3G importantes: UMTS e CDMA-2000. A quais padrões 2G e 2,5G cada um desses dois padrões deve sua linhagem?

Problemas

1. Considere o exemplo do remetente CDMA único na Figura 6.4. Qual seria a saída do remetente (para os 2 bits de dados mostrados) se o código do remetente CDMA fosse (1, −1, 1, −1, 1, −1, 1, −1)?
2. Considere o remetente 2 na Figura 6.5. Qual é a saída do remetente para o canal (antes de ser adicionada ao sinal vindo do remetente 1) $Z_{i,m}^2$?

3. Suponha que o receptor na Figura 6.5 queira receber os dados que estão sendo enviados pelo remetente 2. Mostre (por cálculo) que o receptor pode, na verdade, recuperar dados do remetente 2 do sinal agregado do canal usando o código do remetente 2.
4. Suponha que dois ISPs fornecem acesso Wi-Fi em um determinado café, e que cada um deles opera seu próprio AP e tem seu próprio bloco de endereços IP.
   a. Suponha ainda mais, que, por acidente, cada ISP configurou seu AP para operar no canal 11. O protocolo 802.11 falhará totalmente nessa situação? Discuta o que acontece quando duas estações, cada uma associada com um ISP diferente, tentam transmitir ao mesmo tempo.
   b. Agora suponha que um AP opera no canal 1 e outro no canal 11.
5. Na etapa 4 do protocolo CSMA/CA, uma estação que transmite um quadro com sucesso inicia o protocolo CSMA/CA para um segundo quadro na etapa 2, e não na etapa 1. Quais seriam as razões que os projetistas do CSMA/CA provavelmente tinham em mente para fazer com que essa estação não transmitisse o segundo quadro imediatamente (se o canal fosse percebido como ocioso)?
6. Suponha que uma estação 802.11b seja configurada para sempre reservar o canal com a seqüência RTS/CTS. Suponha que essa estação repentinamente queira transmitir 1.000 bytes de dados e que todas as outras estações estão ociosas nesse momento. Calcule o tempo requerido para transmitir o quadro e receber o reconhecimento como uma função de SIFS e DIFS, ignorando atraso de propagação e admitindo que não haja erros de bits.
7. Na Seção 6.5, uma solução proposta que permitia que usuários móveis mantivessem seu endereço IP à medida que transitavam entre redes externas era fazer com que uma rede externa anunciasse ao usuário móvel uma rota altamente específica e usasse a infra-estrutura de roteamento existente para propagar essa informação por toda a rede. Uma das preocupações que identificamos foi a escalabilidade. Suponha que, quando um usuário móvel passe de uma rede para outra, a nova rede externa anuncie uma rota específica para o usuário móvel e a antiga rede externa retire sua rota. Considere como informações de roteamento se propagam em um algoritmo vetor de distâncias (particularmente para o caso de roteamento interdomínios entre redes que abrangem o globo terrestre).
   a. Outros roteadores conseguirão rotear datagramas imediatamente para a nova rede externa tão logo essa rede comece a anunciar sua rota?
   b. É possível que roteadores diferentes acreditem que redes externas diferentes contenham o usuário móvel?
   c. Discuta a escala temporal segundo a qual outros roteadores na rede eventualmente aprenderão o caminho até os usuários móveis.
8. Suponha que o correspondente na Figura 6.17 fosse móvel. Faça um desenho esquemático da infra-estrutura adicional de camada de rede que seria necessária para rotear o datagrama do usuário móvel original até o correspondente (que agora é móvel). Mostre a estrutura do(s) datagrama(s) entre o usuário móvel original e o correspondente (agora móvel), como na Figura 6.18.
9. Em IP móvel, que efeito terá a mobilidade sobre atrasos fim-a-fim de datagramas entre a fonte e o destino?
10. Considere o exemplo de encadeamento discutido no final da Seção 6.7.2. Suponha que um usuário móvel visite as redes externas A, B e C, e que um correspondente inicie uma conexão com o usuário móvel enquanto este reside na rede externa A. Relacione a seqüência de mensagens entre agentes externos e entre agentes externos e o agente nativo, enquanto o usuário passa da rede A para a rede B e para a rede C. Em seguida, suponha que não é executado encadeamento e que as mudanças no endereço administrado do usuário móvel devem ser notificadas explicitamente ao correspondente (bem como ao agente nativo). Relacione a seqüência de mensagens que seria necessário trocar nesse segundo cenário.
11. Considere dois nós móveis em uma rede externa que tem um agente externo. É possível que os dois nós móveis utilizem o mesmo endereço administrado em IP móvel? Explique sua resposta.

12. Quando discutimos como o VLR atualizava o HLR com informações sobre a localização corrente de usuários móveis, quais eram as vantagens e as desvantagens de fornecer ao HLR o MSRN em vez do endereço do VLR?

Questões dissertativas

1. Relacione cinco produtos existentes hoje no mercado que forneçam uma interface Bluetooth ou 802.15.
2. O serviço 3G sem fio está disponível em sua região? Quando custa? Quais aplicações ele suporta?
3. Que tipos de problemas você observou como usuário do IEEE 802.11? Como os projetos do 802.11 podem evoluir para superar esses problemas?

Ethereal Lab

No Companion Website deste livro, `www.aw.com/kurose_br`, *você encontrará um Ethereal Lab, em inglês, para este capítulo que captura e estuda os quadros 802.11 trocados entre um laptop sem fio e um ponto de acesso.*

Entrevista

Charlie Perkins

Charles E. Perkins é um Nokia Fellow (Pesquisador Associado Nokia) no Communication System Laboratory e no Nokia Research Center, onde estuda a interconexão de redes móveis sem fio e protocolos de configuração dinâmica. É editor de diversos periódicos da ACM e do IEEE para áreas relacionadas à interconexão de redes sem fio. Trabalha como editor de documentos para o grupo de trabalho IP móvel da Internet Engineering Task Force (IETF) e é autor ou co-autor de documentos de padrões para os grupos de trabalho mobileip, manet, IPv6 e seamoby (Seamless Mobility). Também é editor associado da *Mobile Communications and Computing Review*, a publicação oficial do ACM SIGMOBILE, e trabalha no quadro editorial da revista *IEEE Internet Computing*. Charles escreveu e editou livros sobre IP Móvel e Redes Ad Hoc e publicou vários estudos e artigos premiados nas áreas de redes móveis, redes ad hoc, otimização de rota para redes móveis, descoberta de recursos e configuração automática para computadores móveis.

O que o fez se decidir pela especialização em sem fio/mobilidade?

Meu envolvimento com redes sem fio e mobilidade foi uma conseqüência natural do meu trabalho em projetos na IBM Research no final da década de 1980. Nós tínhamos enlaces de rádio e estávamos tentando construir um equipamento ao estilo do 'ThinkPad' (digamos, um Palm Pilot) com conectividade sem fio e reconhecimento de escrita.

Construímos uma solução simples (mais tarde denominada 'Mobile IP' (IP Móvel)) e percebemos que ela funcionava. É claro que isso não é comum em comparação com grande parte das soluções de pesquisa. Aproveitando nossa experiência com o Mobile IP, projetamos uma modificação rápida e efetiva para o RIP — e que também funcionou muito bem. Aqui, 'funcionar' quer dizer que as aplicações rodavam bem sem nenhuma modificação e que o desempenho da rede não foi prejudicado por nossos novos projetos. Essas propriedades ficaram conhecidas como 'transparência da aplicação' e 'escalabilidade'. É claro que funcionar em laboratório é muito diferente de alcançar sucesso comercial, e ambas as tecnologias têm muito potencial ainda não realizado comercialmente.

Qual foi seu primeiro emprego no setor de computação?

Trabalhei na TRW Controls, em Houston, Texas. Foi uma mudança drástica em relação ao estudo universitário.

Uma coisa que aprendi na TRW Controls é como o suporte de software é precário até mesmo para os sistemas mais críticos de controle de utilidades. Esses sistemas deviam controlar o fluxo de energia elétrica em redes de enorme potência e o modo como era construído o software subjacente era de arrepiar os cabelos. E mais, os cronogramas eram sempre apertados e os programadores eram profundamente céticos em relação às suas condições de trabalho e às intenções da diretoria. Era preciso projetar o sistema inteiro a partir do zero. Não tenho muitas razões para achar que as coisas mudaram nesses últimos 30 anos, especialmente considerando os eventos recentes que cercaram o apagão de 2003. Na verdade, dada a desregulamentação, certamente ficaram até piores. Para mim foi uma felicidade sair da TRW Controls e passar para a Tektronix (Tek Labs).

Qual parte do seu trabalho lhe propõe mais desafios?

A parte do meu trabalho que me propõe mais desafios é entender o que devo fazer para ajudar minha empresa. E também considero como parte do meu trabalho moldar as tecnologias sem fio com as quais entro em contato e fornecer às pessoas serviços melhores e uma experiência diária mais agradável. Minha empresa trabalha no ramo de 'conectar pessoas' e espero ajudá-la a fazer com que essas conexões sejam o mais harmoniosas e suaves possível. Conseguir fazer isso de um modo tal que também maximize o lucro potencial das tecnologias que desenvolvemos transforma cada dia em um novo desafio. Especificamente nas áreas sem fio, acho que a tecnologia de segurança tem de ser desenvolvida de modo a ser conveniente e apreciada (como uma capa de chuva), em vez de pesada e temida (como é, em grande parte, hoje).

Em um nível técnico mais detalhado, no qual certamente me sinto mais à vontade, tento resolver problemas de protocolos de rede de modo a aplicar o mínimo de carga possível sobre os equipamentos sem fio (e suas baterias!) e provocar a mínima inconveniência aos usuários. Interconectar os telefones sem fio de hoje com a Internet por meio de tecnologias sem fio de alta velocidade é extremamente interessante tecnicamente e oferece potencial ilimitado de sucesso comercial para quem conseguir encontrar os caminhos certos para seguir em frente.

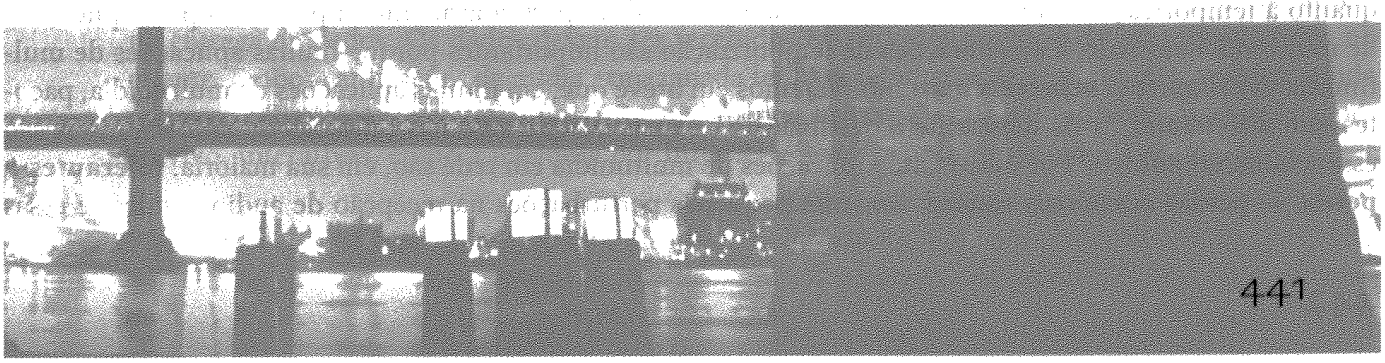
Em sua opinião, qual é o futuro da tecnologia sem fio?

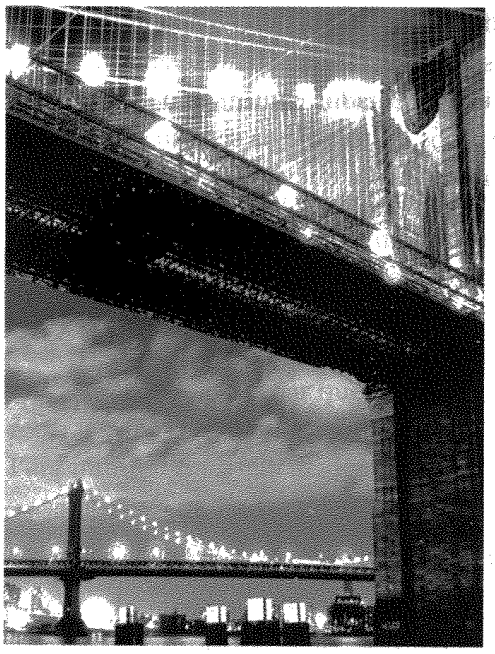
O setor, como um todo, está passando por tremendas mudanças e ainda não há nenhum final à vista. Estão surgindo novas tecnologias sem fio de alta velocidade que poderão causar efeitos práticos imprevisíveis e poderiam mudar fundamentalmente a sociedade. Nossas expectativas atuais em relação à privacidade e às limitações à nossa capacidade de nos comunicar uns com os outros (voz, imagem e dados) poderão ser irreconhecíveis dentro de dez anos. À medida que as empresas adotam cada vez mais comunicações sem fio, é bem possível que novas medidas de segurança sejam tomadas, o que mudaria significativamente nossa experiência no local de trabalho.

Parece bem claro que conseguiremos alocar mais espectro a vários esquemas de comunicações por rádio, cujas velocidades poderão ser muito altas. A minha expectativa é que logo será muito viável para as comunidades oferecer a seus cidadãos comunicações sem fio de alta velocidade completas, como se a cidade inteira fosse uma rede local. Isso poderia ter um efeito revigorante sobre o sentimento de comunidade que há muito tempo se perdeu em nossa sociedade, pelo menos nos Estados Unidos. É claro que a comunidade ainda demandaria acesso à Internet. O custo do armazenamento em disco está ficando tão baixo que podemos imaginar acesso sem fio de alta velocidade a acervos imensuráveis de tesouros intelectuais da humanidade, bem como às últimas notícias.

A tecnologia sem fio provavelmente acelerará o crescimento da Internet. À medida que equipamentos sem fio ficarem cada vez mais baratos, as comunicações pela Internet estarão por toda a parte (em plugues de ouvido, em jogos multiusuário, em leitoras de bilhetes de metrô), o que motivará novas aplicações e novas soluções de segurança.

Essas previsões já foram feitas muitas vezes, mas somente nos últimos anos a tecnologia ficou disponível. Agora, as grandes barreiras são o gerenciamento de direitos e os controles de acesso. Se os cidadãos não se engajarem no processo de formulação de seus direitos nessas questões, os sonhos há muito acalentados continuarão sonhos. Ou, pior, poderão ser substituídos por novos pesadelos. A Internet é assunto de todos e a tecnologia sem fio a torna algo muito presente e pessoal. E eu estou aqui para ajudá-la a ser a ferramenta maravilhosa que deve ser.





Capítulo 7

Redes Multimídia

Com a conclusão de nossa jornada pela pilha de protocolos na primeira parte deste livro, temos agora uma base segura de princípios e práticas de redes de computadores. Essa base será de grande valia para abordar um tópico que atravessa muitas camadas da pilha de protocolos: as redes multimídia.

Nos últimos anos, houve um crescimento extraordinário do desenvolvimento e da disseminação de aplicações em rede que transmitem e recebem áudio e vídeo pela Internet. Novas aplicações de rede multimídia (também conhecidas como aplicações de mídia de taxa constante) — vídeo em tempo real, telefonia IP, rádio por Internet, teleconferência, jogos interativos, mundos virtuais, aprendizado a distância e muitas outras aplicações — são anunciadas todos os dias. Os requisitos de serviço dessas aplicações são significativamente diferentes daqueles das aplicações elásticas (e-mail, WEB, login remoto, compartilhamento de arquivos) que examinamos no Capítulo 2. Em particular, muitas aplicações de multimídia são muito sensíveis ao atraso fim-a-fim e à variação do atraso, mas podem tolerar perdas de dados ocasionais. Na primeira metade deste capítulo estudaremos como aplicações multimídia podem ser projetadas para obter o máximo do serviço de melhor esforço da Internet, que não dá nenhuma garantia contra atrasos fim-a-fim. Na segunda parte deste capítulo, examinaremos várias atividades que estão sendo desenvolvidas atualmente para ampliar a arquitetura da Internet de modo a prover suporte explícito para os requisitos de serviço de aplicações multimídia.

7.1 Aplicações de rede multimídia

Em nossa discussão sobre requisitos de serviço de aplicação no Capítulo 2, identificamos algumas linhas ao longo das quais esses requisitos podem ser classificados. Duas dessas linhas — considerações quanto à temporização e à tolerância à perda de dados — são particularmente importantes para aplicações de multimídia em rede. Considerações de temporização são importantes porque muitas aplicações de multimídia são altamente **sensíveis a atraso**. Veremos em breve que, em muitas aplicações de multimídia, pacotes que sofrem atrasos de remetente a destinatário de mais do que algumas centenas de milissegundos são essencialmente inúteis. Por outro lado, aplicações de multimídia em rede são, em sua maioria, **tolerantes à perda** — perdas ocasionais causam somente pequenas perturbações na recepção de áudio e vídeo, e essas

perdas podem ser parcial ou totalmente encobertas. Essas características de sensibilidade a atraso e de tolerância à perda são claramente diferentes daquelas das aplicações elásticas, como a Web, e-mail, FTP e Telnet. Para aplicações elásticas, atrasos longos são incômodos, mas não particularmente prejudiciais, e a completude e a integridade dos dados transferidos são de suma importância.

7.1.1 Exemplos de aplicações multimídia

A Internet comporta uma grande variedade de aplicações de multimídia interessantes. Nesta subseção, consideraremos três classes abrangentes de aplicações de multimídia: áudio e vídeo de fluxo contínuo armazenados, áudio e vídeo de fluxo contínuo ao vivo e áudio e vídeo interativos em tempo real.

Neste capítulo *não* consideraremos aplicações cujos dados são transportados antes e reproduzidos depois, tal como um arquivo MP3, que é descarregado completamente por meio de uma aplicação de compartilhamento de arquivos P2P antes de ser ouvido. Na verdade, aplicações desse tipo são elásticas, isto é, são aplicações de transferência de arquivos que não têm nenhum requisito especial quanto a atrasos. Examinamos transferência de arquivos (HTTP e FTP) e sistemas de compartilhamento de arquivos P2P no Capítulo 2.

Áudio e vídeo de fluxo contínuo armazenados

Nessa classe de aplicações, clientes requisitam, sob demanda, arquivos de áudio e vídeo comprimidos que estão armazenados em servidores. Arquivos de áudio armazenados podem conter o áudio da conferência de um professor (o leitor pode visitar o site Web deste livro e fazer um teste), canções de rock, sinfonias, arquivos de programas de rádio famosos ou gravações históricas. Arquivos de vídeo armazenados podem conter o vídeo de uma palestra, filmes de longa duração, programas de televisão gravados, documentários, arquivos em vídeo de eventos históricos, desenhos animados ou videoclipes. Essa classe de aplicações tem três características fundamentais.

- *Mídia armazenada.* O conteúdo multimídia foi pré-gravado e está armazenado num servidor. Conseqüentemente, o usuário pode fazer pausa, voltar, avançar ou escolher itens no índice de conteúdo da gravação. O tempo decorrido entre a solicitação de uma dessas ações pelo cliente e a execução da ação deve ser da ordem de 1 a 10 segundos para que a resposta seja considerada aceitável.
- *Fluxo contínuo.* Em uma aplicação de áudio/vídeo armazenado, normalmente o cliente inicia a reprodução do áudio/vídeo alguns segundos após começar a receber o arquivo do servidor. Isso significa que o cliente estará reproduzindo áudio/vídeo de uma parte do arquivo ao mesmo tempo que está recebendo do servidor partes do arquivo que estão mais à frente. Essa técnica, conhecida como **fluxo contínuo** (streaming), evita ter de descarregar o arquivo inteiro (e incorrer em atraso potencialmente longo) antes de começar a reproduzi-lo. Há muitos produtos para reprodução de multimídia de fluxo contínuo, entre eles o RealPlayer, da RealNetworks [RealNetworks, 2000], o QuickTime, da Apple [QuickTime, 2004] e o Windows Media, da Microsoft [Microsoft Windows Media Player, 2004].
- *Reprodução contínua.* Assim que se inicia a reprodução do conteúdo da multimídia, ela deve prosseguir de acordo com a temporização original da gravação. Isso impõe sérias restrições ao atraso na entrega de dados. Os dados devem ser recebidos do servidor a tempo de ser reproduzidos no cliente. Embora aplicações de mídia armazenada tenham requisitos de reprodução contínua, ainda assim suas limitações de atraso fim-a-fim são menos severas do que as limitações para aplicações ao vivo, interativas, como a telefonia por Internet e a videoconferência (veja a seguir).

História

Realnetworks: levando o áudio para a linha de frente da internet

A RealNetworks, pioneira em produtos de fluxo contínuo de áudio e vídeo, foi a primeira empresa a levar o áudio para a linha de frente da Internet. A empresa surgiu em 1995 com o nome de Progressive Networks. Seu produto inicial — o sistema RealAudio — continha um codificador, um servidor e um reprodutor de áudio. O sistema RealAudio permitia que os usuários pesquisassem, selecionassem e reproduzissem conteúdo de áudio sob demanda com tanta facilidade quanto se estivessem usando um aparelho reprodutor/gravador de videocassete comum. O produto alcançou rápida popularidade entre provedores de conteúdo de entretenimento, informações e notícias, para entrega de serviços de áudio sob demanda que podem ser acessados e reproduzidos imediatamente. No início de 1997, a RealNetworks ampliou sua linha de produtos para incluir vídeo, além de áudio.

Nos últimos anos, a RealNetworks vem enfrentando forte concorrência da Microsoft (que também começou a comercializar seus próprios produtos de mídia de fluxo contínuo no final da década de 1990). A RealNetworks e a Microsoft têm divergido quanto à escolha de algumas das tecnologias subjacentes para seus transdutores. Declarada a guerra no mercado e nos grupos de padrões da Internet, ambas as empresas estão tentando fazer com que seus próprios formatos e protocolos se transformem em padrão para a Internet.

Áudio e vídeo de fluxo contínuo ao vivo

Essa classe de aplicação é semelhante à transmissão tradicional de rádio e televisão, exceto que a transmissão é realizada pela Internet. Essas aplicações permitem que um usuário receba uma transmissão de rádio ou televisão ao vivo de qualquer parte do mundo. Por exemplo, um dos autores deste livro sempre ouve suas emissoras de rádio favoritas da Filadélfia quando está viajando. O outro autor ouviu regularmente as transmissões ao vivo dos jogos de seu amado time de basquete universitário quando morou na França por um ano.

Como o fluxo contínuo de áudio e vídeo ao vivo não é armazenado, um cliente não pode adiantar o programa que está recebendo. Contudo, com armazenamento local de dados recebidos, outras operações interativas, como pausa e retrocesso em transmissões de multimídia ao vivo, são possíveis em algumas aplicações. No caso de aplicações ao vivo de transmissão *broadcast*, freqüentemente há muitos clientes recebendo o mesmo programa de áudio/vídeo. A distribuição de áudio/vídeo ao vivo para muitos receptores pode ser realizada eficientemente utilizando as técnicas de multicasting IP que estudamos na Seção 4.7. Na época em que estávamos escrevendo este livro, no entanto, a distribuição de áudio/vídeo ao vivo era mais comumente realizada por meio de fluxos *unicast* múltiplos separados. Como acontece com a multimídia armazenada de fluxo contínuo, ela requer reprodução contínua, embora as limitações de temporização sejam menos severas do que para aplicações interativas em tempo real. Atrasos de até dezenas de segundos desde o instante em que o usuário requisita a entrega/reprodução de uma transmissão ao vivo até o início da reprodução podem ser tolerados.

Áudio e vídeo interativos em tempo real

Essa classe de aplicações permite que as pessoas usem áudio/vídeo para se comunicar entre si em tempo real. Áudio interativo em tempo real pela Internet é freqüentemente denominado **telefone por Internet**, já que, da perspectiva do usuário, é semelhante ao tradicional serviço telefônico por comutação de circuitos. O telefone por Internet pode fornecer, potencialmente, serviços telefônicos de central privada de comutação telefônica (PBX), serviços locais e de longa distância, a custo muito baixo. Também pode facilitar a disponibilização de novos serviços que não são suportados com facilidade pelas redes tradicionais de comutação de circuitos, incluindo integração telefone/Web, comunicação em grupo em tempo real, serviços de lista telefônica, filtragem de chamadas e outros mais. Hoje, há centenas de produtos de telefonia por Internet dis-

poníveis [VON, 2004]. Por exemplo, usuários do Instant Messenger da Microsoft podem fazer chamadas por voz de computador para telefone e de computador para computador. Com vídeo interativo em tempo real, também denominado videoconferência, indivíduos se comunicam visualmente, assim como oralmente. Há também muitos produtos de vídeo interativo em tempo real disponíveis para a Internet, incluindo o NetMeeting da Microsoft. Note que, em uma aplicação de áudio/vídeo interativo em tempo real, um usuário pode falar ou se mover a qualquer instante. Para uma conversação com interação entre vários usuários, o atraso entre o momento em que um usuário fala ou se move e o momento em que a ação se manifesta nos hospedeiros receptores deve ser menor do que algumas centenas de milissegundos. No caso da voz, atrasos menores do que 150 milissegundos não são percebidos pelo ouvido humano, atrasos entre 150 e 400 milissegundos podem ser aceitáveis e atrasos que excedem 400 milissegundos podem resultar em conversas frustrantes, se não completamente ininteligíveis.

7.1.2 Obstáculos para a multimídia na Internet de hoje

Lembre-se de que o protocolo IP disponibilizado na Internet de hoje provê um **serviço de melhor** esforço a todos os datagramas que transporta. Em outras palavras, a Internet faz seu melhor esforço para transportar cada datagrama do remetente ao receptor o mais rapidamente possível, mas não faz nenhuma promessa sequer sobre o atraso fim-a-fim para um pacote individual. Tampouco faz promessas quanto à variação do atraso de pacote dentro de uma corrente de pacotes. Como TCP e UDP executam sobre IP, nenhum desses protocolos pode dar alguma garantia contra atraso às aplicações requisitantes. Devido à falta de qualquer esforço especial para entregar pacotes a tempo, é um problema extremamente desafiador desenvolver aplicações de rede multimídia de sucesso para a Internet. Até o momento, a multimídia pela Internet alcançou um sucesso significativo, porém limitado. Por exemplo, a recepção em fluxo contínuo de áudio/vídeo armazenado com atrasos de 5 a 10 segundos na interatividade com o usuário agora é comum na Internet. Mas, durante os horários de pico de tráfego, o desempenho pode ser insatisfatório, sobretudo quando os enlaces intervenientes estão congestionados (como os congestionados enlaces transoceânicos).

Telefone por Internet e vídeo interativo em tempo real alcançaram, até agora, sucesso bem menor do que a recepção em fluxo contínuo de áudio/vídeo armazenado. Na verdade, voz e vídeo interativos em tempo real impõem rígidas limitações ao atraso de pacote e à variação de atraso do pacote. **Variação de atraso** de pacote significa que os pacotes experimentam atrasos diferentes dentro da mesma corrente de pacotes. Voz e vídeo em tempo real funcionam bem em regiões onde a oferta de largura de banda é abundante e, portanto, o atraso e a variação de atraso são mínimos. Mas a qualidade pode se deteriorar até níveis inaceitáveis assim que a corrente de pacotes de voz e vídeo atinge um enlace moderadamente congestionado.

O projeto de aplicações de multimídia seria certamente menos complicado se houvesse algum tipo de classificação dos serviços na Internet em primeira e segunda classe, pela qual pacotes de primeira classe teriam número limitado e receberiam serviço prioritário em filas de roteadores. Esse serviço de primeira classe poderia ser satisfatório para aplicações sensíveis ao atraso. Mas, até agora, a Internet adotou preferencialmente uma postura igualitária em relação ao escalonamento de pacotes em filas de roteadores. Todos os pacotes recebem serviço igual; nenhum pacote, incluindo os pacotes de áudio e vídeo, sensíveis ao atraso, recebe prioridade especial nas filas de roteadores. Não importa quanto dinheiro você tenha ou quão importante você seja: você tem de se juntar ao fim da fila e esperar sua vez! Na segunda metade deste capítulo, examinaremos as arquiteturas propostas que visam a acabar com essa limitação.

Assim, por enquanto temos de viver com o serviço de melhor esforço. Mas, dada essa limitação, podemos tomar diversas decisões quanto ao projeto e utilizar algumas artimanhas para melhorar a qualidade de uma aplicação de rede multimídia percebida pelo usuário. Por exemplo, podemos enviar áudio e vídeo sobre UDP e, assim, evitar a baixa vazão do TCP quando este entra em sua fase de partida lenta. Podemos retardar a reprodução no receptor em 100 milissegundos ou mais para reduzir os efeitos da variação de atraso induzida pela rede. Podemos colocar marca de tempo nos pacotes no remetente, de modo que o receptor saiba quando eles devem ser reproduzidos. Para áudio/vídeo armazenados, podemos adquirir dados anteci-

História

Voz pela Internet

Dada a popularidade universal do sistema de telefonia, desde o final da década de 1980 muitos visionários da Internet fizeram repetidas previsões de que a próxima aplicação a ganhar imensa popularidade na Internet seria algum tipo de aplicação de voz. Essas previsões foram acompanhadas de pesquisas sobre telefonia por Internet e desenvolvimento de produtos. Por exemplo, pesquisadores criaram protótipos de telefone Internet na década de 1980, aproveitando pesquisas sobre transmissão de voz empacotada realizadas na década de 1970 [Cohen, 1977; RFC 741], anos antes da popularização da Web. Numerosas empresas estreantes no mercado produziram produtos de telefonia por Internet PC para PC durante toda a década de 1990. Mas nenhum desses protótipos ou produtos foi realmente aceito pelos usuários da Internet (mesmo quando alguns vinham conjugados a browsers populares). Foi somente em 1999 que a comunicação por voz começou a se popularizar na Internet.

A aplicação PC para telefone começou a conquistar significativa utilização no final da década de 1990. Essa aplicação permite que um usuário da Internet, possuidor de uma conexão com a Internet e um microfone, faça ligações para qualquer telefone comum. A Net2Phone [Net2Phone, 2004] e a Dialpad [Dialpad, 2004] são duas empresas que fornecem serviços PC para telefone. Como o custo desses serviços é baixo, eles são populares entre pessoas que adoram falar ao telefone, mas cujo orçamento é apertado. Além disso, várias empresas oferecem ao mundo corporativo serviços semelhantes ao PC para telefone. Como discutiremos neste capítulo, quando se faz uma chamada telefônica PC para telefone, a chamada é roteada pela Internet de um PC até um roteador de borda (gateway) e por redes de telefonia de comutação de circuitos do roteador de borda até o telefone (fixo ou móvel). Vários protocolos são muito utilizados para comunicação pela Internet entre o PC e o roteador de borda, entre eles RTP, SIP e H.323 — todos são discutidos neste capítulo. Uma segunda classe de aplicações é telefone para telefone na Internet, que permite que consumidores façam chamadas de longa distância a custo mais baixo utilizando aparelhos telefônicos normais em ambas as extremidades. Muitos dos cartões telefônicos pré-pagos que podem ser comprados em bancas de jornais e algumas lojas utilizam o modo telefone-Internet-telefone. Uma terceira classe de aplicações é a transmissão assíncrona de voz pela Internet [Wimba, 2004]. Grandes empresas fornecedoras de equipamentos para redes de computadores — entre elas Cisco, Lucent e Alcatel — oferecem linhas completas de produtos de voz por IP para operadoras de telefonia e clientes empresariais [VON, 2004].

padamente durante a reprodução, quando houver capacidade de arquivo e largura de banda extra disponíveis no cliente. Podemos até enviar informações redundantes para atenuar os efeitos da perda de pacotes induzida pela rede. Examinaremos muitas dessas técnicas no restante da primeira metade deste capítulo.

7.1.3 Como a Internet deveria evoluir para dar melhor suporte à multimídia?

Hoje, há um debate extraordinário — e, às vezes, feroz — sobre como a Internet deveria evoluir para melhor atender o tráfego multimídia com suas limitações rígidas de temporização. Em um extremo, alguns pesquisadores argumentam que devem ser feitas modificações fundamentais na Internet para que aplicações possam reservar explicitamente largura de banda fim-a-fim. Eles acham, por exemplo, que, se um usuário quiser fazer uma chamada telefônica pela Internet do hospedeiro A para o hospedeiro B, a aplicação de telefone por Internet do usuário deveria poder reservar largura de banda explicitamente em cada enlace ao longo da rota entre os dois hospedeiros. Mas permitir que as aplicações façam reservas e exigir que a rede honre essas reservas implica algumas grandes mudanças. Primeiramente, precisamos de um protocolo que reserve largura de banda de enlace no caminho entre os remetentes e seus receptores, em nome das aplicações. Em segundo lugar, devemos modificar políticas de escalonamento nas filas dos roteadores, de modo que as reservas de largura de banda possam ser honradas. Com essas novas políticas de escalonamento, nem todos os pacotes receberiam o mesmo tratamento; em vez disso, quem reservar (e pagar) mais, recebe mais.

Em terceiro lugar, para honrar as reservas, as aplicações devem dar à rede uma descrição do tráfego que pretendem enviar para dentro dela. A rede deve, então, inspecionar o tráfego de cada aplicação para assegurar que ela cumpra o que descreveu. Finalmente, a rede deve dispor de meios para determinar se tem largura de banda suficiente para suportar qualquer nova solicitação de reserva. Esses mecanismos, quando combinados, exigem novos e complexos programas nos hospedeiros e roteadores, bem como novos tipos de serviços. Examinaremos esses mecanismos mais detalhadamente quando abordarmos o modelo Intserv na Seção 7.8.

No outro extremo, alguns pesquisadores argumentam que não é necessário fazer nenhuma modificação fundamental no serviço de melhor esforço e nos protocolos subjacentes da Internet. Em vez disso, defendem uma abordagem liberal:

- À medida que a demanda aumenta, os ISPs (de nível mais alto e de nível mais baixo) ampliarão suas redes para atendê-la. Especificamente, as provedoras adicionarão mais largura de banda e capacidade de comutação para oferecer desempenho satisfatório de atraso e perda de pacotes dentro de suas redes. Desse modo, essas empresas oferecerão melhor serviço a seus clientes (usuários e ISPs clientes), que serão traduzidos em receitas mais altas resultantes de um número maior de clientes e de tarifas de serviços também mais altas. Como discutimos no Capítulo 2, ISPs também podem instalar caches em suas redes, o que traria o conteúdo armazenado (páginas Web, bem como áudio e vídeo armazenados) para mais perto dos usuários, reduzindo, assim, o tráfego em ISPs de nível mais alto.
- Redes de distribuição de conteúdo (*content distribution networks* — CDNs) duplicariam conteúdo armazenado e o colocariam nas bordas da Internet. Dado que uma grande fração do tráfego que flui pela Internet é conteúdo armazenado (páginas Web, MP3, vídeo), as CDNs podem aliviar significativamente as cargas de tráfego nos ISPs e nas interfaces de formação de pares (peering) entre ISPs. Além do mais, CDNs oferecem um serviço diferenciado para provedores de conteúdo: os que pagam pelos serviços de uma CDN podem entregar conteúdo mais rapidamente e com mais eficiência. Estudaremos CDNs mais adiante neste capítulo, na Seção 7.5.
- No caso de tráfego de fluxo contínuo ao vivo (como a transmissão de um evento esportivo) que está sendo enviado a milhões de usuários simultaneamente, podem ser disponibilizadas **redes *multicast* de sobreposição**. Uma rede *multicast* de sobreposição consiste em servidores espalhados por toda a rede do ISP (e, potencialmente, por toda a Internet). Esses servidores e os enlaces lógicos entre eles formam, coletivamente, uma rede de sobreposição que transmite tráfego *multicast* (veja Seção 4.7) da origem até milhões de usuários. Diferentemente do *multicast* IP, no qual a função *multicast* é manipulada por roteadores na camada do IP, em redes de sobreposição o *multicast* ocorre na camada de aplicação. Por exemplo, o hospedeiro de origem poderia enviar uma corrente a três servidores de sobreposição; cada um desses servidores pode transmitir a corrente para mais três servidores de sobreposição; o processo continua, criando, sobre a rede IP subjacente, uma árvore de distribuição com roteadores e hospedeiros. A transmissão *multicast* de tráfego muito requisitado, ao vivo, por redes de sobreposição pode reduzir a carga total de tráfego na Internet em relação à distribuição *unicast*.

Entre o campo da reserva de largura de banda e o campo da abordagem liberal, há ainda um terceiro — o campo dos serviços diferenciados (Diffserv). Esse campo quer fazer mudanças relativamente pequenas nas camadas de rede e de transporte e introduzir esquemas simples de determinação de preços e inspeção na borda da rede (isto é, na interface entre o usuário e o ISP do usuário). A idéia é introduzir um pequeno número de classes de tráfego (possivelmente apenas duas), designar cada datagrama a uma das classes, atribuir diferentes níveis de serviço nas filas de roteadores aos datagramas conforme suas classes e cobrar dos usuários segundo a classe de pacotes que os roteadores estão enviando pela rede. Abordaremos serviços diferenciados na Seção 7.8.

7.1.4 Compressão de áudio e vídeo

Antes que áudio e vídeo possam ser transmitidos por uma rede de computadores, eles devem ser digitalizados e comprimidos. A necessidade de digitalização é óbvia: redes de computadores transmitem bits; portanto, toda informação transmitida deve ser representada como uma seqüência de bits. A compressão é importante porque áudio e vídeo não comprimidos consomem uma quantidade extraordinária de capacidade de armazenamento e de largura de banda; remover as redundâncias inerentes aos sinais digitalizados de áudio e vídeo pode reduzir, em muitas vezes, a quantidade de dados que precisa ser armazenada e transmitida. Como exemplo, sem compressão, uma única imagem constituída de 1.024 × 1.024 pixels, com cada pixel codificado em 24 bits — 8 bits para cada uma das cores vermelha, verde e azul —, exige 3 Mbytes de armazenamento. Levaria 7 minutos para enviar essa imagem por um enlace de 64 kbps. Se a imagem for comprimida na modesta razão de compressão de 10:1, a necessidade de armazenamento será reduzida para 300 kbytes e o tempo de transmissão também diminuirá por um fator de 10.

O campo de compressão de áudio e vídeo é vasto. Áreas ativas de pesquisa nesse campo já existem há mais de 50 anos, e agora há literalmente centenas de técnicas e padrões populares para compressão tanto de áudio quanto de vídeo. A maioria das universidades tem cursos completos de compressão de áudio e vídeo e muitas vezes oferece um curso separado para cada um. Portanto, apresentamos aqui uma breve introdução de alto nível ao assunto.

Compressão de áudio na Internet

Um sinal analógico de áudio continuamente variável (que poderia emanar de voz ou de música) é normalmente convertido em um sinal digital como segue:

- O sinal analógico de áudio é primeiramente amostrado a alguma taxa fixa; por exemplo, 8.000 amostras por segundo. O valor de cada amostra é um número real arbitrário.
- Cada uma das amostras é então arredondada para um valor entre um número finito de valores. Essa operação é denominada **quantização**. O número de valores finitos — denominados valores de quantização — é tipicamente uma potência de 2, por exemplo, 256 valores de quantização.
- Cada um dos valores de quantização é representado por um número fixo de bits. Por exemplo, se houver 256 valores de quantização, então cada valor — e, portanto, cada amostra — será representado por 1 byte. Cada uma das amostras é convertida à sua representação por bits. As representações por bits de todas as amostras são, então, concatenadas em conjunto para formar a representação digitalizada do sinal.

Como exemplo, se um sinal de áudio for amostrado a uma taxa de 8.000 amostras por segundo e cada amostra for quantizada e representada por 8 bits, então o sinal digital resultante terá uma taxa de 64.000 bits por segundo. Esse sinal digital pode então ser reconvertido — isto é, decodificado — em um sinal analógico para ser reproduzido. Contudo, o sinal analógico decodificado é caracteristicamente diferente do sinal de áudio original. Aumentando a taxa de amostragem e o número de valores de quantização, o sinal decodificado pode se aproximar do sinal analógico original. Assim, há uma clara permuta entre a qualidade do sinal decodificado e as exigências de armazenamento e largura de banda do sinal digital.

A técnica básica de codificação que acabamos de descrever é denominada **modulação por codificação de pulso** (*pulse code modulation* — PCM). A codificação de voz freqüentemente usa PCM, com uma taxa de amostragem de 8.000 amostras por segundo e 8 bits por amostra, o que dá uma taxa de 64 kbps. O disco compacto de áudio (CD) também usa PCM, com taxa de amostragem de 44.100 amostras por segundo e 16 bits por amostra; isso dá uma taxa de 705,6 kbps para mono e 1,411 Mbps para estéreo.

Uma taxa de bits de 1,411 Mbps para música estéreo ultrapassa a maioria das taxas de acesso, e mesmo os 64 kbps para voz ultrapassam a taxa de acesso de um modem discado. Por essas razões, voz e música codificadas em PCM raramente são usadas na Internet. Em vez disso, são utilizadas técnicas de compressão

para reduzir a taxa de bits. Entre as técnicas de compressão populares para voz estão **GSM** (13 kbps), **G.729** (8 kbps) e **G.723.3** (6,4 e 5,3 kbps), e um grande número de técnicas proprietárias, incluindo as usadas pela RealNetworks. Uma técnica de compressão popular para música estéreo com qualidade próxima à do CD é a **MPEG 1 de camada 3**, mais conhecida como **MP3**. Codificadores MP3 normalmente comprimem para taxas de 96 kbps, 128 kbps ou 112 kbps e causam pouquíssima degradação de som. Quando um arquivo de MP3 é fragmentado em pedaços, cada pedaço ainda é reproduzível. Esse formato de arquivo sem cabeçalho permite que arquivos de música em MP3 sejam transmitidos pela Internet em fluxo contínuo (admitindo que a taxa de bits da reprodução e a velocidade da conexão com a Internet sejam compatíveis). O padrão de compressão MP3 é complexo — usa máscara psicoacústica, redução de redundância e buffer com reservatório de bits.

Compressão de vídeo na Internet

Um vídeo é uma seqüência de imagens que normalmente são apresentadas a uma taxa constante; por exemplo, 24 ou 30 imagens por segundo. Uma imagem não comprimida, codificada digitalmente, é constituída de um conjunto de pixels, e cada pixel é codificado por um número de bits para representar luminância e cor. Há dois tipos de redundância em vídeo, e ambos podem ser explorados para compressão. Redundância espacial é a redundância dentro de uma dada imagem. Por exemplo, uma imagem cuja maior parte é constituída de espaço em branco pode ser comprimida eficientemente. Redundância temporal reflete a repetição de imagens subseqüentes. Se, por exemplo, uma imagem e a imagem subseqüente forem exatamente as mesmas, não haverá razão para codificar novamente a imagem subseqüente; será mais eficiente simplesmente indicar, durante a codificação, que a imagem subseqüente é exatamente a mesma.

Os padrões de compressão MPEG estão entre as técnicas de compressão mais populares. Entre essas técnicas estão a **MPEG 1** para vídeo com qualidade de CD-ROM (1,5 Mbps), a **MPEG 2** para vídeo **DVD** de alta qualidade (3 a 6 Mbps) e a **MPEG 4** para compressão de vídeo orientada para objeto. O padrão MPEG tem muito do padrão JPEG para compressão de imagens. Os padrões **H.261** para compressão de vídeo são também muito populares na Internet. Além desses padrões, existem numerosos esquemas proprietários, entre eles o QuickTime da Apple e os codificadores da Real Networks. Leitores interessados em aprender mais sobre a codificação de áudio e vídeo podem consultar [Rao, 1996] e [Solari, 1997]. Um bom livro sobre rede multimídia em geral é [Crowcroft, 1999].

7.2 Áudio e vídeo de fluxo contínuo armazenados

Nos últimos anos, as aplicações de áudio e vídeo de fluxo contínuo tornaram-se populares e consumidoras significativas de largura de banda. Essa tendência provavelmente continuará por diversas razões. Em primeiro lugar, o custo de armazenamento em disco continua a diminuir rapidamente, dando espaço para arquivos multimídia ávidos de armazenamento. Hoje já estão disponíveis recursos de armazenamento da ordem de terabyes, capazes de conter milhares de vídeos MPEG 2. Em segundo lugar, melhorias na infra-estrutura da Internet — tais como acesso residencial de alta velocidade (isto é, modem a cabo e ADSL, como discutimos no Capítulo 1), técnicas de distribuição de conteúdo como caching e CDNs (veja Seção 7.5) e os novos protocolos de Internet orientados para QoS (veja as seções 7.6 a 7.9) — facilitarão demais a distribuição de vídeo e áudio armazenados. E, em terceiro lugar, há uma enorme demanda reprimida por vídeo de fluxo contínuo de alta qualidade sob demanda, uma aplicação que combina duas tecnologias de comunicação sedutoras — televisão e Web sob demanda.

Em áudio e vídeo de fluxo contínuo, clientes solicitam arquivos de áudio e vídeo comprimidos que residem em servidores. Como discutiremos nesta seção, esses servidores podem ser servidores Web comuns ou servidores de fluxo contínuo especiais configurados para a aplicação de áudio e vídeo de fluxo contínuo. A pedido do cliente, o servidor dirige a ele um arquivo de áudio/vídeo enviando o arquivo para uma porta. Ambas as portas, TCP e UDP, são usadas na prática. Antes de ser enviado para a rede, o arquivo de áudio/

vídeo é segmentado e os segmentos são caracteristicamente encapsulados com cabeçalhos especiais apropriados para o tráfego de áudio e vídeo. O RTP (*real-time protocol* — protocolo de tempo real), discutido na Seção 7.4, é um padrão de domínio público para o encapsulamento desses segmentos. Tão logo o arquivo de áudio/vídeo solicitado comece a chegar, o cliente começa a reproduzi-lo (normalmente) dentro de alguns segundos. A maior parte dos produtos existentes também permite interatividade com o usuário, por exemplo, pausa/reinício e saltos temporais dentro do arquivo de áudio/vídeo. Essa interatividade com o usuário também requer um protocolo para a interação cliente/servidor. O RTSP (*real-time streaming protocol* — protocolo de fluxo contínuo em tempo real), discutido no final desta seção, é um protocolo de domínio público que provê interatividade com o usuário.

Os usuários quase sempre requisitam reprodução de áudio e vídeo de fluxo contínuo por meio de um cliente Web (isto é, um browser). Mas, como a reprodução de áudio/vídeo não está integrada diretamente aos clientes Web atuais, é preciso uma **aplicação auxiliar** separada para reproduzir a apresentação. A aplicação auxiliar é denominada **transdutor** (mídia player); hoje, os transdutores mais populares são o RealPlayer, da RealNetworks, e o Windows Media Player, da Microsoft. O transdutor desempenha diversas funções, entre as quais:

- *Descompressão.* Áudio e vídeo quase sempre são comprimidos para economizar espaço de armazenamento em disco e largura de banda de rede. Um transdutor deve descomprimir áudio e vídeo enquanto são reproduzidos.
- *Eliminação da variação de atraso.* A variação de atraso do pacote é a variabilidade dos atrasos origem-destino dos pacotes dentro da mesma corrente de pacotes. Uma vez que áudio e vídeo têm de ser reproduzidos à mesma taxa em que foram gravados, o receptor colocará os pacotes recebidos em um buffer durante um curto período de tempo para eliminar essa variação. Examinaremos esse tópico detalhadamente na Seção 7.3.
- *Correção de erros.* Devido à imprevisibilidade do congestionamento na Internet, uma fração de pacotes na corrente de pacotes pode ser perdida. Se essa fração se tornar muito grande, a qualidade de áudio/vídeo percebida pelo usuário se tornará inaceitável. Para que isso não aconteça, muitos sistemas de fluxo contínuo tentam recuperar perdas (1) reconstruindo os pacotes perdidos por meio da transmissão de pacotes redundantes, (2) fazendo com que o cliente solicite explicitamente a retransmissão de pacotes perdidos ou (3) mascarando as perdas pela interpolação dos dados que faltam nas mensagens recebidas.

O transdutor tem uma interface gráfica de usuário com botões de controle, com a qual o usuário realmente interage. Essa interface normalmente tem controles de volume, teclas de pausa/reinício, controles deslizantes para fazer saltos temporais na seqüência de áudio/vídeo e assim por diante.

Podem ser usados programas plug-in para inserir a interface de usuário do transdutor dentro da janela do browser Web. Para fazer isso, o browser reserva espaço de tela na página corrente da Web e o transdutor fica incumbido de gerenciar o espaço de tela. Mas, quer apareça em janela separada, quer dentro da janela do browser (como um plug-in), o transdutor é um programa executado separadamente do browser.

7.2.1 Acesso a áudio e vídeo por meio de um servidor Web

Áudio e vídeo armazenados podem residir no servidor Web que entrega áudio/vídeo para o cliente sobre HTTP ou em um servidor de fluxo contínuo de áudio/vídeo que realiza essa entrega sobre outros protocolos que não o HTTP (protocolos que podem ser proprietários ou padrões abertos). Nesta subseção, examinaremos a entrega de áudio/vídeo a partir de um servidor Web; na subseção seguinte, abordaremos a entrega de áudio/vídeo a partir de um servidor de fluxo contínuo.

Considere, em primeiro lugar, o caso de áudio de fluxo contínuo. Quando um arquivo de áudio reside em um servidor Web, ele é um objeto comum dentro do sistema de arquivos do servidor, exatamente como os arquivos HTML e JPEG. Quando um usuário quer ouvir o arquivo de áudio, o hospedeiro do usuário

estabelece uma conexão TCP com o servidor Web e envia uma requisição HTTP para o objeto (veja a Seção 2.2). Ao receber a requisição, o servidor Web encapsula o arquivo de áudio em uma mensagem de resposta HTTP e devolve a mensagem de resposta para a conexão TCP. No caso do vídeo, essa operação pode ser um pouco mais complicada, porque a parte sonora e a parte visual do vídeo podem estar armazenadas em dois arquivos, isto é, podem ser dois objetos diferentes dentro do sistema de arquivo do servidor Web. Nesse caso, duas requisições HTTP separadas são enviadas ao servidor (por duas conexões TCP separadas no caso do HTTP/1.0) e os arquivos da parte sonora (áudio) e da parte visual (vídeo) chegam ao cliente em paralelo. É do cliente a incumbência de gerenciar a sincronização das duas correntes. Também é possível que o áudio e o vídeo estejam intercalados no mesmo arquivo, de modo que somente um objeto precise ser enviado ao cliente. Para simplificar a discussão, no caso de um vídeo propriamente dito, admitiremos que o áudio e o vídeo estejam contidos em um único arquivo.

Uma arquitetura simples para áudio e vídeo de fluxo contínuo está ilustrada na Figura 7.1 Nessa arquitetura:

- O processo do browser estabelece uma conexão TCP com o servidor Web e requisita o arquivo de áudio/vídeo com uma mensagem de requisição HTTP.
- O servidor Web envia ao browser o arquivo de áudio/vídeo dentro de uma mensagem de resposta HTTP.
- A linha de cabeçalho de tipo de conteúdo na mensagem de resposta HTTP indica uma codificação específica de áudio/vídeo. O browser cliente examina o tipo de conteúdo da mensagem de resposta, lança o transdutor associado e passa o arquivo ao transdutor.
- O transdutor, então, reproduz o arquivo de áudio/vídeo.

Embora seja muito simples, essa abordagem tem uma desvantagem importante: o transdutor (isto é, a aplicação auxiliar) tem de interagir com o servidor por intermédio de um browser Web, o que pode acarretar muitos problemas. Em particular, quando um browser é um intermediário, o objeto inteiro deve ser descarregado antes de o browser passá-lo a uma aplicação auxiliar. O atraso resultante antes do início da reprodução é tipicamente inaceitável para clipes de áudio/vídeo de tamanho moderado. Por essa razão, implementações de áudio/vídeo de fluxo contínuo em geral obrigam o servidor a enviar o arquivo de áudio/vídeo diretamente ao processo transdutor. Em outras palavras, é feita uma conexão direta entre o processo servidor e o processo transdutor. Como mostra a Figura 7.2, isso é feito pela utilização de um **metarquivo**, um arquivo que fornece informações (por exemplo, URL, tipo de codificação) sobre o arquivo de áudio/vídeo de fluxo contínuo a ser entregue.

Uma conexão TCP direta entre o servidor e o transdutor é obtida como segue:

1. O usuário clica sobre um hiperlink de um arquivo de áudio/vídeo.

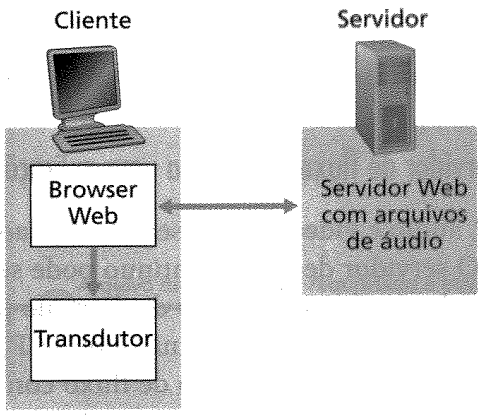


Figura 7.1 Uma implementação simples de áudio de fluxo contínuo

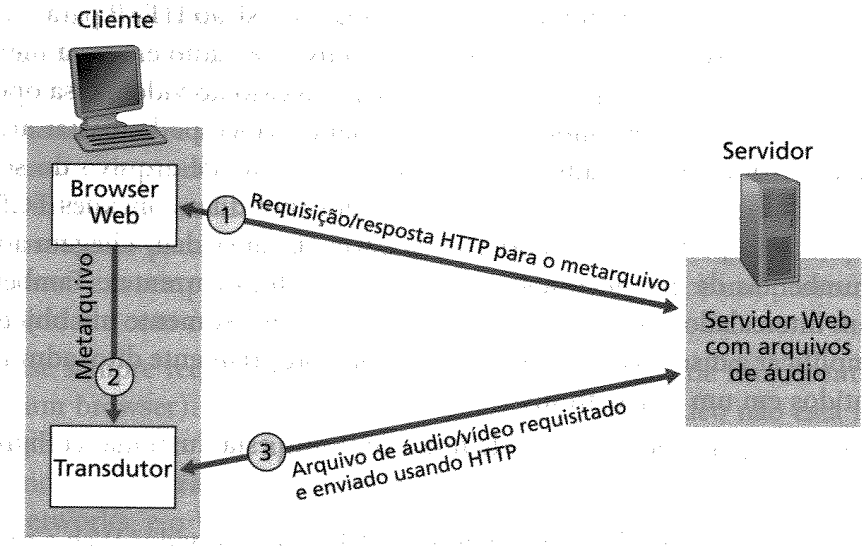


Figura 7.2 Servidor Web envia áudio/vídeo diretamente ao transdutor

2. O hiperlink não aponta diretamente para o arquivo de áudio/vídeo, mas para um metarquivo. O metarquivo contém o URL do arquivo de áudio/vídeo. A mensagem de resposta HTTP que encapsula o metarquivo contém uma linha de cabeçalho de tipo de conteúdo que indica a aplicação de áudio/vídeo específica.
3. O browser cliente examina a linha de cabeçalho de tipo de conteúdo da mensagem de resposta, lança o transdutor associado e passa todo o corpo da mensagem de resposta (isto é, o metarquivo) para o transdutor.
4. O transdutor estabelece uma conexão TCP diretamente com o servidor HTTP e envia uma mensagem de requisição HTTP do arquivo de áudio/vídeo para a conexão TCP.
5. O arquivo de áudio/vídeo é enviado ao transdutor dentro de uma mensagem de resposta HTTP. O transdutor exibe o arquivo de áudio/vídeo de fluxo contínuo.

A importância da etapa intermediária de aquisição do metarquivo é clara. Ao verificar o tipo de conteúdo do arquivo, o browser pode lançar o transdutor adequado e, dessa maneira, fazer com que o transdutor contate diretamente o servidor.

Acabamos de aprender como um metarquivo pode permitir que um transdutor dialogue diretamente com um servidor Web que armazena um arquivo de áudio/vídeo. Ainda assim, muitas empresas que vendem produtos que entregam áudio/vídeo de fluxo contínuo não recomendam a arquitetura que acabamos de descrever. Isso porque ela obriga o transdutor a se comunicar com o servidor por HTTP e, portanto, também por TCP. O HTTP é muitas vezes considerado insuficientemente rico para permitir interação satisfatória do usuário com o servidor; em particular, não é fácil para o HTTP permitir que um usuário envie ao servidor (por meio do transdutor) comandos de pausa/reinício, avanço rápido e salto temporal.

7.2.2 Envio de multimídia de um servidor de fluxo contínuo a uma aplicação auxiliar

Para evitar o HTTP e/ou o TCP, áudio e vídeo podem ser armazenados em um servidor de fluxo contínuo e enviados dele ao transdutor. O servidor de fluxo contínuo pode ser proprietário, tal como os comercializados pela RealNetworks e pela Microsoft, ou pode ser de domínio público. Com um servidor de fluxo contínuo, áudio e vídeo podem ser enviados por UDP (e não por TCP) usando protocolos de camada de rede que podem ser melhor configurados para áudio/vídeo de fluxo contínuo do que o HTTP.

Essa arquitetura requer dois servidores, como mostra a Figura 7.3. Um dos servidores, o servidor HTTP, serve páginas Web (incluindo metarquivos). O segundo servidor, o **servidor de fluxo contínuo**, serve os

arquivos de áudio/vídeo. Os dois servidores podem rodar no mesmo sistema final ou em dois sistemas finais distintos. As etapas dessa arquitetura são semelhantes às descritas na subseção anterior. Contudo, nesse caso o transdutor requisita o arquivo de um servidor de fluxo contínuo, e não de um servidor Web, e o transdutor e o servidor de fluxo contínuo podem interagir usando seus próprios protocolos. Esses protocolos permitem uma interação mais rica do usuário com o fluxo de áudio/vídeo.

Na arquitetura da Figura 7.3 há muitas opções para a entrega de áudio/vídeo do servidor de fluxo contínuo ao transdutor. Uma lista parcial de opções é dada a seguir:

1. O áudio/vídeo é enviado por UDP a uma taxa constante igual à taxa de reprodução do receptor (que é a taxa codificada para o áudio/vídeo). Por exemplo, se o áudio for comprimido usando GSM a uma taxa de 13 kbps, então o servidor transmitirá o arquivo comprimido de áudio a uma taxa de 13 kbps. Tão logo receba o áudio/vídeo comprimido da rede, o cliente o descomprime e o reproduz.
2. É igual à primeira opção, mas o transdutor atrasa a reprodução de 2 a 5 segundos para eliminar a variação de atraso induzida pela rede. O cliente realiza essa tarefa colocando a mídia comprimida que recebe da rede em um **buffer cliente**, como mostra a Figura 7.4. Tão logo tenha pré-capturado alguns poucos segundos de mídia, o cliente começa a descarregar o buffer. Para essa opção, e também para a anterior, a taxa de preenchimento $x(t)$ é igual à taxa de saída d, exceto quando há perda de pacotes, caso em que $x(t)$ é momentaneamente menor que d.
3. A mídia é enviada por TCP. O servidor lança o arquivo de mídia para dentro da porta TCP o mais rápido possível; o cliente (isto é, o transdutor) lê da porta TCP o mais rápido possível e coloca o vídeo comprimido no buffer do transdutor. Após um atraso inicial de 2 a 5 segundos, o transdutor lê de seu buffer a uma taxa d e repassa a mídia comprimida para descompressão e reprodução. Como o TCP retransmite pacotes perdidos, ele tem o potencial de fornecer melhor qualidade de som do que o UDP. Por outro lado, a taxa de preenchimento $x(t)$ varia com o tempo devido ao controle de congestionamento e ao controle de fluxo de janela do TCP. De fato, após perda de pacotes, o controle de congestionamento TCP pode reduzir a taxa instantânea a valores menores do que d por longos períodos de tempo. Isso pode esvaziar o buffer cliente (um processo conhecido como **inanição**) e introduzir pausas indesejáveis na saída de uma corrente de áudio/vídeo no cliente.

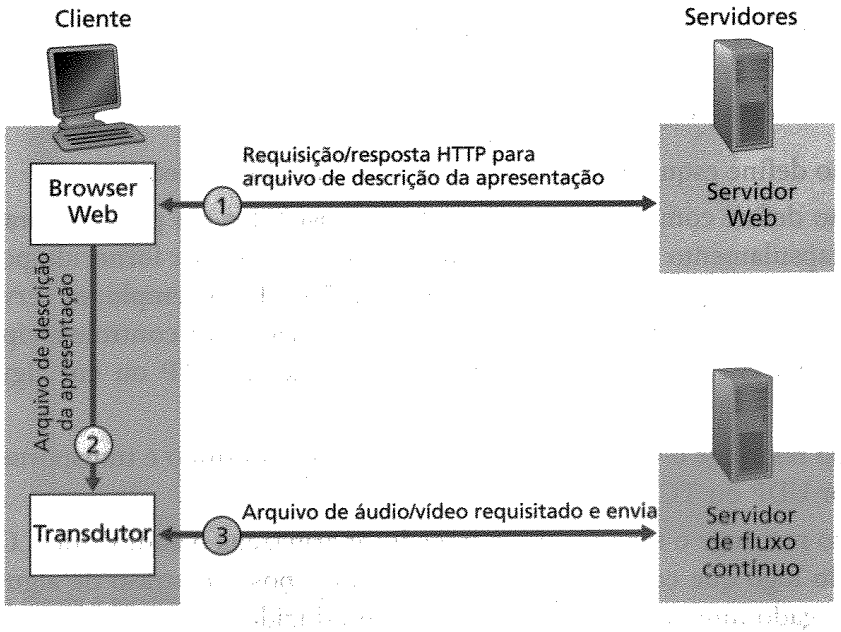


Figura 7.3 Envio de fluxo contínuo de um servidor de fluxo contínuo para um transdutor

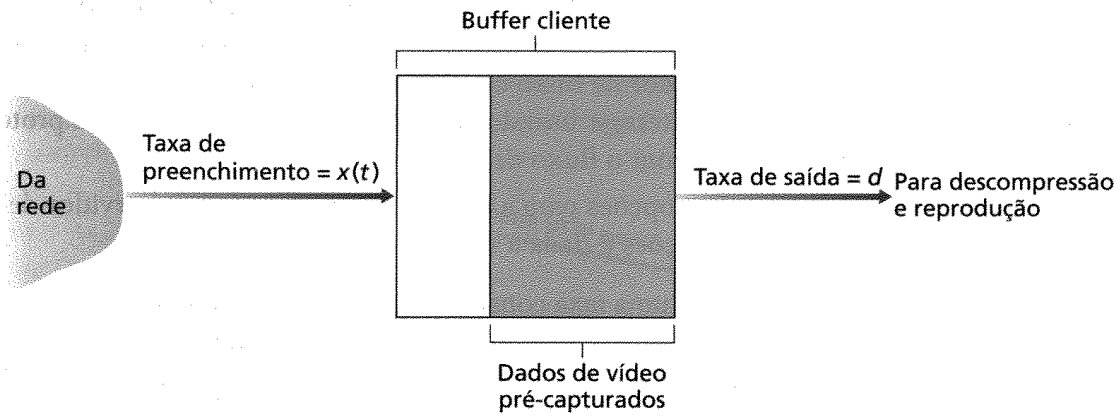


Figura 7.4 Buffer cliente preenchido a uma taxa x(t) e esvaziado a uma taxa d

Para a terceira opção, o comportamento de $x(t)$ dependerá muitíssimo do tamanho do buffer cliente (que não deve ser confundido com o buffer de recepção TCP). Se o buffer for grande o suficiente para conter todo o arquivo de mídia (possivelmente por armazenagem em disco), então o TCP fará uso de toda a largura de banda instantânea disponível para a conexão, de modo que $x(t)$ pode se tornar muito maior do que d. Se $x(t)$ permanecer muito maior do que d por longos períodos de tempo, então uma grande porção da mídia será pré-capturada para dentro do cliente e uma subseqüente inanição do cliente será pouco provável. Se, por outro lado, o buffer cliente for pequeno, então $x(t)$ flutuará ao redor da taxa de saída d. O risco de inanição do cliente é maior nesse caso.

7.2.3 RTSP (protocolo de fluxo contínuo em tempo real)

Muitos usuários de multimídia pela Internet (em especial aqueles que cresceram com um controle remoto de TV na mão) vão querer controlar a reprodução de mídia de taxa constante fazendo pausas na reprodução, reposicionando a reprodução em um ponto do tempo futuro ou passado, avançando ou atrasando a reprodução em modo rápido e assim por diante. Essa funcionalidade é semelhante à que um aparelho de DVD oferece ao usuário quando este assiste a um DVD ou à que oferece um aparelho de CD para ouvir a música gravada no CD. Para permitir que um usuário controle a reprodução, o transdutor e o servidor precisam de um protocolo para trocar informações de controle de reprodução. O RTSP, definido no RFC 2326, é esse protocolo.

Antes de entrar nos detalhes do RTSP, vamos apresentar o que o RTSP não faz:

- O RTSP não define esquemas de compressão para áudio e vídeo.
- O RTSP não define como áudio e vídeo são encapsulados em pacotes para transmissão por uma rede; o encapsulamento para mídia de fluxo contínuo pode ser fornecido por RTP ou por um protocolo proprietário. (O RTP é discutido na Seção 7.4.) Por exemplo, os servidores e os transdutores da RealNetworks usam o RTSP para enviar informações de controle reciprocamente, mas a corrente de mídia em si pode ser encapsulada em pacotes RTP ou em algum formato de dados proprietário.
- O RTSP não restringe o modo como a mídia de fluxo contínuo é transportada; ela pode ser transportada por UDP ou TCP.
- O RTSP não restringe o modo como o transdutor armazena o áudio/vídeo. O áudio/vídeo pode ser reproduzido logo que começa a chegar ao cliente, após um atraso de alguns segundos, ou pode ser descarregado integralmente antes de ser reproduzido.

Portanto, se o RTSP não executa nenhuma dessas funções, o que ele faz? O RTSP permite que um transdutor controle a transmissão de uma corrente de mídia. Como mencionamos há pouco, as ações de controle incluem pausa/reinício, reposicionamento da reprodução, avanço e retrocesso rápidos. O RTSP é um protocolo 'fora da banda'. Em particular, as mensagens RTSP são enviadas fora da banda, ao passo que a corrente de mídia, cuja estrutura de pacote não é definida pelo RTSP, é considerada 'dentro da banda'. Mensagens RTSP usam o número de porta 544; a corrente de mídia utiliza um número diferente. A especificação do RTSP [RFC 2326] permite que as mensagens RTSP sejam enviadas por TCP ou UDP.

Lembre-se de que o protocolo de transferência de arquivos (FTP — [Seção 2.3]) também usa a noção de fora da banda. Em particular, o FTP usa dois pares de portas cliente-servidor, cada par com seu próprio número de porta: um par de portas cliente-servidor suporta uma conexão TCP que transporta informações de controle; o outro par de portas cliente-servidor suporta uma conexão TCP que transporta o arquivo propriamente dito. O canal RTSP é semelhante, em muitos aspectos, ao canal de controle do FTP.

Vamos agora percorrer um exemplo simples de RTSP, que é ilustrado na Figura 7.5. O browser Web primeiramente solicita um arquivo de descrição da apresentação a um servidor Web. Esse arquivo pode conter referências a diversos arquivos de mídia de taxa constante, bem como diretrizes para sincronização dos arquivos de mídia de taxa contínua. Cada referência a um arquivo de mídia de taxa contínua começa com o método URL, rtsp://. Fornecemos a seguir uma amostra de arquivo de apresentação que foi adaptado de [Schulzrinne, 1997]. Nessa apresentação, uma corrente de áudio e uma de vídeo são reproduzidas em paralelo e com voz sincronizada (como parte do mesmo grupo). Para a corrente de áudio, o transdutor pode escolher (comutar) entre duas gravações de áudio, uma de baixa fidelidade e outra de alta fidelidade. (O formato do arquivo é semelhante ao SMIL [SMIL, 2004],* que é usado por muitos produtos de taxa contínua para definir apresentações sincronizadas de multimídia.)

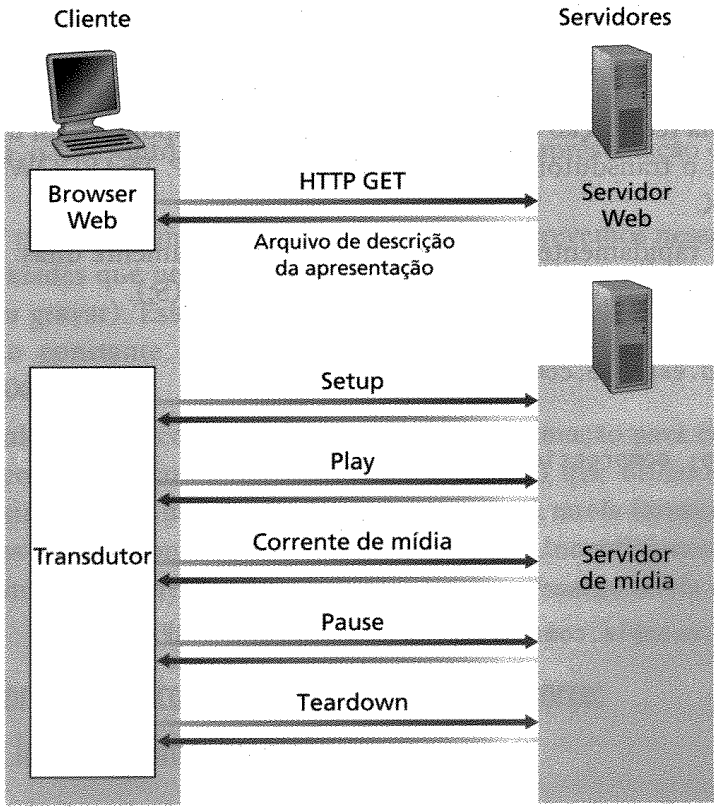


Figura 7.5 Interação entre cliente e servidor usando RTSP

* Trata-se de um padrão definido pelo W3C. (N. do R.T.)

```
<title>Twister<title>
<session>
  <group language=en lipsync>
    <switch>
      <track type=audio
        e="PCMU/8000/1"
        src = "rtsp://audio.example.com/twister/audio.en/lofi">
      <track type=audio
        e="DVI4/16000/2" pt="90 DVI4/8000/1"
        src="rtsp://audio.example.com/twister/audio.en/hifi">
        </switch>
      <track type="video/jpeg"
        src="rtsp://video.example.com/twister/video">
  </group>
</session>
```

O servidor Web encapsula o arquivo de descrição da apresentação em uma mensagem de resposta HTTP e envia a mensagem ao browser. Quando recebe a mensagem de resposta HTTP, o browser invoca um transdutor (isto é, uma aplicação auxiliar) com base no campo de tipo de conteúdo da mensagem. O arquivo de descrição da apresentação contém referências às correntes de mídia, usando o método URL rtsp://, como mostrado no exemplo anterior. Como ilustra a Figura 7.5, o transdutor e o servidor então enviam um ao outro uma série de mensagens RTSP. O transdutor envia uma requisição RTSP SETUP e o servidor responde com uma mensagem RTSP OK. O transdutor envia uma requisição RTSP PLAY, digamos, para áudio de baixa fidelidade e o servidor responde com uma mensagem RTSP OK. Nesse ponto, o servidor de fluxo contínuo bombeia o áudio de baixa fidelidade para dentro de seu próprio canal 'dentro da banda'. Mais tarde, o transdutor envia uma requisição RTSP PAUSE e o servidor responde com uma mensagem RTSP OK. Quando o usuário termina, o transdutor envia uma requisição RTSP TEARDOWN e o servidor confirma com uma resposta RTSP OK.

Agora vamos examinar rapidamente as mensagens RTSP propriamente ditas. Segue um exemplo simplificado de uma sessão RTSP entre um cliente (C:) e um remetente (S:).

```
C: SETUP rtsp://audio.example.com/twister/audio RTSP/1.0
   Cseq: 1
   Transport: rtp/udp; compression; port=3056; mode=PLAY
S: RTSP/1.0 200 OK
   Cseq: 1
   Session: 4231
C: PLAY rtsp://audio.example.com/twister/audio.en/lofi RTSP/1.0
   Range: npt=0-
   Cseq: 2
   Session: 4231
S: RTSP/1.0 200 OK
   Cseq:2
   Session:4231
C: PAUSE rtsp://audio.example.com/twister/audio.en/lofi RTSP/1.0
```

```
    Range: npt=37
    Cseq: 3
    Session: 4231
S: RTSP/1.0 200 OK
    Cseq: 3
    Session: 4231
C: TEARDOWN rtsp://audio.example.com/twister/audio.en/lofi RTSP/1.0
    Cseq: 4
    Session: 4231
S: RTSP/1.0 200 OK
    Cseq: 4
    Session: 4231
```

É interessante notar as similaridades entre HTTP e RTSP. Todas as mensagens de requisição e resposta são em texto ASCII, o cliente emprega métodos padronizados (SETUP, PLAY, PAUSE e assim por diante) e o servidor responde com códigos padronizados de resposta. Uma diferença importante, contudo, é que o servidor RTSP monitora o estado do cliente para cada sessão RTSP em curso. Por exemplo, o servidor monitora se o cliente está em um estado de inicialização, um estado de reprodução (play) ou um estado de pausa (ver a tarefa de programação ao final deste capítulo). Os números de sessão e de seqüência, que fazem parte de cada requisição e resposta RTSP, ajudam o servidor a monitorar o estado da sessão. O número da sessão é fixo durante toda a sessão; o cliente incrementa o número de seqüência cada vez que envia uma nova mensagem; o servidor devolve um eco do número de sessão e do número de seqüência corrente.

Como o exemplo demonstra, o cliente inicia a sessão com a requisição SETUP, fornecendo o URL do arquivo que deverá ser transmitido e a versão do RTSP. A mensagem de estabelecimento (SETUP) inclui o número de porta do cliente para o qual a mídia deve ser enviada. Essa mensagem também indica que a mídia deve ser enviada por UDP usando o protocolo de empacotamento RTP (que será discutido na Seção 7.4). Note que, nesse exemplo, o transdutor escolheu não reproduzir a apresentação completa, mas apenas a porção de baixa fidelidade.

Na verdade, o RTSP pode fazer muito mais do que o que descrevemos nessa breve introdução. Em particular, o RTSP tem facilidades que permitem que os clientes enviem um fluxo contínuo na direção do servidor (por exemplo, para gravar). Esse protocolo foi adotado pela RealNetworks, uma das líderes no setor de áudio/vídeo de fluxo constante. Henning Schulzrinne disponibiliza uma página Web sobre RTSP [Schulzrinne — RTSP, 2004].

Ao final deste capítulo, você encontrará uma tarefa de programação para criar um sistema de reprodução de vídeo em fluxo contínuo (para ambos, servidor e cliente) que utiliza o RTSP. Essa tarefa envolve escrever código que constrói e envia mensagens RTSP no cliente. A tarefa fornece o código do servidor RTSP, que analisa as mensagens RTSP e constrói respostas apropriadas. Incentivamos os leitores interessados em aprofundar seus conhecimentos sobre RTSP a executar essa interessante tarefa de programação.

7.3 Fazendo o melhor possível com o serviço de melhor esforço: um telefone por Internet como exemplo

O protocolo de camada de rede da Internet, IP, presta um serviço de melhor esforço. Isso quer dizer que o serviço faz seu melhor esforço para transportar cada datagrama da fonte ao destino o mais rápido possível. Contudo, o serviço de melhor esforço nada promete quanto à dimensão do atraso fim-a-fim para um pacote individual ou quanto à dimensão da variação do atraso ou da perda de pacotes dentro da corrente de pacotes. A falta de garantias quanto ao atraso e à variação do atraso propõe desafios significativos ao proje-

to de aplicações de multimídia em tempo real, como telefone por Internet e videoconferência em tempo real, que são muito sensíveis ao atraso, à variação do atraso e à perda de pacotes. Felizmente, os projetistas dessas aplicações podem introduzir diversos mecanismos úteis que conseguem preservar a boa qualidade do áudio e do vídeo contanto que o atraso, a variação do atraso e a perda não sejam excessivos. Nesta seção, examinaremos alguns desses mecanismos. Para manter a discussão no terreno concreto, discutiremos esses mecanismos no contexto de uma **aplicação de telefone por Internet**, descrita a seguir. A situação é semelhante para aplicações de videoconferência em tempo real [Bolot, 1994].

O interlocutor de nossa aplicação de telefone por Internet gera um sinal de áudio constituído de rajadas de voz alternadas com períodos de silêncio. Para conservar largura de banda, nossa aplicação de telefone por Internet gera pacotes somente durante as rajadas de voz. Durante uma rajada de voz, o remetente gera bytes a uma taxa de 8.000 bytes por segundo e, a cada 20 milissegundos, reúne os bytes em porções. Assim, o número de bytes em uma porção é (20 ms) · (8.000 bytes/s) = 160 bytes. Um cabeçalho especial, cujo conteúdo é discutido a seguir, é agregado a cada porção. A porção e seu cabeçalho são encapsulados em um segmento UDP por meio da chamada à interface da porta. Assim, durante uma rajada de voz, um datagrama UDP é enviado a cada 20 milissegundos.

Se cada pacote conseguir chegar ao receptor e se tiver um atraso fim-a-fim pequeno e constante, então pacotes chegarão ao receptor periodicamente, a cada 20 milissegundos, durante uma rajada de voz. Nessas condições ideais, o receptor pode simplesmente reproduzir cada porção assim que ela chega. Mas infelizmente alguns pacotes podem ser perdidos e a maioria dos pacotes não sofrerá o mesmo atraso fim-a-fim, mesmo quando a Internet estiver apenas levemente congestionada. Por essa razão, o receptor deve tomar mais cuidado ao determinar (1) quando reproduzir uma rajada e (2) o que fazer com uma porção perdida.

7.3.1 As limitações de um serviço de melhor esforço

Mencionamos que o serviço de melhor esforço pode acarretar perda de pacotes, excessivo atraso fim-a-fim e variação de atraso. Vamos examinar esses assuntos com maiores detalhes.

Perda de pacotes

Considere um dos segmentos UDP gerados por nossa aplicação de telefone por Internet. O segmento UDP é encapsulado em um datagrama IP. Enquanto o datagrama vagueia pela rede, ele passa por buffers (isto é, por filas) nos roteadores, para poder alcançar os enlaces de saída. É possível que um ou mais dos buffers na rota entre o remetente e o destinatário estejam lotados e não possam aceitar o datagrama IP. Nesse caso, o datagrama IP será descartado e nunca chegará à aplicação receptora.

A perda pode ser eliminada enviando os pacotes por TCP, em vez de por UDP. Lembre-se de que o TCP retransmite pacotes que não chegam ao destino. Contudo, os mecanismos de retransmissão freqüentemente são considerados inaceitáveis para aplicações interativas de áudio em tempo real, como o telefone por Internet, porque aumentam o atraso fim-a-fim [Bolot, 1996]. Além disso, devido ao controle de congestionamento do TCP, após a perda de pacote a taxa de transmissão no remetente pode ser reduzida a uma taxa mais baixa do que a taxa de reprodução no receptor, o que pode causar um forte impacto sobre a inteligibilidade da voz no receptor. Por essas razões, quase todas as aplicações de telefone por Internet existentes executam em UDP e não se preocupam em retransmitir pacotes perdidos.

Mas perder pacotes não é necessariamente tão grave quanto se possa imaginar. Na verdade, taxas de perda de pacotes entre 1 e 20 por cento podem ser toleradas, dependendo de como a voz é codificada e transmitida e de como a perda é ocultada no receptor. Por exemplo, o mecanismo de correção de erros de repasse (*forward error correction* — FEC) pode ajudar a ocultar a perda de pacotes. Veremos mais adiante que, com a FEC, a informação redundante é transmitida junto com a informação original, de modo que parte dos dados originais perdidos pode ser recuperada da informação redundante. Não obstante, se um ou mais dos enlaces entre remetente e receptor estiverem fortemente congestionados e a perda de pacotes exceder 10–20 por cento (embora essas taxas raramente sejam observadas em redes bem provisionadas

[Boutremans, 2002]), então, na realidade, nada poderá ser feito para conseguir uma qualidade de som aceitável. Assim, fica claro que o serviço de melhor esforço tem suas limitações.

Atraso fim-a-fim

Atraso fim-a-fim é o acúmulo de atrasos de processamento, de transmissão e de formação de filas nos roteadores; atrasos de propagação nos enlaces e atrasos de processamento em sistemas finais. Para as aplicações de áudio altamente interativas, como o telefone por Internet, atrasos fim-a-fim menores do que 150 milissegundos não são percebidos pelo ouvido humano; atrasos entre 150 e 400 milissegundos podem ser aceitáveis, mas não são o ideal, e atrasos que excedem 400 milissegundos podem atrapalhar seriamente a interatividade em conversações por voz. O lado receptor de uma aplicação de telefone por Internet em geral desconsiderará quaisquer pacotes cujos atrasos ultrapassarem um determinado patamar, por exemplo, mais do que 400 milissegundos. Assim, os pacotes cujos atrasos ultrapassem o patamar são efetivamente perdidos.

Variação de atraso

Um componente crucial do atraso fim-a-fim são os atrasos aleatórios de fila nos roteadores. Por causa desses atrasos variáveis dentro da rede, o tempo decorrido entre o momento em que um pacote é gerado na fonte e o momento em que é recebido no destinatário pode variar de pacote para pacote. Esse fenômeno é denominado **variação de atraso**.

Como exemplo, considere dois pacotes consecutivos dentro de uma rajada de voz em nossa aplicação de telefone por Internet. O remetente envia o segundo pacote 20 milissegundos após ter enviado o primeiro. Mas, no receptor, o espaçamento entre esses pacotes pode ficar maior do que 20 milissegundos. Para entender, suponha que o primeiro pacote chegue a uma fila de roteador quase vazia, mas que, exatamente antes de o segundo pacote chegar à fila, um grande número de pacotes vindos de outras fontes chegue à mesma fila. Como o primeiro pacote sofre um pequeno atraso de fila e o segundo pacote sofre um grande atraso de fila nesse roteador, o primeiro e o segundo pacotes ficam espaçados em mais de 20 milissegundos. O espaçamento entre pacotes consecutivos também pode ficar menor do que 20 milissegundos. Para entender, considere novamente dois pacotes consecutivos dentro de uma rajada de voz. Suponha que o primeiro pacote se junte ao final de uma fila com um grande número de pacotes e que o segundo pacote chegue à fila antes dos pacotes de outras fontes. Nesse caso, nossos dois pacotes se encontram um exatamente atrás do outro na fila. Se o tempo que leva para transmitir um pacote no enlace de saída do roteador for menor do que 20 milissegundos, então o primeiro e o segundo pacotes ficarão espaçados por menos de 20 milissegundos.

A situação é análoga a dirigir carros em rodovias. Suponha que você e um amigo seu, cada um dirigindo seu próprio carro, estejam viajando de San Diego a Phoenix, tenham um estilo semelhante de dirigir e mantenham a velocidade de 100 km/h, se o tráfego existente permitir. Por fim, suponha que seu amigo parta uma hora antes de você. Então, dependendo do tráfego, você pode chegar a Phoenix mais ou menos uma hora depois de seu amigo.

Se o receptor ignorar a presença de variação de atraso e reproduzir as porções assim que elas chegam, então a qualidade de áudio resultante poderá facilmente se tornar ininteligível no receptor. Felizmente, a variação de atraso freqüentemente pode ser eliminada com a utilização de **números de seqüência**, **marcas de tempo** e **atraso de reprodução**, como discutido a seguir.

7.3.2 Eliminação da variação de atraso no receptor para áudio

Para uma aplicação de voz como telefone por Internet ou áudio sob demanda, o receptor deve tentar prover reprodução sincronizada de porções de voz na presença de variação de atraso aleatório, o que normalmente é feito combinando os três mecanismos seguintes:

- *Preceder cada porção com um* ***número de seqüência***. O remetente aumenta em um o número de seqüência para cada um dos pacotes que gera.

- *Preceder cada porção com uma* ***marca de tempo***. O remetente marca cada porção com o instante em que ela foi gerada.
- ***Atrasar a reprodução*** *de porções no receptor.* O atraso da reprodução das porções de áudio recebidas deve ser suficientemente longo para que a maioria dos pacotes seja recebida antes de seus tempos de reprodução programados. O atraso da reprodução pode ser fixado para todo o período de duração da sessão de áudio ou pode variar adaptativamente durante o período útil da sessão. Os pacotes que não chegarem antes de seu tempo de reprodução programado serão considerados perdidos e esquecidos; como observado antes, o receptor pode usar algum tipo de interpolação de voz para tentar disfarçar a perda.

Discutiremos agora como esses três mecanismos, quando combinados, podem atenuar ou até eliminar os efeitos da variação de atraso. Examinaremos duas estratégias de reprodução: atraso de reprodução fixo e atraso de reprodução adaptativo.

Atraso de reprodução fixo

Com a estratégia do atraso fixo, o receptor tenta reproduzir cada porção exatamente q milissegundos após a porção ter sido gerada. Assim, se a marca de tempo de uma porção for t, o receptor reproduz a porção no tempo $t + q$, admitindo-se que a porção já tenha chegado naquele instante. Os pacotes que chegam após seus tempos de reprodução programados são descartados e considerados perdidos.

Qual é uma boa escolha para q? Telefone por Internet pode suportar atrasos de até cerca de 400 milissegundos, embora com valores menores que q a experiência interativa resultante seja mais satisfatória. Por outro lado, caso se escolha um q muito menor do que 400 milissegundos, muitos pacotes podem perder seus tempos de reprodução programados devido à variação de atraso induzida pela rede. Em termos gerais, se grandes variações no atraso fim-a-fim forem características, será preferível usar um q grande; por outro lado, se o atraso for pequeno e as variações no atraso também forem pequenas, será preferível usar um q pequeno, talvez menor do que 150 milissegundos.

O compromisso entre o atraso de reprodução e a perda de pacotes é ilustrado na Figura 7.6. A figura mostra os tempos em que os pacotes são gerados e reproduzidos para uma única rajada de voz. Dois atrasos de reprodução iniciais distintos são considerados. Como mostram os degraus mais à esquerda do gráfico, o remetente gera pacotes a intervalos regulares — digamos, a cada 20 milissegundos. O primeiro pacote dessa rajada de voz é recebido no tempo r. Como mostra a figura, as chegadas dos pacotes subseqüentes não são espaçadas regularmente devido à variação de atraso da rede.

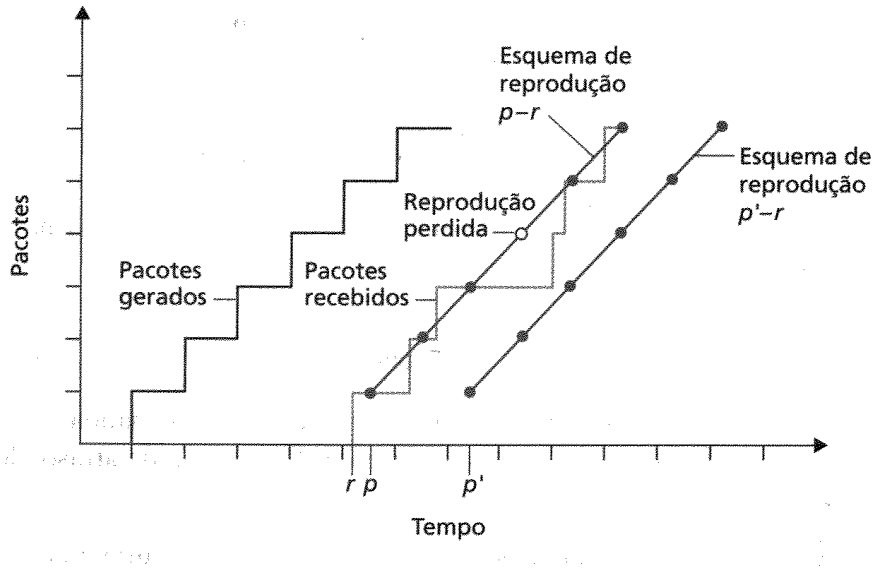


Figura 7.6 Perda de pacotes para diferentes atrasos de reprodução fixos

Para o primeiro esquema de reprodução, o atraso inicial de reprodução está estabelecido em $p - r$. Com esse esquema, o quarto pacote não chega dentro de seu atraso de reprodução programado e o receptor o considera perdido. Pelo segundo esquema de reprodução, o atraso inicial de reprodução fixo está estabelecido em $p' - r$. Por esse esquema, todos os pacotes chegam antes de seu tempo de reprodução e, portanto, não há perda.

Atraso de reprodução adaptativo

O exemplo anterior demonstra um importante compromisso entre atraso e perda que surge ao se projetar uma estratégia de reprodução com atrasos de reprodução fixos. Ao se decidir por um atraso inicial de reprodução grande, a maioria dos pacotes cumprirá suas programações e, portanto, a perda será insignificante; contudo, para serviços interativos como telefone por Internet, atrasos longos podem se tornar incômodos, se não intoleráveis. Idealmente, gostaríamos que o atraso de reprodução fosse minimizado, mas que mantivesse a limitação de perda abaixo de uns poucos por cento.

A maneira natural de tratar esse compromisso é estimar o atraso de rede e a variância de atraso de rede e ajustar o atraso de reprodução de acordo com o resultado, no início de cada rajada de voz. Esse ajuste adaptativo de atrasos de reprodução no início das rajadas de voz fará com que os períodos de silêncio no receptor sejam comprimidos e alongados; contudo, compressão e alongamento de silêncio em pequenas quantidades não são percebidos durante a fala.

Tomando [Ramjee, 1994] como base, descrevemos agora um algoritmo genérico que o receptor pode usar para ajustar seus atrasos de reprodução adaptativamente. Para tal, consideremos:

t_i = marca de tempo do i-ésimo pacote = o instante em que o pacote foi gerado pelo remetente

r_i = o instante em que o pacote i é recebido pelo receptor

p_i = o instante em que o pacote i é reproduzido no receptor

O atraso de rede fim-a-fim do i-ésimo pacote é $r_i - t_i$. Devido à variação de atraso da rede, esse atraso variará de pacote a pacote. Seja d_i a estimativa do atraso de rede médio para a recepção do i-ésimo pacote. Essa estimativa é obtida das marcas de tempo como segue:

$$d_i = (1 - u)\, d_{i-1} + u(r_i - t_i),$$

onde u é uma constante fixa (por exemplo, $u = 0{,}01$). Assim, d_i é uma média ajustada dos atrasos de rede observados $r_1 - t_1$, ..., $r_i - t_i$. A estimativa atribui maior peso aos atrasos de rede verificados mais recentemente do que aos atrasos de rede ocorridos no passado distante. Esse modelo de estimativa não deve ser tão fora do comum; uma idéia semelhante é usada para estimar os tempos de viagem de ida e volta no TCP, como discutido no Capítulo 3. Seja v_i uma estimativa do desvio médio do atraso em relação ao atraso médio estimado. A estimativa também é obtida com base nas marcas de tempo:

$$v_i = (1 - u)\, v_{i-1} + u|r_i - t_i - d_i|.$$

As estimativas d_i e v_i são calculadas para todos os pacotes recebidos, embora somente sejam usadas para determinar o ponto de reprodução para o primeiro pacote em qualquer rajada de voz.

Uma vez calculadas essas estimativas, o receptor emprega o seguinte algoritmo para a reprodução de pacotes. Se o pacote i for o primeiro pacote de uma rajada de voz, seu tempo de reprodução p_i será computado como:

$$p_i = t_i + d_i + Kv_i,$$

onde K é uma constante positiva (por exemplo, $K = 4$). A finalidade do termo Kv_i é estabelecer o tempo de reprodução em um futuro suficientemente longínquo, de modo que apenas uma pequena fração dos pacotes que estão chegando na rajada de voz seja perdida devido às chegadas atrasadas. O ponto de reprodução para qualquer pacote subseqüente em uma rajada de voz é computado como um deslocamento em relação ao ponto no tempo em que o primeiro pacote da rajada de voz foi reproduzido. Em particular, seja

$$q_i = p_i - t_i$$

a duração do tempo decorrido desde a geração do primeiro pacote da rajada de voz até sua reprodução. Se o pacote j também pertencer a essa rajada de voz, ele será reproduzido no tempo

$$p_j = t_j + q_i.$$

O algoritmo que acabamos de descrever tem perfeito sentido admitindo-se que o receptor possa dizer se um pacote é o primeiro pacote na rajada de voz. Se não houver perda de pacotes, então o receptor poderá determinar se o pacote i é o primeiro pacote da rajada de voz comparando a marca de tempo do i-ésimo pacote com a marca de tempo do pacote de ordem $(i - 1)$. Realmente, se $t_i - t_{i-1} > 20$ milissegundos, então o receptor sabe que o i-ésimo pacote inicia uma nova rajada de voz. Mas suponha agora que haja uma perda de pacote ocasional. Nesse caso, dois pacotes sucessivos recebidos no destino podem ter marcas de tempo que diferem de mais de 20 milissegundos quando os dois pacotes pertencem à mesma rajada de voz. E é nesse caso que os números de seqüência são particularmente úteis. O receptor pode usar os números de seqüência para determinar se uma diferença de mais de 20 milissegundos nas marcas de tempo é devida a uma nova rajada de voz ou a pacotes perdidos.

7.3.3 Recuperação de perda de pacotes

Já discutimos com algum detalhe como uma aplicação de telefone por Internet pode lidar com a variação de atraso de pacote. Agora, descreveremos brevemente diversos esquemas que tentam preservar uma qualidade aceitável de áudio na presença da perda de pacotes. Esses esquemas são denominados **esquemas de recuperação de perdas**. Definimos aqui a perda de pacotes em um sentido amplo: um pacote será considerado perdido se nunca chegar ao receptor ou se chegar após o tempo de reprodução programado. Nosso exemplo do telefone por Internet servirá outra vez de contexto para a descrição de esquemas de recuperação de perdas.

Como mencionamos no início desta seção, em geral a retransmissão de pacotes perdidos não é apropriada para aplicações interativas em tempo real como telefone por Internet. Na verdade, a retransmissão de um pacote que perdeu seu prazo de reprodução não serve para absolutamente nada. E a retransmissão de um pacote que transbordou de uma fila de roteador normalmente não pode ser feita com a necessária rapidez. Devido a essas considerações, aplicações de telefone por Internet freqüentemente usam algum tipo de esquema de prevenção de perda. Dois desses esquemas são a **FEC** e a **intercalação**.

FEC (correção de erros de repasse)

A idéia básica da FEC é adicionar informações redundantes à corrente de pacotes original. Ao custo de aumentar marginalmente a taxa de transmissão do áudio da corrente, a informação redundante pode ser usada para reconstruir aproximações ou versões exatas de alguns pacotes perdidos. Esboçamos, agora, dois mecanismos de FEC segundo [Bolot, 1996] e [Perkins, 1998]. O primeiro mecanismo envia uma porção redundante codificada após cada n porções. A porção redundante é obtida por XOR das n porções originais [Shacham, 1990]. Desse modo, se qualquer pacote do grupo de $n + 1$ pacotes for perdido, o receptor poderá reconstruir integralmente o pacote perdido. Mas, se dois ou mais pacotes de um grupo forem perdidos, o receptor não poderá reconstruir os pacotes perdidos. Mantendo $n + 1$ (o tamanho do grupo) pequeno, uma

grande fração dos pacotes perdidos pode ser recuperada, quando a perda não for excessiva. Contudo, quanto menor o tamanho do grupo, maior será o aumento relativo da taxa de transmissão da corrente de áudio. Em particular, a taxa de transmissão aumenta de um fator de $1/n$; por exemplo, se $n = 3$, então a taxa de transmissão aumenta 33 por cento. Além disso, esse esquema simples aumenta o atraso de reprodução, pois o receptor tem de esperar o término da recepção do grupo de pacotes inteiro antes de poder começar a reproduzir. Se o leitor quiser mais detalhes práticos sobre como a FEC funciona para transporte de multimídia, consulte [RFC 2733].

O segundo mecanismo de FEC é enviar uma corrente de áudio de resolução mais baixa como informação redundante. Por exemplo, o remetente pode criar uma corrente de áudio nominal e uma corrente de áudio correspondente de baixa resolução e baixa taxa de bits. (A corrente nominal poderia ser uma codificação PCM de 64 kbps e a corrente de qualidade mais baixa, uma codificação GSM de 13 kbps.) A corrente de baixa taxa de bits é denominada corrente redundante. Como mostra a Figura 7.7, o remetente constrói o enésimo pacote tomando a enésima porção da corrente nominal e anexando a ela a porção de ordem $(n - 1)$ da corrente redundante. Dessa maneira, sempre que houver perda de pacote não consecutiva, o receptor pode ocultar a perda reproduzindo a porção codificada de baixa taxa de bits que chegar com o pacote subseqüente. É evidente que as porções de baixa taxa de bits oferecem qualidade mais baixa do que as porções nominais. Contudo, uma corrente na qual a maioria das porções é de alta qualidade, as porções de baixa qualidade são ocasionais e nenhuma porção é perdida oferece boa qualidade geral de áudio. Note que, nesse esquema, o receptor tem de receber somente dois pacotes antes da reprodução, de modo que o aumento no atraso de reprodução é pequeno. Além disso, se a codificação de baixa taxa de bits for muito menor do que a codificação nominal, o aumento marginal da taxa de transmissão será pequeno.

Para enfrentar perdas consecutivas, podemos utilizar uma variação simples. Em vez de anexar apenas a porção de baixa taxa de bits de ordem $(n - 1)$ à enésima porção nominal, o remetente pode anexar as porções de baixa taxa de bits de ordens $(n - 1)$ e $(n - 2)$ ou anexar as porções de baixa taxa de bits de ordens $(n - 1)$ e $(n - 3)$ e assim por diante. Anexando mais porções de baixa taxa de bits a cada porção nominal, a qualidade do áudio no receptor fica aceitável para uma variedade mais ampla de ambientes de melhor esforço adversos. Por outro lado, as porções adicionais aumentam a largura de banda de transmissão e o atraso de reprodução.

Free Phone [Freephone, 2004] e RAT [RAT, 2004] são aplicações de telefone por Internet bem documentadas que usam FEC. Elas podem transmitir correntes de áudio de qualidade mais baixa juntamente com a corrente de áudio nominal, como descrito antes. Veja também [Rosenberg, 2000].

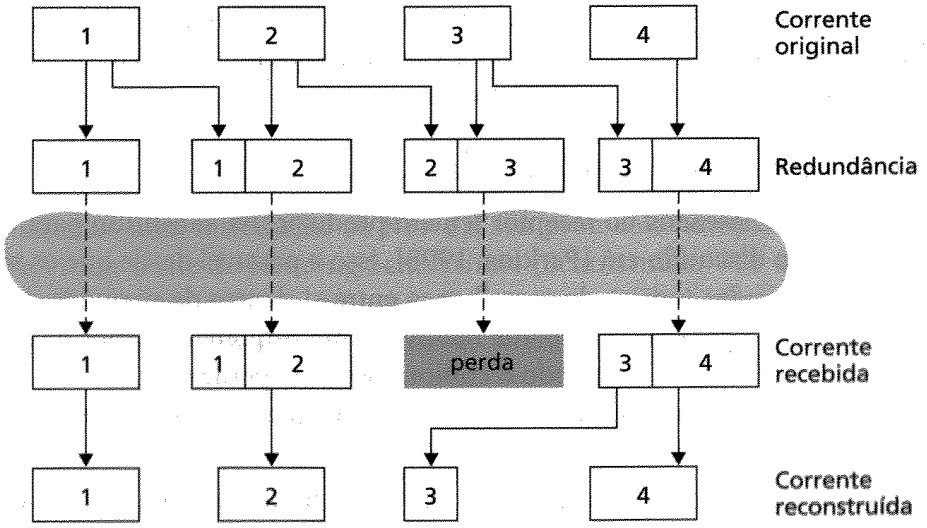


Figura 7.7 Dando carona à informação redundante de qualidade mais baixa

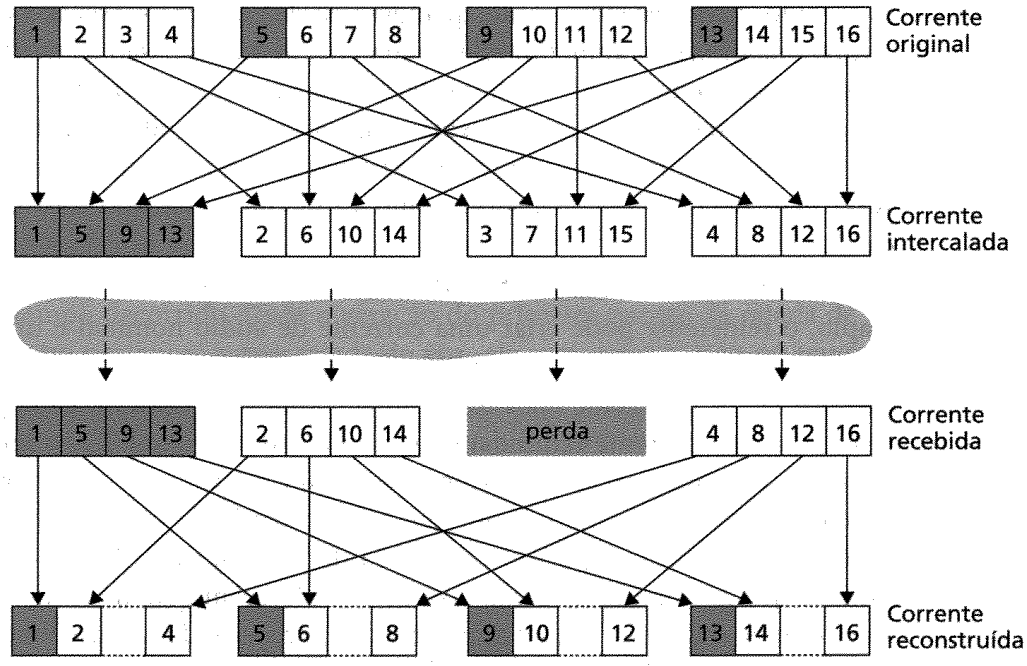


Figura 7.8 Envio de áudio intercalado

Intercalação

Como uma alternativa à transmissão redundante, uma aplicação de telefone por Internet pode enviar áudio intercalado. Como mostra a Figura 7.8, o remetente rearranja a seqüência das unidades de dados de áudio antes da transmissão, de modo que unidades originalmente adjacentes fiquem separadas por uma certa distância na corrente transmitida. A intercalação pode atenuar o efeito da perda de pacotes. Se, por exemplo, as unidades têm comprimento de 5 milissegundos e as porções são de 20 milissegundos (isto é, 4 unidades por porção), então a primeira porção pode conter unidades 1, 5, 9, 13; a segunda porção pode conter unidades 2, 6, 10, 14, e assim por diante. A Figura 7.8 mostra que a perda de um único pacote de uma corrente intercalada resulta em várias pequenas lacunas na corrente reconstruída, em vez de em uma única lacuna grande que ocorreria com um sistema não intercalado.

A intercalação pode melhorar significativamente a qualidade percebida de uma corrente de áudio [Perkins, 1998]. Além disso, a sobrecarga é baixa. A desvantagem óbvia da intercalação é que ela aumenta a latência, o que limita seu uso em aplicações interativas como o telefone por Internet, embora possa funcionar bem para fluxo contínuo de áudio armazenado. Uma importante vantagem da intercalação é que ela não aumenta as exigências de largura de banda de uma corrente.

Reparação de correntes de áudio danificadas realizada no receptor

Esquemas de recuperação baseados no receptor tentam produzir um substituto para um pacote perdido semelhante ao original. Como discutido em [Perkins, 1998], isso é possível desde que os sinais de áudio, em especial os de voz, exibam grandes índices de semelhança entre si dentro de períodos de tempo reduzidos. Assim, essas técnicas funcionam para taxas de perda relativamente pequenas (menos de 15 por cento) e para pacotes pequenos (4–40 milissegundos). Quando o comprimento da perda se aproxima do comprimento de um fonema (5–100 milissegundos), essas técnicas causam pane, já que fonemas inteiros podem ser perdidos pelo ouvinte.

Talvez o modo mais simples de recuperação baseada no receptor seja a repetição de pacotes. A repetição de pacotes substitui pacotes perdidos por cópias de pacotes que chegaram imediatamente antes da perda. É de baixa complexidade computacional e funciona razoavelmente bem. Outro modo de recuperação baseada no receptor é a interpolação, que usa áudio antes e após a perda para interpolar um pacote ade-

quado para cobrir a perda. Ela funciona um pouco melhor do que a repetição de pacotes, mas é significativamente mais trabalhosa para processamento [Perkins, 1998].

7.3.4 Áudio e vídeo de fluxo contínuo armazenados

Vamos concluir esta seção falando um pouco sobre fluxo contínuo de áudio e vídeo armazenados. Aplicações de fluxo contínuo de áudio/vídeo armazenado também usam, tipicamente, números de seqüência, marcas de tempo e atraso de reprodução para aliviar ou até eliminar os efeitos da variação de atraso da rede. Contudo, há uma importante diferença entre áudio/vídeo interativo em tempo real e fluxo contínuo de áudio/vídeo armazenado. Especificamente, fluxo contínuo de áudio/vídeo pode tolerar atrasos significativamente maiores. De fato, quando um usuário requisita um clipe de áudio/vídeo, é provável que ele ache aceitável esperar 5 segundos ou mais antes que a reprodução se inicie. E a maioria dos usuários pode tolerar atrasos semelhantes após ações interativas, como um salto temporal dentro de uma corrente de mídia. Essa tolerância maior ao atraso dá ao desenvolvedor de aplicações maior flexibilidade ao projetar aplicações de mídias armazenadas.

7.4 Protocolos para aplicações interativas em tempo real

Aplicações interativas em tempo real, incluindo telefone por Internet e videoconferência, prometem liderar grande parte do crescimento futuro da Internet. Portanto, não é surpresa que organismos padronizadores, tais como a IETF e o ITU, tenham se ocupado durante tantos anos (e ainda se ocupem!) com a produção de padrões para essa classe de aplicações. Tendo à mão padrões apropriados para aplicações interativas em tempo real, empresas independentes poderão criar novos produtos atraentes que interagem uns com os outros. Nesta seção estudamos RTP, SIP e H.323 para aplicações interativas em tempo real. Todos os três conjuntos de padrões estão sendo implementados amplamente em produtos do setor.

7.4.1 Protocolo de tempo real (RTP)

Na seção anterior, aprendemos que o lado remetente de uma aplicação de multimídia anexa campos de cabeçalho às porções de áudio/vídeo antes de passá-las à camada de transporte. Esses campos de cabeçalhos contêm números de seqüência e marcas de tempo. Já que a maioria das aplicações de rede multimídia pode fazer uso de números de seqüência e de marcas de tempo, é conveniente ter uma estrutura de pacote padronizada que inclua campos para dados de áudio/vídeo, números de seqüência e marcas de tempo, bem como outros campos potencialmente úteis. O RTP, definido no RFC 1889, é um padrão desse tipo. Ele pode ser usado para transportar formatos comuns como PCM, GSM e MP3 para som e MPEG e H.263 para vídeo. O protocolo também pode ser usado para transportar formatos proprietários de som e de vídeo. Hoje, o RTP é amplamente implementado em centenas de protótipos de produtos e de pesquisa. Também é complementar a outros importantes protocolos interativos de tempo real, entre eles SIP e H.323.

Nesta seção, faremos uma introdução ao RTP e a seu protocolo companheiro, o RTCP. Também aconselhamos o leitor a visitar o site de Henning Schulzrinne [Schulzrinne, RTP 2004], que trata do RTP e traz uma profusão de informações sobre o assunto. O leitor também poderá visitar o site da Free Phone [Freephone, 2004], que documenta uma aplicação de telefone por Internet que usa RTP.

O básico do RTP

O RTP comumente roda sobre UDP. O lado remetente encapsula uma porção de mídia dentro de um pacote RTP, em seguida encapsula o pacote em um segmento UDP, e então passa o segmento para o IP. O lado receptor extrai o pacote RTP do segmento UDP, em seguida extrai a porção de mídia do pacote RTP e então passa a porção para o transdutor para decodificação e apresentação.

Como exemplo, considere a utilização do RTP para transportar voz. Suponha que a fonte de voz esteja codificada (isto é, amostrada, quantizada e digitalizada) em PCM a 64 kbps. Suponha também que a aplicação colete os dados codificados em porções de 20 milissegundos, isto é, 160 bytes por porção. O lado remetente precede cada porção dos dados de áudio com um **cabeçalho RTP** que contém o tipo de codificação de áudio, um número de seqüência e uma marca de tempo. O tamanho do cabeçalho RTP é normalmente 12 bytes. A porção de áudio, juntamente com o cabeçalho RTP, forma o **pacote RTP**. O pacote RTP é, então, enviado para dentro do socket de interface UDP. No lado receptor, a aplicação recebe o pacote RTP da interface do seu socket. A aplicação extrai a porção de áudio do pacote RTP e usa os campos de cabeçalho do pacote RTP para decodificar e reproduzir adequadamente a porção de áudio.

Se uma aplicação incorporar o RTP — em vez de um esquema proprietário para fornecer tipo de carga útil, número de seqüência ou marcas de tempo —, ela interagirá mais facilmente com as outras aplicações de rede multimídia. Por exemplo, se duas empresas diferentes desenvolvem um software de telefone por Internet e ambas incorporam o RTP a seu produto, pode haver alguma chance de um usuário que estiver usando um dos produtos de telefone por Internet conseguir se comunicar com um usuário que estiver usando o outro produto de telefone por Internet. Na Seção 7.4.3, veremos que o RTP é muito utilizado em conjunto com os padrões de telefonia por Internet.

Devemos enfatizar que o RTP não fornece nenhum mecanismo que assegure a entrega de dados a tempo nem fornece outras garantias de qualidade de serviço (QoS); ele não garante nem mesmo a entrega dos pacotes nem impede a entrega de pacotes fora de ordem. Na verdade, o encapsulamento realizado pelo RTP é visto somente nos sistemas finais. Os roteadores não distinguem datagramas IP que carregam pacotes RTP de datagramas IP que não os carregam.

O RTP permite que seja atribuída a cada fonte (por exemplo, uma câmera ou um microfone) sua própria corrente independente de pacotes RTP. Por exemplo, para uma videoconferência entre dois participantes, quatro correntes RTP podem ser abertas — duas correntes para transmitir o áudio (uma em cada direção) e duas correntes para transmitir o vídeo (uma em cada direção). Contudo, muitas técnicas de codificação populares — incluindo MPEG1 e o MPEG2 — conjugam áudio e vídeo em uma única corrente durante o processo de codificação. Quando áudio e vídeo são conjugados pelo codificador, somente uma corrente RTP é gerada em cada direção.

Pacotes RTP não são limitados às aplicações *unicast*. Eles podem também ser enviados por árvores *multicast* um-para-muitos e muitos-para-muitos. Para uma sessão *multicast* muitos-para-muitos, todos os remetentes e fontes da sessão em geral usam o mesmo grupo *multicast* para enviar suas correntes RTP. Correntes *multicast* RTP que formam um conjunto, como correntes de áudio e vídeo que emanam de vários remetentes em uma aplicação de videoconferência, pertencem a uma **sessão RTP**.

Campos de cabeçalho do pacote RTP

Como mostra a Figura 7.9, os quatro principais campos de cabeçalho do pacote RTP são os campos de tipo da carga útil, número de seqüência, marca de tempo e os campos identificadores da fonte.

O campo de tipo de carga útil do pacote RTP tem 7 bits de comprimento. Para uma corrente de áudio, o campo de tipo de carga útil serve para indicar o tipo de codificação de áudio (por exemplo, PCM, modulação delta adaptativa, codificação por previsão linear) que está sendo usado. Se um remetente decidir mudar a codificação no meio de uma sessão, ele poderá informar a mudança ao receptor por meio desse campo de tipo de carga útil. O remetente pode querer mudar o código para aumentar a qualidade do áudio ou para reduzir a taxa de bits da corrente RTP. A Tabela 7.1 apresenta alguns dos tipos de carga útil de áudio correntemente suportados pelo RTP.

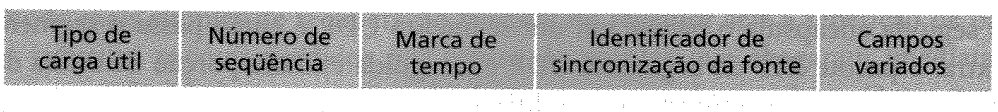


Figura 7.9 Campos de cabeçalho do RTP

Número do tipo de carga útil	Formato de áudio	Taxa de amostragem	Vazão
0	PCM lei μ	8 kHz	64 kbps
1	1016	8 kHz	4,8 kbps
3	GSM	8 kHz	13 kbps
7	LPC	8 kHz	2,4 kbps
9	G.722	16 kHz	48–64 kbps
14	áudio MPEG	90 kHz	—
15	G.728	8 kHz	16 kbps

Tabela 7.1 Tipos de carga útil de áudio suportados pelo RTP

Para uma corrente de vídeo, o tipo de carga útil é usado para indicar o tipo de codificação de vídeo (por exemplo, JPEG com movimento, MPEG1, MPEG2, H.261). Novamente, o remetente pode mudar a codificação de vídeo durante a sessão. A Tabela 7.2 apresenta alguns tipos de carga útil de vídeo atualmente suportados pelo RTP. Os outros campos importantes são:

- *Campo do número de seqüência.* O campo do número de seqüência tem comprimento de 16 bits. O número de seqüência é incrementado de uma unidade a cada pacote RTP enviado e pode ser usado pelo receptor para detectar perda de pacotes e restaurar a seqüência de pacotes. Por exemplo, se o lado receptor da aplicação receber uma corrente de pacotes RTP com uma lacuna entre os números de seqüência 86 e 89, então o receptor saberá que os pacotes 87 e 88 estão faltando. O receptor poderá, então, tentar ocultar os dados perdidos.
- *Campo de marca de tempo.* O campo de marca de tempo tem 32 bits de comprimento. Ele reflete o instante da amostragem do primeiro byte no pacote RTP. Como vimos na seção anterior, o receptor pode usar marcas de tempo para eliminar a variação de atraso dos pacotes introduzida na rede e fornecer recepção sincronizada no receptor. A marca de tempo é derivada de um relógio de amostragem no remetente. Como exemplo, para áudio, o relógio de marca de tempo é incrementado de uma unidade para cada período de amostragem (por exemplo, a cada 125 microssegundos para um relógio de amostragem de 8 kHz); se a aplicação de áudio gerar porções consistindo em 160 amostras codificadas, então a marca de tempo aumentará em 160 para cada pacote RTP enquanto a fonte estiver ativa. O relógio de marca de tempo continua a aumentar a uma taxa constante, mesmo que a fonte esteja inativa.
- *Identificador de sincronização da fonte* (*synchronization source identifier* — SSRC). O campo SSRC tem 32 bits de comprimento. Ele identifica a fonte da corrente RTP. Em geral, cada fonte de uma sessão RTP tem um SSRC distinto. O SSRC não é o endereço IP do remetente, mas um número atribuído aleatoriamente pela fonte quando uma nova corrente é iniciada. A probabilidade de que seja atribuído o mesmo SSRC a duas correntes é muito pequena. Se isso acontecer, as duas fontes escolherão novos valores SSRC.

Número do tipo de carga útil	Formato de vídeo
26	Motion JPEG
31	H.261
32	Vídeo MPEG1
33	Vídeo MPEG2

Tabela 7.2 Alguns tipos de carga útil de vídeo suportados pelo RTP

Desenvolvimento de aplicações de software com RTP

Há duas abordagens para desenvolver uma aplicação de rede baseada em RTP. A primeira abordagem é o desenvolvedor incorporar o RTP à mão, isto é, escrever, ele mesmo, o código que executa o encapsulamento RTP no lado remetente e decodifica o RTP no lado do receptor. A segunda abordagem é o desenvolvedor da aplicação utilizar as bibliotecas de RTP (para programadores que usam linguagem C) e as classes Java (para programadores em Java) existentes, que executam o encapsulamento e a decodificação para a aplicação. Como você pode estar ansioso para escrever sua primeira aplicação de rede multimídia usando RTP, vamos nos aprofundar um pouco nessas duas abordagens. (A tarefa de programação ao final deste capítulo servirá de guia para a criação de uma aplicação RTP.) Faremos isso no contexto da comunicação *unicast* (em vez de *multicast*).

Lembre-se de que, no Capítulo 2, mostramos que a API UDP requer que, para cada segmento UDP que envia, o processo remetente determine o endereço IP de destino e o número da porta de destino antes de enviar o pacote para dentro do socket UDP. Então, o segmento UDP vagueará pela Internet e (se não for perdido, por exemplo, por causa de transbordamento de buffer de roteador), a qualquer momento, chegará ao socket do processo receptor para a aplicação. Esse socket terá um endereço completo, ou seja, o endereço IP de destino e o número da porta de destino. Na verdade, qualquer datagrama IP que contenha esse endereço IP de destino e número de porta de destino será direcionado ao socket UDP do processo receptor. (A API UDP também permite que o desenvolvedor da aplicação estabeleça o número UDP de porta da fonte; contudo, esse valor não tem nenhum efeito sobre o processo para o qual o segmento é enviado.) É importante notar que o RTP não impõe um número de porta específico. Quando o desenvolvedor de aplicações cria uma aplicação RTP, ele especificará os números de porta para os dois lados da aplicação.

Como parte da tarefa de programação deste capítulo, você escreverá um servidor RTP que encapsula quadros de vídeo armazenado dentro de pacotes RTP. Você fará isso à mão; isto é, sua aplicação pegará um quadro de vídeo, adicionará os cabeçalhos RTP ao quadro para criar um pacote RTP e então passará o quadro RTP para a porta UDP. Para fazer isso, você precisará criar campos de reserva para os vários cabeçalhos RTP, incluindo um campo de número de seqüência e um campo de marca de tempo. E, para cada pacote RTP que for criado, você terá de determinar o número de seqüência e a marca de tempo apropriados. Você codificará explicitamente todas essas operações RTP dentro do lado remetente de sua aplicação. Como mostra a Figura 7.10, sua API com a rede será o socket padrão UDP.

Uma abordagem alternativa (que não é desenvolvida na tarefa de programação) é usar uma classe Java RTP (ou uma biblioteca RTP em C para programadores que utilizam essa linguagem) para implementar as operações RTP. Como mostra a Figura 7.11, com essa abordagem o desenvolvedor terá a impressão de que o RTP é parte da camada de transporte. Sem entrar nos aspectos fundamentais (pois eles dependem da classe/biblioteca), quando envia uma porção de mídia para dentro da API, o lado remetente da aplicação precisa fornecer à interface a porção de mídia em si, um número de tipo de carga útil, um SSRC e uma marca de tempo, juntamente com um número de porta de destino e um endereço IP de destino. Mencionamos aqui que a Java Media Framework (JFM) inclui uma implementação completa de RTP.

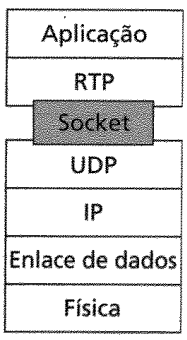


Figura 7.10 RTP é parte da aplicação e está acima do socket UDP

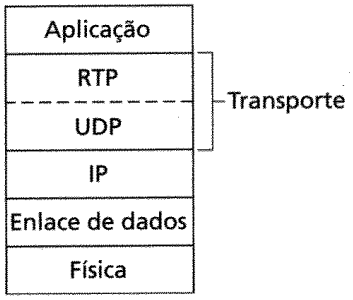


Figura 7.11 RTP pode ser visto como uma subcamada da camada de transporte

7.4.2 Protocolo de Controle RTP (RTCP)

O RFC 1889 também especifica o RTCP, um protocolo que uma aplicação de rede multimídia pode usar juntamente com o RTP. Como mostrado no cenário *multicast* da Figura 7.12, pacotes RTCP são transmitidos por cada participante de uma sessão RTP para todos os outros participantes da sessão usando IP *multicast*. Para uma sessão RTP há, tipicamente, um único endereço *multicast* e todos os pacotes RTP e RTCP pertencentes à sessão usam o endereço *multicast*. Pacotes RTP e RTCP se distinguem uns dos outros pela utilização de números de porta distintos. (O número de porta RTCP é configurado para ser igual ao número da porta RTP mais uma unidade.)

Pacotes RTCP não encapsulam porções de áudio ou de vídeo. Em vez disso, eles são enviados periodicamente e contêm relatórios de remetente e/ou receptor com dados estatísticos que podem ser úteis para a aplicação. Esses dados estatísticos contêm número de pacotes enviados, número de pacotes perdidos e variação de atraso entre chegadas. A especificação RTP [RFC 3550] não determina o que a aplicação deve fazer com essas informações de retorno; isso depende do desenvolvedor da aplicação. Remetentes podem usar as informações, por exemplo, para modificar suas taxas de transmissão. Essas informações também podem ser utilizadas para finalidades de diagnóstico; por exemplo, receptores podem determinar se os problemas são locais, regionais ou globais.

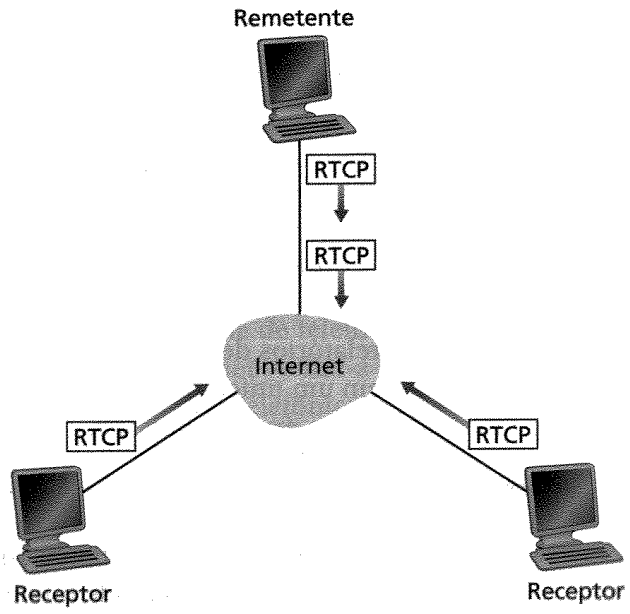


Figura 7.12 Ambos, remetentes e receptores, enviam mensagens RTCP

Tipos de pacotes RTCP

Para cada corrente RTP que um receptor recebe como parte de uma sessão, ele gera um relatório de recepção. O receptor agrega seus relatórios de recepção em um único pacote RTCP. O pacote é então enviado para a árvore *multicast* que conecta todos os participantes da sessão. O relatório de recepção contém diversos campos; os mais importantes são relacionados a seguir:

- O SSRC da corrente RTP para a qual o relatório de recepção está sendo gerado.
- A fração de pacotes perdida dentro da corrente RTP. Cada receptor calcula o número de pacotes RTP perdidos dividido pelo número de pacotes RTP enviados como parte da corrente. Se um remetente receber relatórios de recepção indicando que os receptores estão recebendo apenas uma pequena fração dos pacotes transmitidos por ele, ele poderá passar para uma taxa de codificação mais baixa, com o objetivo de reduzir o congestionamento da rede e melhorar a taxa de recepção.
- O último número de seqüência recebido na corrente de pacotes RTP.
- A variação de atraso entre chegadas, que é uma estimativa amortecida da variação do intervalo de tempo entre chegadas de pacotes sucessivos na corrente RTP.

Para cada corrente RTP que um remetente estiver transmitindo, ele criará e transmitirá pacotes de relatório de remetente RTCP. Esses pacotes contêm informações sobre a corrente RTP, incluindo:

- O SSRC da corrente RTP.
- A marca de tempo e o tempo do relógio físico (tempo real) do pacote da corrente RTP gerado mais recentemente.
- O número de pacotes enviados na corrente.
- O número de bytes enviados na corrente.

Relatórios de remetente podem ser usados para sincronizar diferentes correntes de mídia dentro de uma sessão RTP. Por exemplo, considere uma aplicação de videoconferência para a qual cada remetente gera duas correntes RTP independentes, uma para vídeo e uma para áudio. As marcas de tempo desses pacotes RTP estão vinculadas aos relógios de amostragem de vídeo e áudio, e não ao tempo do relógio físico (isto é, ao tempo real). Cada relatório de remetente RTCP contém, para o pacote gerado mais recentemente na corrente RTP associada, a marca de tempo do pacote RTP e o tempo do relógio físico referente à criação do pacote. Assim, os pacotes de relatório de remetente RTCP associam o relógio de amostragem ao relógio físico, de tempo real. Os receptores podem usar essa associação presente nos relatórios de remetente para sincronizar a reprodução de áudio e vídeo.

Para cada corrente RTP que está transmitindo, o remetente também cria e transmite pacotes de descrição da fonte. Esses pacotes contêm informações sobre a fonte, como o endereço de e-mail do remetente, o nome do remetente e a aplicação que gera a corrente RTP. Contêm também o SSRC da corrente RTP associada e fornecem um mapeamento entre o identificador da fonte (isto é, o SSRC) e o nome do usuário/hospedeiro.

Pacotes RTCP podem ser empilhados, isto é, relatórios de recepção do receptor, relatórios de remetentes e descrições da fonte podem ser concatenados em um único pacote. O pacote resultante é, então, encapsulado dentro de um segmento UDP e repassado para a árvore *multicast*.

Escalabilidade da largura de banda do RTCP

Você já deve ter observado que o RTCP apresenta um problema potencial de escalabilidade. Considere, por exemplo, uma sessão RTP que consista em um remetente e um grande número de receptores. Se cada um dos receptores gerar periodicamente pacotes RTCP, então a taxa de transmissão agregada de pacotes RTCP poderá exceder em muito a taxa de pacotes RTP enviados pelo remetente. Observe que o total de tráfego RTP enviado à árvore *multicast* não muda à medida que o número de receptores aumenta, ao passo que o total de tráfego RTCP aumenta linearmente com o número de receptores. Para resolver esse problema de

escalabilidade, o RTCP muda a taxa com a qual um participante envia pacotes RTCP para dentro da árvore *multicast* como uma função do número de participantes da sessão. Além disso, como cada participante envia pacotes de controle para todos os outros, cada um deles pode estimar o número total de participantes da sessão [Friedman, 1999].

O RTCP tenta limitar seu tráfego a 5 por cento da largura de banda da sessão. Por exemplo, suponha que haja um remetente que está enviando vídeo a uma taxa de 2 Mbps. Então, o RTCP tenta limitar seu tráfego a 5 por cento de 2 Mbps, ou a 100 kbps, como a seguir. O protocolo dá 75 por cento dessa taxa, ou seja, 75 kbps, aos receptores; dá os restantes 25 por cento da taxa, ou 25 kbps, ao remetente. Os 75 kbps dedicados aos receptores são compartilhados igualmente entre eles. Assim, se houver R receptores, cada receptor conseguirá enviar tráfego RTCP a uma taxa de 75/R kbps e o remetente conseguirá enviar tráfego RTCP a uma taxa de 25 kbps. Um participante (remetente ou receptor) determina o período de transmissão do pacote RTCP calculando dinamicamente o tamanho médio do pacote RTCP (para toda a sessão) e dividindo o tamanho médio do pacote RTCP pela taxa alocada a ele, participante. Em suma, para um remetente, o período para transmitir pacotes RTCP é:

$$T = \frac{\text{número de remetentes}}{0{,}25 \cdot 0{,}05 \cdot \text{largura de banda da sessão}} \text{ (tamanho médio do pacote RTCP)}$$

E, para um receptor, o período para transmitir pacotes RTCP é:

$$T = \frac{\text{número de recebedores}}{0{,}75 \cdot 0{,}05 \cdot \text{largura da banda da sessão}} \text{ (tamanho médio do pacote RTCP)}$$

7.4.3 SIP

Imagine um mundo no qual, enquanto você está trabalhando em seu computador, ele recebe suas chamadas telefônicas pela Internet. E, quando você sai para descansar um pouco, as novas chamadas telefônicas que chegam são automaticamente roteadas para seu PDA. E, quando você está dirigindo, novas chamadas telefônicas são automaticamente roteadas para algum equipamento conectado à Internet instalado no seu carro. Nesse mesmo mundo, durante uma conferência em rede você pode acessar uma agenda de endereços e convidar outras pessoas para participar da reunião virtual. Esses outros participantes poderão estar trabalhando em seus computadores ou em trânsito com seus PDAs à mão ou em seus carros — não importa onde estejam, seu convite será roteado para eles de modo transparente. Ainda nesse mesmo mundo, quando você visitar alguma home page pessoal, encontrará um link 'Ligue para mim'; ao clicar sobre esse link, será estabelecida uma sessão de telefone Internet entre seu computador e o proprietário da home page (onde quer que ele esteja).

Nesse mundo não existirá mais uma rede de telefonia por comutação de circuitos. Em vez disso, todas as chamadas telefônicas passarão pela Internet — fim-a-fim. Ainda nesse mesmo mundo, empresas não utilizarão mais centrais privadas de comutação telefônica (*private automatic branch exchange* — PABX), isto é, mesas locais de comutação de circuitos, para manipular chamadas telefônicas internas. Em vez disso, o tráfego telefônico interno fluirá pela LAN de alta velocidade da empresa.

Tudo isso pode parecer ficção científica. E é claro que as redes de comutação de circuitos e centrais de PABX de hoje não vão desaparecer completamente no futuro próximo (Jiang, 2001). Não obstante, já existem protocolos e produtos para transformar essa visão em realidade. Entre os protocolos mais promissores para essa finalidade está o Protocolo de Inicialização de Sessão (*Session Initiation Protocol* — SIP), definido no RFC 3261. O SIP é um protocolo simples que faz o seguinte:

- Provê mecanismos para estabelecer chamadas entre dois interlocutores por uma rede IP. Permite que quem chama avise ao que é chamado que quer iniciar uma chamada. Permite que os participantes concordem com a codificação da mídia. E também permite que encerrem as chamadas.
- Provê mecanismos que permitem a quem chama determinar o endereço IP corrente de quem é chamado. Os usuários não têm um endereço IP único, fixo, porque podem receber endereços dinamicamente (usando DHCP) e porque podem ter vários equipamentos IP, cada um com um endereço IP diferente.
- Provê mecanismos para gerenciamento de chamadas, tais como adicionar novas correntes de mídia, mudar a codificação, convidar outros participantes, tudo durante a chamada, e ainda transferir e segurar chamadas.

Estabelecendo uma chamada para um endereço IP conhecido

Para entender a essência do SIP, é melhor examinar um exemplo concreto. Nesse exemplo, Alice está trabalhando em seu computador e quer chamar Bob, que também está trabalhando em seu próprio computador. Ambos os PCs estão equipados com software baseado em SIP para fazer e receber chamadas telefônicas. Nesse exemplo inicial, admitiremos que Alice conhece o endereço IP do PC de Bob. A Figura 7.13 ilustra o processo de estabelecimento de chamada do SIP.

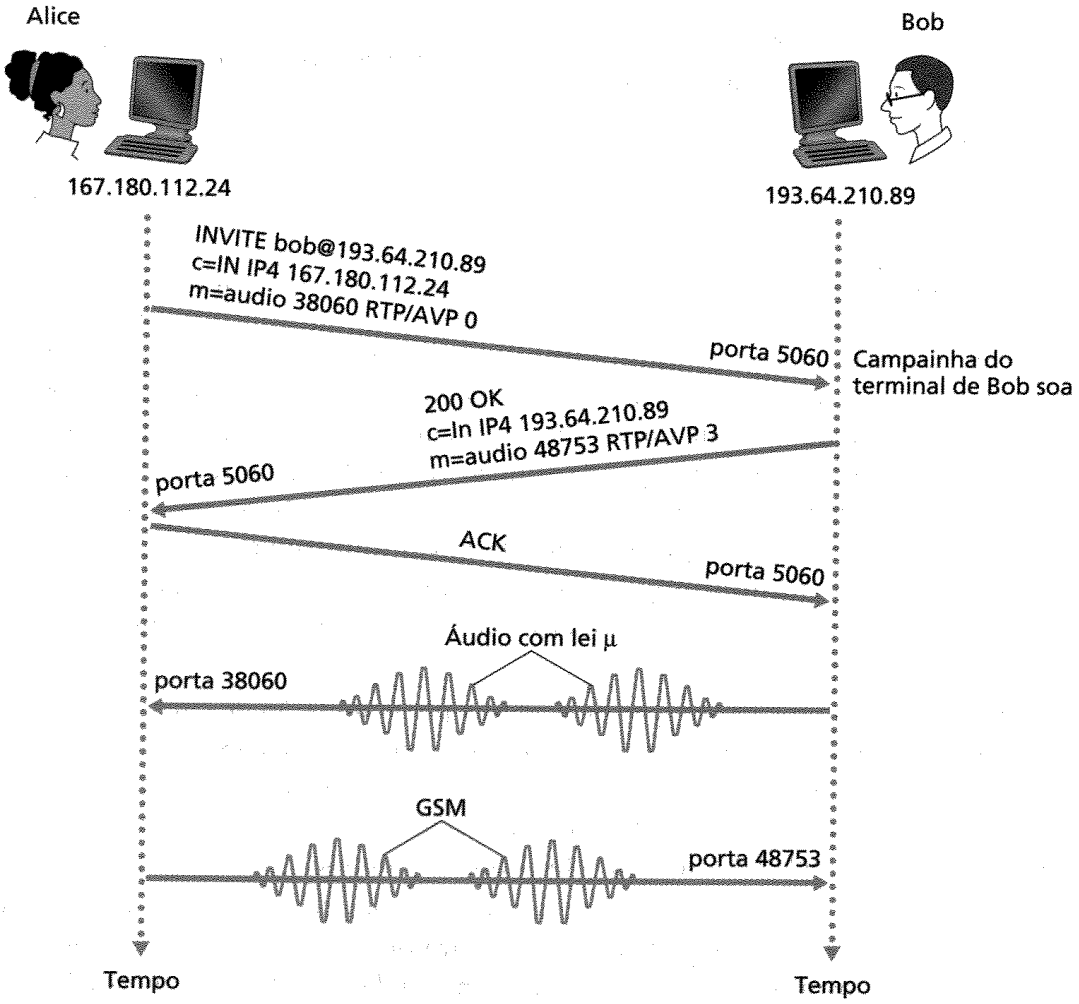


Figura 7.13 Estabelecimento de chamada SIP quando Alice conhece o endereço IP de Bob

Na Figura 7.13, vemos que uma sessão SIP começa quando Alice envia a Bob uma mensagem INVITE, que é parecida com uma mensagem de requisição HTTP. Essa mensagem INVITE é enviada por UDP à porta bem conhecida 5060, que é a porta do SIP. (Mensagens SIP também podem ser enviadas por TCP). A mensagem INVITE inclui um identificador para Bob (bob@193.64.210.89), uma indicação do endereço IP corrente de Alice, uma indicação de que Alice deseja receber áudio, o qual deve ser codificado em formato AVP 0 (PCM codificado com lei μ) e encapsulado em RTP e uma indicação de que ela quer receber os pacotes RTP na porta 38060. Após receber a mensagem INVITE de Alice, Bob envia uma mensagem de resposta SIP, que é parecida com uma mensagem de resposta HTTP. Essa mensagem de resposta SIP também é enviada à porta SIP 5060. A resposta de Bob inclui um 200 OK, bem como uma indicação de seu endereço IP, o código e o empacotamento que deseja para recepção e seu número de porta para a qual os pacotes de áudio devem ser enviados.

Note que, nesse exemplo, Alice e Bob vão usar mecanismos diferentes de codificação de áudio: Alice deve codificar seu áudio com GSM, ao passo que Bob deve codificar seu áudio com a lei μ do PCM. Após receber a resposta de Bob, Alice lhe envia uma mensagem SIP de reconhecimento (ACK). Após essa transação SIP, Bob e Alice podem conversar. (Para melhor visualização, a Figura 7.13 mostra Alice falando depois de Bob, mas, na verdade, normalmente eles falariam ao mesmo tempo.) Bob codificará e empacotará o áudio como solicitado e enviará os pacotes de áudio para a porta número 38060 no endereço IP 167.180.112.24. Alice também codificará e empacotará o áudio como solicitado e enviará os pacotes de áudio para a porta número 48753 no endereço IP 193.64.210.89.

Com esse exemplo simples, aprendemos várias características fundamentais do SIP. Em primeiro lugar, o SIP é um protocolo 'fora da banda': as mensagens SIP são enviadas e recebidas em portas diferentes das utilizadas para enviar e receber os dados da mídia. Em segundo lugar, as próprias mensagens SIP podem ser lidas em ASCII e são parecidas com mensagens HTTP. Em terceiro lugar, o SIP requer que todas as mensagens sejam reconhecidas, portanto, ele pode executar sobre UDP ou TCP.

Nesse exemplo, vamos considerar o que aconteceria se Bob não tivesse um codec PCM lei μ para codificar áudio. Nesse caso, em vez de responder com 200 OK, Bob responderia com um 600 Not Acceptable e apresentaria na mensagem uma lista com todos os codecs que ele pode usar. Então, Alice escolheria um dos codecs da lista e enviaria outra mensagem INVITE, desta vez anunciando o codec escolhido. Bob também poderia simplesmente rejeitar a chamada enviando um dos muitos códigos de rejeição de resposta possíveis. (Há muitos desses códigos, entre eles: 'ocupado' (busy), 'encerrado' (gone), 'pagamento solicitado' (payment required) e 'proibido' (forbidden).

Endereços SIP

No exemplo anterior, o endereço SIP de Bob é sip:bob@193.64.210.89. Contudo, esperamos que muitos endereços SIP — se não a maioria — sejam parecidos com endereços de e-mail. Por exemplo, o endereço de Bob poderia ser sip:bob@domain.com. Quando o dispositivo SIP de Alice enviasse uma mensagem INVITE, a mensagem incluiria esse endereço semelhante a um endereço de e-mail; a infra-estrutura SIP então rotearia a mensagem ao dispositivo IP que Bob está usando no momento em questão (como discutiremos mais adiante). Outras formas possíveis para o endereço SIP poderiam ser o número de telefone legado de Bob ou simplesmente seu nome próprio, ou seu primeiro ou segundo sobrenome (admitindo-se que sejam únicos).

Uma característica interessante de endereços SIP é que eles podem ser incluídos em páginas Web, exatamente como endereços de e-mail são incluídos nessas páginas juntamente com o URL de envio. Por exemplo, suponha que Bob tem uma home page pessoal e quer fornecer um meio para que os visitantes da página entrem em contato com ele. Ele poderia, então, simplesmente incluir o URL 'sip:bob@domain.com'. Quando o visitante clicar sobre o URL, a aplicação SIP no dispositivo do visitante será lançada e uma mensagem INVITE será enviada a Bob.

Mensagens SIP

Nessa curta introdução ao SIP, não abordaremos todos os tipos e cabeçalhos de mensagens SIP. Em lugar disso, examinaremos brevemente a mensagem SIP INVITE, juntamente com algumas linhas de cabeçalho comuns. Mais uma vez, vamos supor que Alice quer iniciar uma chamada telefônica IP com Bob e, desta vez, ela conhece somente o endereço SIP de Bob, bob@domain.com, e não conhece o endereço IP do dispositivo que Bob está correntemente usando. Então, sua mensagem poderia ser algo parecido com:

```
INVITE sip:bob@domain.com SIP/2.0
Via: SIP/2.0/UDP 167.180.112.24
From: sip:alice@hereway.com
To: sip:bob@domain.com
Call-ID: a2e3a@pigeon.hereway.com
Content-Type: application/sdp
Content-Length: 885

c=IN IP4 167.180.112.24
m=audio 38060 RTP/AVP 0
```

A linha INVITE inclui a versão do SIP, assim como uma mensagem de requisição HTTP. Sempre que uma mensagem SIP passa por um dispositivo SIP (incluindo o dispositivo que origina a mensagem), ele anexa um cabeçalho Via, que indica o endereço IP do dispositivo. (Veremos, em breve, que uma mensagem INVITE típica passa por muitos dispositivos SIP antes de chegar à aplicação SIP do usuário que foi chamado.) Semelhante a uma mensagem de e-mail, a mensagem SIP inclui uma linha de cabeçalho From e uma linha de cabeçalho To. Inclui também um Call-ID (identificador de chamadas), que identifica a chamada exclusivamente (semelhante à mensagem-ID no e-mail). Inclui uma linha de cabeçalho Content-Type (Tipo de Conteúdo), que define o formato usado para descrever o conteúdo da mensagem SIP. Inclui também uma linha de cabeçalho Content-Lenght (Tamanho de Conteúdo), que indica o comprimento em bytes do conteúdo na mensagem. Finalmente, após um carriage return e um line feed, a mensagem contém o corpo de informação. Nesse caso, o conteúdo fornece informações sobre o endereço IP de Alice e como Alice quer receber o áudio.

Tradução de nome e localização de usuário

No exemplo da Figura 7.13, admitimos que o dispositivo SIP de Alice conhecia o endereço IP onde Bob podia ser contatado. Mas essa premissa é pouco realista, não somente porque muitas vezes os endereços IP são atribuídos dinamicamente com DHCP, mas também porque Bob pode ter vários dispositivos IP (por exemplo, dispositivos diferentes em casa, no trabalho e no carro). Portanto, agora vamos supor que Alice conhece somente o endereço de e-mail de Bob, bob@domain.com, e que esse mesmo endereço é usado para chamadas SIP. Nesse caso, Alice precisa obter o endereço IP do dispositivo que o usuário bob@domain.com está usando naquele momento. Para descobrir esse endereço, Alice cria uma mensagem INVITE que começa com INVITE bob@domain.com SIP/2.0, e envia essa mensagem a um proxy SIP. O proxy responderá com uma resposta SIP que poderá incluir o endereço IP do dispositivo que bob@domain.com está usando naquele momento. Alternativamente, a resposta poderia incluir o endereço IP da caixa postal de voz de Bob, ou poderia incluir um URL de uma página Web (que diz "Bob está dormindo. Não perturbe!"). Além disso, o resultado devolvido pelo proxy poderia depender de quem chama: se a chamada vier da sogra de Bob, ele poderia responder com o URL que aponta para a página Web do "estou dormindo"!

Agora, você provavelmente está imaginando como o servidor proxy pode determinar o endereço IP corrente para bob@domain.com. Para responder a essa pergunta, é preciso dizer algumas palavras sobre um outro dispositivo SIP, a entidade registradora SIP. Cada usuário SIP tem uma entidade registradora associada. Sempre que um usuário lança uma aplicação SIP em um dispositivo, a aplicação envia uma mensagem de registro SIP à entidade registradora, informando seu endereço IP corrente. Por exemplo, quando Bob lançasse sua aplicação SIP em seu PDA, a aplicação enviaria uma mensagem semelhante a essa:

```
REGISTER sip:domain.com SIP/2.0
Via: SIP/2.0/UDP 193.64.210.89
From: sip:bob@domain.com
To: sip:bob@domain.com
Expires: 3600
```

A entidade registradora monitora o endereço IP corrente de Bob. Sempre que ele mudar para um novo dispositivo SIP, esse novo dispositivo enviará uma nova mensagem de registro, indicando o novo endereço IP. Além disso, se Bob permanecer no mesmo dispositivo durante um longo período de tempo, o dispositivo enviará mensagens de confirmação de registro, indicando que o endereço IP mais recentemente enviado ainda é válido. (No exemplo acima, é preciso enviar mensagens de confirmação a cada 3.600 segundos para manter o endereço no servidor da entidade registradora.) O fato de a entidade registradora ser análoga a um servidor de nomes DNS com autoridade não é digno de nota; o servidor DNS traduz nomes de hospedeiros fixos para endereços IP fixos; a entidade registradora SIP traduz identificadores fixos de pessoas (por exemplo, bob@domain.com) para endereços IP dinâmicos. Muitas vezes, as entidades registradoras SIP e os proxies SIP são executadas no mesmo hospedeiro.

Agora vamos examinar como o servidor proxy SIP de Alice obtém o endereço IP corrente de Bob. Vimos, na discussão precedente, que o servidor proxy precisa simplesmente transmitir a mensagem INVITE de Alice à entidade registradora/proxy de Bob. Então, a registradora/proxy poderia transmitir a mensagem ao dispositivo SIP corrente de Bob. Finalmente, como agora Bob já recebeu a mensagem INVITE de Alice, ele pode enviar a ela uma resposta SIP.

Como exemplo, considere a Figura 7.14, na qual jim@umass.edu, que está correntemente trabalhando em 217.123.56.89, quer iniciar uma sessão de voz sobre IP com keith@upenn.edu, correntemente trabalhando em 197.87.54.21. São seguidas as seguintes etapas: (1) Jim envia uma mensagem INVITE ao proxy SIP de umass. (2) O proxy faz uma consulta DNS para a entidade registradora SIP de upenn.edu (que não é mostrada no diagrama) e então envia a mensagem ao servidor da registradora. (3) Como keith@upenn.edu não está mais registrado na entidade registradora de upenn, esta envia uma resposta de redirecionamento, indicando que é preciso tentar keith@eurecom.fr. (4) O proxy de umass envia uma mensagem INVITE à registradora SIP de eurecom. (5) A registradora de eurecom conhece o endereço IP de keith@eurecom.fr e repassa a mensagem INVITE ao hospedeiro 197.87.54.21, que está executando o cliente SIP de Keith. (6–8) Uma resposta SIP é devolvida por meio de registradoras/proxies ao cliente SIP em 217.123.56.89. (9) Então a mídia é enviada diretamente entre os dois clientes. (Também há uma mensagem de reconhecimento SIP, que não é mostrada.)

Nossa discussão sobre o SIP focalizou a inicialização de chamadas para chamadas de voz. Como o SIP é um protocolo de sinalização para inicializar e encerrar chamadas em geral, ele pode ser usado para chamadas de videoconferência, bem como para sessões de texto. Na verdade, o SIP tornou-se um componente fundamental em muitas aplicações de mensagem instantânea. Os leitores que quiserem saber mais sobre esse protocolo devem visitar o site Web de Henning Schulzrinne [Schulzrinne-SIP, 2004]. Em particular, nesse site você encontrará software de código-fonte aberto para clientes e servidores SIP [SIP software, 2004].

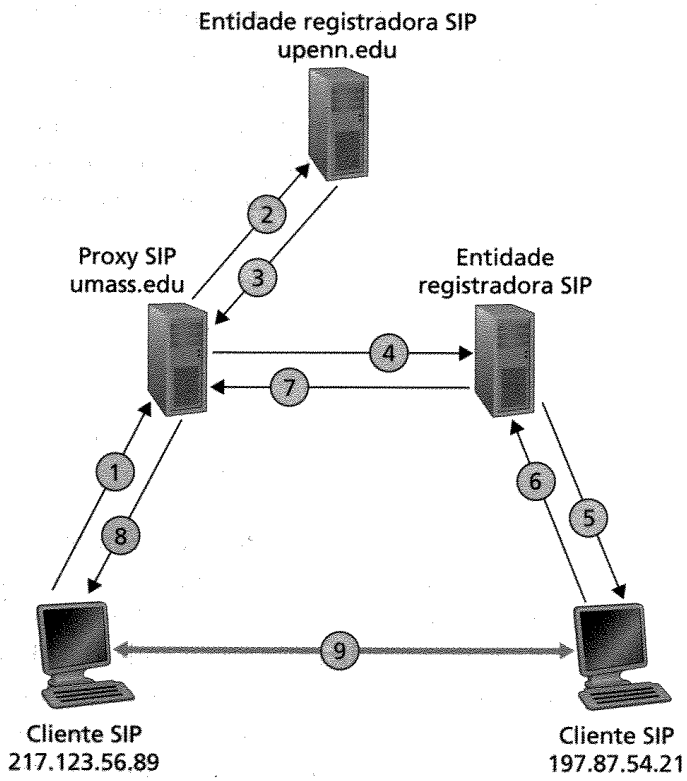


Figura 7.14 Inicialização de sessão envolvendo proxies e entidades registradoras SIP

7.4.4 H.323

Como uma alternativa ao SIP, o H.323 é um padrão popular para audioconferência e videoconferência entre sistemas finais na Internet. Como ilustrado na Figura 7.15, o padrão também abrange a maneira como sistemas finais ligados à Internet se comunicam com telefones ligados às redes normais de telefonia de comutação de circuitos. (O SIP também faz isso, embora não tenhamos discutido esse assunto.) O gatekeeper H.323 é um dispositivo semelhante a uma entidade registradora SIP.

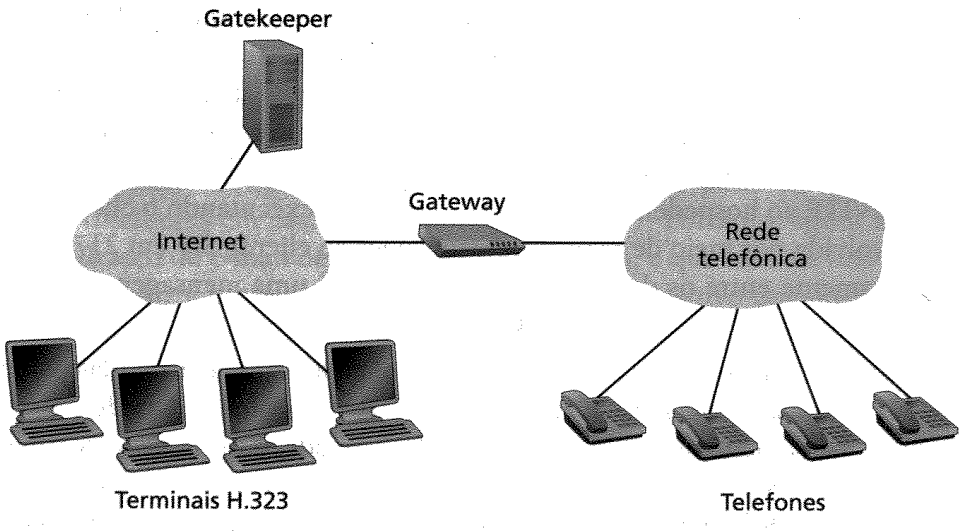


Figura 7.15 Sistemas finais H.323 ligados à Internet podem se comunicar com telefones ligados a uma rede telefônica de comutação de circuitos

O padrão H.323 é uma especificação guarda-chuva que inclui as seguintes especificações:

- Uma especificação para o modo como os terminais negociam codificações comuns de áudio/vídeo. Como o H.323 suporta uma variedade de padrões de codificação de áudio e vídeo, é preciso um protocolo para permitir que os terminais comunicantes cheguem a um acordo quanto a uma codificação comum.
- Uma especificação para o modo como porções de áudio e vídeo são encapsuladas e enviadas à rede. Em particular, para essa finalidade o H.323 impõe o RTP.
- Uma especificação para o modo como terminais se comunicam com seus respectivos gatekeepers.
- Uma especificação para o modo como telefones por Internet se comunicam por meio do gateway com os telefones comuns da rede pública de telefonia por comutação de circuitos.

No mínimo, cada terminal H.323 tem de suportar o padrão de compressão de voz G.711. O G.711 usa PCM para gerar voz digitalizada a 56 kbps ou 64 kbps. Embora o H.323 exija que cada terminal seja habilitado para voz (por meio do G.711), capacidades de vídeo são opcionais. Como o suporte de vídeo é opcional, os fabricantes de terminais podem vender tanto terminais de voz mais simples como terminais mais complexos que suportam áudio e vídeo. Capacidades de vídeo para um terminal H.323 são opcionais. Todavia, se um terminal suportar vídeo, então ele deverá (no mínimo) suportar o padrão de vídeo QCIF H.261 (176 x 144 pixels).

O H.323 é um padrão guarda-chuva abrangente que, além dos padrões e protocolos que acabamos de descrever, impõe um protocolo de controle H.245, um canal de sinalização Q.931 e um protocolo RAS para registro no gatekeeper.

Concluímos esta seção destacando algumas das diferenças mais importantes entre H.323 e SIP.

- O H.323 é um conjunto de protocolos completo, integrado verticalmente, para conferência multimídia: sinalização, registro, controle de admissão, transporte e codecs.
- O SIP, por outro lado, aborda apenas inicialização e gerenciamento de sessão e é um componente separado. O SIP trabalha com RTP, mas não o impõe. Trabalha com codecs de voz G.711 e codecs de vídeo QCIF H.261, mas não os impõe. Pode ser combinado com outros protocolos e serviços.
- O H.323 vem da ITU (telefonia), enquanto o SIP vem da IETF e toma emprestado muitos conceitos da Web, do DNS e do e-mail da Internet.
- O H.323, como é um padrão guarda-chuva, é grande e complexo. O SIP usa o princípio KISS: mantenha a simplicidade, seu ignorante (keep it simple, stupid)!

Há uma excelente discussão sobre protocolos H.323, SIP e de voz sobre IP em geral, em [Hersent, 2000].

7.5 Distribuição de multimídia: redes de distribuição de conteúdo

Com taxas de vídeo em fluxo contínuo na faixa de centenas de kbps para vídeos de baixa resolução até vários Mbps para vídeo em DVD, a tarefa de transmitir um vídeo armazenado em fluxo contínuo, sob demanda, a um grande número de usuários distribuídos geograficamente é um desafio e tanto! A abordagem mais simples seria armazenar o vídeo em um único servidor e simplesmente transmiti-lo de um servidor de vídeo (ou de uma unidade lógica de processamento corporativo — *server farm*) a um cliente, para cada requisição de cliente, como foi discutido na Seção 7.2. Mas há dois problemas óbvios nessa solução. Em primeiro lugar, como um cliente pode estar muito distante do servidor, os pacotes de servidor para o cliente podem passar por muitos ISPs, aumentando a probabilidade de atrasos e perdas significativos. Em segundo lugar, se o vídeo for muito popular, provavelmente será enviado muitas vezes através dos mesmos ISPs (e pelos mesmos enlaces de comunicação), consumindo, por conseguinte, largura de banda significa-

tiva. No Capítulo 2 discutimos como o cache pode atenuar esses problemas. Embora tenhamos discutido cache em termos do conteúdo tradicional da Web, deve ficar claro que ele também é apropriado para conteúdo de multimídia, como áudio e vídeo armazenados. Nesta seção, discutiremos **redes de distribuição de conteúdo** (*content distribution networks* — CDNs), que oferecem uma abordagem alternativa à distribuição de conteúdo de multimídia armazenado (bem como à distribuição de conteúdo Web tradicional).

A filosofia básica das CDNs é que, se o cliente não pode vir até o conteúdo (porque o caminho de melhor esforço do cliente ao servidor não pode suportar vídeo de fluxo contínuo), então o conteúdo deve ser levado até o cliente. Assim, CDNs usam um modelo diferente do cache na Web. Para uma CDN, os clientes pagantes não são mais as ISPs, mas as provedoras de conteúdo. Uma provedora de conteúdo que tenha um vídeo para distribuir (tal como a CNN) paga a uma empresa CDN (como a Akamai) para levar seu vídeo até os usuários requisitantes com os menores atrasos possíveis.

Uma empresa CDN tipicamente provê seu serviço de distribuição de conteúdo da seguinte maneira:

1. A empresa de CDN instala centenas de **servidores CDN** por toda a Internet. Em geral a empresa instala seus servidores CDN em uma **central de dados**. Uma central de dados, de propriedade de terceiros e executada por terceiros, normalmente é um edifício cheio de hospedeiros servidores. Essas centrais de dados muitas vezes estão em ISPs de nível mais baixo, próximas às redes de acesso ISP e aos clientes.
2. A CDN duplica o conteúdo de seu cliente nos servidores CDN. Sempre que seu cliente atualiza seu conteúdo, a CDN redistribui o conteúdo renovado aos servidores CDN.
3. A empresa CDN provê um mecanismo que entrega, a um cliente Web que tenha requisitado, conteúdo pelo servidor CDN que melhor possa atendê-lo. Esse servidor pode ser o que estiver mais próximo do cliente (talvez no mesmo ISP) ou pode ser um servidor CDN cujo caminho até o cliente está livre de congestionamento.

É interessante notar que há muitas empresas independentes envolvidas no paradigma CDN. Uma provedora de conteúdo (tal como a CNN) distribui seu conteúdo por meio de uma empresa CDN (como a Akamai). A empresa CDN compra servidores CDN de um fabricante desses equipamentos (como Cisco ou IBM) e os instala em centrais de hospedagem de propriedade de uma empresa de hospedagem (como a AT&T). Assim, sem contar as ISPs, há quatro empresas independentes envolvidas no paradigma CDN!

A Figura 7.16 mostra a interação entre a provedora de conteúdo e a empresa CDN. A provedora de conteúdo primeiro determina qual de seus objetos (por exemplo, vídeos) ela quer que a CDN distribua. (A provedora de conteúdo distribui os objetos remanescentes sem a intervenção da CDN.) A provedora de conteúdo rotula o conteúdo e então o envia para um nó CDN, que, por sua vez, duplica esse conteúdo e o retransmite para todos os seus servidores CDN. A empresa CDN pode ser proprietária de uma rede privada que levará o conteúdo do nó CDN até os servidores CDN. Sempre que a provedora de conteúdo modifica um objeto distribuído pela CDN, ela envia a versão atualizada ao nó CDN, que, mais uma vez imediatamente duplica e distribui o objeto aos servidores CDN. É importante ter em mente que cada servidor CDN normalmente contém objetos de muitas provedoras de conteúdo.

Agora surge a pergunta interessante. Quando um browser no hospedeiro de um usuário recebe instruções para extrair um objeto específico (identificado por um URL), como o browser determina se deve extrair o objeto do servidor de origem ou de um dos servidores CDN? Normalmente as CDNs utilizam redirecionamento DNS para guiar os browsers até o servidor correto [Kangasharju, 2000].

Suponha, por exemplo, que o nome de hospedeiro da servidora de conteúdo seja `www.foo.com`. Suponha que o nome da empresa CDN seja `cdn.com`. Suponha ainda que a provedora de conteúdo quer que somente seus arquivos de vídeo mpeg sejam distribuídos pela CDN; todos os outros objetos, incluindo as páginas de base HTML, são distribuídos diretamente pela provedora de conteúdo. Para conseguir isso, a provedora de conteúdo modifica todos os objetos HTML no servidor de origem de modo que os URLs dos arquivos de vídeo tenham o prefixo `http://www.cdn.com`. Assim, se um arquivo HTML na provedora de conteúdo

tinha, originalmente, uma referência a `http://www.foo.com/sports/ruth.mpg`, a provedora de conteúdo rotularia esse objeto substituindo a referência no arquivo HTML por `http://www.cdn.com/www.foo.com/sports/ruth.mpg`.

Quando um browser requisita uma página Web que contém a imagem ruth.mpg, ocorrem as seguintes ações:

1. O browser envia sua requisição para o objeto base HTML ao servidor de origem `www.foo.com`, que envia o objeto HTML requisitado ao browser. O browser analisa o arquivo HTML e encontra a referência a `http://www.cdn.com/www.foo.com/sports/ruth.mpg`.
2. O browser então faz uma consulta DNS em `www.cdn.com`, que é o nome de hospedeiro para o URL referenciado. O DNS é configurado de modo que todas as consultas sobre `www.cdn.com` que chegarem a um servidor DNS raiz sejam enviadas a um servidor DNS com autoridade para `www.cdn.com`. Quando o servidor DNS com autoridade recebe a consulta, extrai o endereço IP do browser requisitante. Usando um mapa interno da rede que construiu para toda a Internet, o servidor DNS da CDN retorna o endereço IP do servidor CDN que parece ser o melhor para o browser requisitante (muitas vezes é o servidor CDN mais próximo do browser).
3. O DNS no cliente requisitante recebe uma resposta DNS com o endereço IP. O browser então envia sua requisição HTTP ao servidor CDN que tem aquele endereço IP. O browser obtém ruth.mpg de seu servidor CDN. Para requisições subseqüentes de `www.cdn.com`, o cliente continua a usar o mesmo servidor CDN, já que o endereço IP para `www.cdn.com` está no cache DNS (no hospedeiro cliente ou no servidor local de nomes DNS).

Em suma, como ilustrado na Figura 7.17, o hospedeiro requisitante primeiro vai ao servidor Web de origem para obter o objeto base HTML, em seguida vai até o servidor DNS com autoridade pertencente à CDN para obter o endereço IP do melhor servidor CDN e, finalmente, vai até aquele servidor CDN para obter o vídeo. Note que, para implementar esse esquema de distribuição, não é preciso fazer nenhuma mudança no HTTP, no DNS nem no browser.

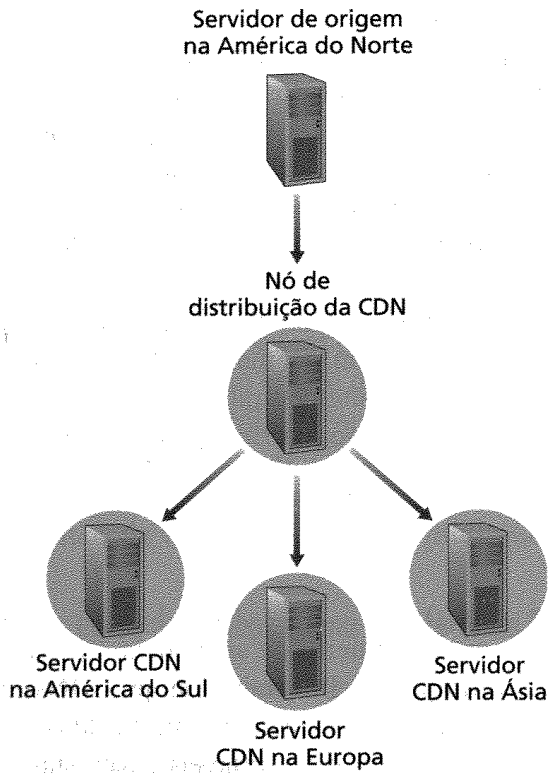


Figura 7.16 A CDN envia a seus servidores CDN objetos rotulados da provedora de conteúdo

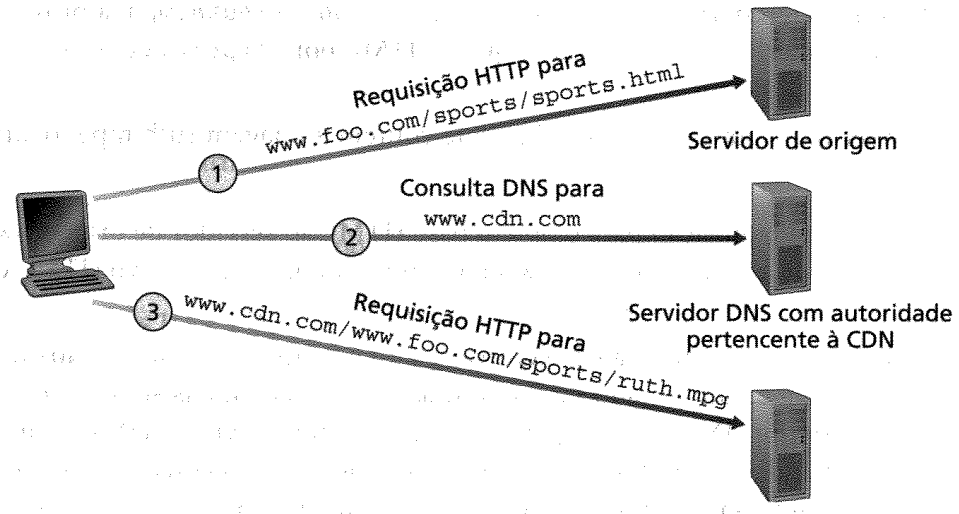


Figura 7.17 A CDN usa DNS para direcionar requisições a um servidor CDN próximo

Ainda resta explicar como uma empresa CDN determina o 'melhor' servidor CDN para o hospedeiro requisitante. Embora cada empresa CDN tenha seu modo proprietário exclusivo para fazer isso, não é difícil ter uma idéia aproximada do que elas fazem. A empresa CDN monitora o melhor servidor CDN para cada acesso ISP na Internet que contenha clientes requisitantes potenciais. Então a CDN determina o melhor servidor CDN com base em seu conhecimento das tabelas de roteamento da Internet (especificamente, as tabelas BGP, que discutimos no Capítulo 4), em estimativas de tempo de viagem de ida e volta e em outros dados de medição que possui, de seus vários servidores a várias redes de acesso.

Se o leitor quiser uma discussão mais detalhada, pode consultar [Verma, 2001]. Dessa maneira, a CDN estima qual servidor CDN provê o melhor serviço de melhor esforço para o ISP. A CDN faz isso para um grande número de acessos ISPs na Internet e utiliza essa informação para configurar o servidor DNS com autoridade.

CDNs também são freqüentemente implantadas por empresas que têm um grande número de escritórios regionais. Em geral, esses escritórios são conectados com a Internet (ou com uma rede privada) por enlaces relativamente lentos (por exemplo, enlaces T1 a 1,544 Mbps). Hoje, muitas dessas empresas distribuem a seus empregados vídeos de aprendizagem eletrônica (e-learning) ou outros vídeos com material corporativo. Para realizar essa tarefa, a empresa compra servidores CDN e os instala em seus escritórios regionais. Quando tem um novo vídeo para distribuir, ela o transmite a todos os seus escritórios regionais e, então, os empregados recebem o vídeo por meio de seu servidor CDN regional. Desse modo, cada vídeo é enviado pelo enlace de acesso de cada escritório regional apenas uma vez, sem sobrecarregar esse enlace com várias distribuições dos mesmos vídeos. Além do mais, CDNs empresariais muitas vezes enviam o vídeo aos servidores CDNs regionais a partir do site central da empresa, quando o vídeo que estão distribuindo tem grande largura de banda, de modo a não interferir com outros tráfegos importantes, por exemplo, à noite.

7.6 Além do melhor esforço

Nas seções anteriores, aprendemos como números de seqüência, marcas de tempo, FEC, RTP e H.323 podem ser usados por aplicações de multimídia na Internet de hoje. CDNs representam uma solução para distribuição de conteúdo de multimídia no âmbito do sistema inteiro. Mas será que somente essas técnicas são suficientes para suportar aplicações de multimídia confiáveis e robustas, como um serviço de telefonia IP equivalente ao serviço atualmente oferecido pela rede telefônica? Antes de responder a essa pergunta,

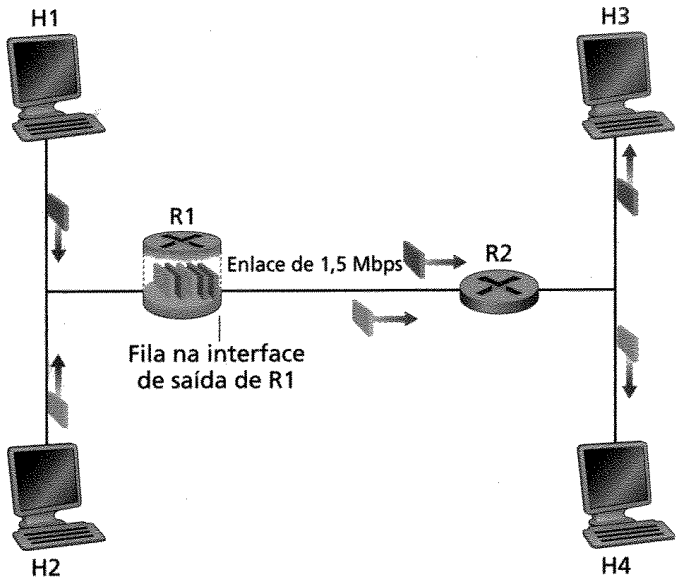


Figura 7.18 Uma rede simples com duas aplicações

vamos lembrar novamente que a Internet de hoje oferece um serviço de melhor esforço a todas as suas aplicações, isto é, ela nada promete quanto à qualidade de serviço (QoS) que uma aplicação receberá. Uma aplicação receberá o nível de desempenho (por exemplo, atraso fim-a-fim e perda de pacotes) que a rede estiver capacitada a dar em determinado momento. Lembre-se também de que a Internet pública de hoje não permite que aplicações de multimídia sensíveis ao atraso requisitem algum tratamento especial. Como todos os pacotes recebem igual tratamento dos roteadores, incluindo os pacotes de áudio e vídeo sensíveis ao atraso, para arruinar a qualidade de uma chamada telefônica IP em andamento basta haver suficiente tráfego interferente (isto é, congestionamento na rede) que aumente de modo significativo o atraso e a perda percebidos por uma chamada telefônica IP.

Nesta seção, identificaremos novos componentes arquitetônicos que podem ser adicionados à arquitetura da Internet para proteger uma aplicação de tal congestionamento e, assim, transformar em realidade aplicações de multimídia de alta qualidade em rede. Muitos dos assuntos que discutiremos nesta e nas seções restantes deste capítulo são ou foram, recentemente, alvo de intensa discussão nos grupos de trabalho Diffserv, Intserv e RSVP da IETF

A Figura 7.18 mostra um cenário simples de rede que usaremos para ilustrar os componentes arquitetônicos mais importantes que têm sido propostos com a finalidade de fornecer suporte explícito para as necessidades de QoS de aplicações de multimídia. Suponha que dois fluxos de pacotes de aplicação se originem nos hospedeiros H1 e H2 em uma LAN e sejam destinados aos hospedeiros H3 e H4 em outra LAN. Os roteadores nas duas LANs são ligados por um enlace de 1,5 Mbps. Vamos admitir que as taxas das LANs sejam significativamente mais altas do que 1,5 Mbps e vamos focalizar a fila de saída do roteador R1; é aqui que ocorrerão atraso e perda de pacotes se a taxa agregada de envio de H1 e H2 exceder 1,5 Mbps. Vamos agora considerar diversos cenários; cada um deles nos dará importantes percepções sobre os princípios subjacentes ao fornecimento de garantias de QoS a aplicações de multimídia.

7.6.1 Cenário 1: uma aplicação de áudio de 1 Mbps e uma transferência FTP

O cenário 1 está ilustrado na Figura 7.19. Nessa figura, uma aplicação de áudio de 1 Mbps (por exemplo, uma chamada de áudio com qualidade de CD) compartilha o enlace de 1,5 Mbps entre R1 e R2 com uma aplicação FTP que está transferindo um arquivo de H2 para H4. Na Internet de melhor esforço, os pacotes de áudio e FTP são misturados na fila de saída de R1 e (caracteristicamente) transmitidos na ordem primeiro a entrar/primeiro a sair (*first-in-first-out* — FIFO). Nesse cenário, uma rajada de pacotes da fonte

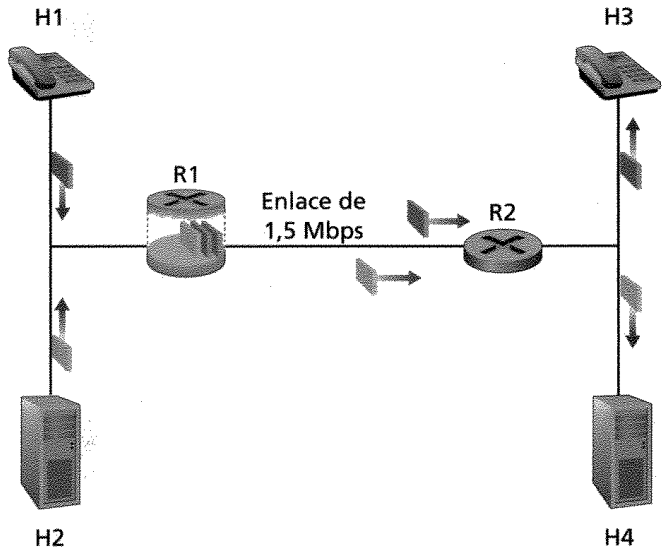


Figura 7.19 Aplicações concorrentes de áudio e FTP

FTP poderia potencialmente lotar a fila, fazendo com que pacotes IP de áudio sofressem atraso excessivo ou perda, devido ao transbordamento do buffer de R1. Como resolveríamos esse problema potencial? Dado que a aplicação FTP não tem limitação de tempo, poderíamos, por intuição, dar prioridade estrita aos pacotes de áudio em R1. Sob uma disciplina restritiva de escalonamento por prioridade, um pacote IP de áudio no buffer de saída de R1 seria sempre transmitido antes de qualquer pacote FTP nesse mesmo buffer de saída de R1. O enlace de R1 a R2 pareceria um enlace dedicado de 1,5 Mbps para o tráfego de áudio e o tráfego FTP usaria o enlace R1–R2 somente quando não houvesse nenhum tráfego de áudio na fila.

Para R1 distinguir os pacotes de áudio dos pacotes FTP em sua fila, cada pacote deve ser marcado como pertencente a uma dessas duas classes de tráfego. Lembre-se de que esse era o objetivo final do campo de tipo de serviço (*Type-of-service* — ToS) do IPv4 (veja a Seção 4.4.1). Por mais óbvio que pareça, esse é, então, nosso primeiro princípio subjacente para o fornecimento de garantias de qualidade de serviço:

Princípio 1: a marcação de pacotes permite que um roteador faça a distinção de pacotes pertencentes a diferentes classes de tráfego.

7.6.2 Cenário 2: uma aplicação de áudio de 1 Mbps e uma transferência FTP de alta prioridade

Nosso segundo cenário é apenas ligeiramente diferente do cenário 1. Suponha agora que o usuário FTP comprou de seu ISP um serviço de acesso à Internet de alto preço, ao passo que o usuário de áudio comprou um serviço de Internet barato, cujo preço é uma minúscula fração do outro serviço. Nesse caso, os pacotes de áudio do usuário do serviço barato deveriam ter prioridade em relação aos pacotes FTP? Pode-se afirmar que não. Aqui, o mais razoável seria distinguir pacotes com base no endereço IP do remetente. De modo mais geral, vemos que é necessário que um roteador classifique pacotes segundo algum critério, o que exige uma ligeira modificação no princípio 1:

Princípio 1 (modificado): a classificação de pacotes permite que um roteador faça a distinção de pacotes pertencentes a diferentes classes de tráfego.

A marcação explícita de pacotes é uma maneira pela qual os pacotes podem ser distinguidos. Contudo, a marca carregada por um pacote não determina por si só que o pacote receberá uma dada qualidade de serviço. A marcação é apenas um dos mecanismos para distinguir pacotes. O modo como um roteador distingue pacotes tratando-os de maneira diferenciada é uma decisão administrativa.

7.6.3 Cenário 3: uma aplicação de áudio malcomportada e uma transferência FTP

Suponha agora que, de alguma maneira (pela utilização de mecanismos que estudaremos nas seções subseqüentes), o roteador saiba que deve dar prioridade a pacotes da aplicação de áudio de 1 Mbps. Como a taxa de saída do enlace é de 1,5 Mbps, mesmo que os pacotes FTP recebam prioridade mais baixa, ainda assim receberão, na média, um serviço de transmissão de 0,5 Mbps. Mas o que acontece se a aplicação de áudio começar a enviar pacotes a uma taxa de 1,5 Mbps ou mais alta (seja com má intenção, seja devido a um erro de aplicação)? Nesse caso, os pacotes FTP morrerão por inanição, isto é, não receberão nenhum serviço no enlace R1–R2. Problemas semelhantes ocorreriam se várias aplicações (por exemplo, várias chamadas de áudio), todas com a mesma prioridade, estivessem compartilhando a largura de banda de um enlace; um fluxo malcomportado poderia degradar e arruinar o desempenho dos outros fluxos. Idealmente, o que se quer é um certo grau de isolamento entre fluxos, para proteger um fluxo do mau comportamento de outro fluxo. Este é o segundo princípio subjacente ao fornecimento de garantias de QoS.

Princípio 2: é desejável fornecer um certo grau de isolamento entre os fluxos de tráfego, de modo que um fluxo não seja afetado adversamente por outro fluxo de comportamento inadequado.

Na próxima seção, examinaremos diversos mecanismos específicos que fornecem isolamento entre fluxos. Observamos aqui que podem ser adotadas duas abordagens amplas. Com a primeira, é possível regular fluxos de tráfego, como mostra a Figura 7.20. Se um fluxo de tráfego deve se ajustar a certos critérios (por exemplo, o fluxo de áudio não deve exceder uma taxa de pico de 1 Mbps), então um mecanismo de regulação pode ser introduzido para assegurar que esse critério seja, de fato, observado. Se a aplicação regulada se comportar mal, o mecanismo de regulação executará alguma ação (por exemplo, descartará ou atrasará pacotes que estão violando o critério), de modo que o tráfego que está entrando na rede obedeça aos critérios. O mecanismo do tipo leaky bucket ('balde furado'), que examinaremos na seção seguinte, talvez seja o mecanismo de regulação mais amplamente usado. Na Figura 7.20, o mecanismo de classificação e marcação de pacotes (Princípio 1) e o mecanismo de regulação (Princípio 2) estão localizados conjuntamente na borda da rede, seja no sistema final, seja em um roteador de borda.

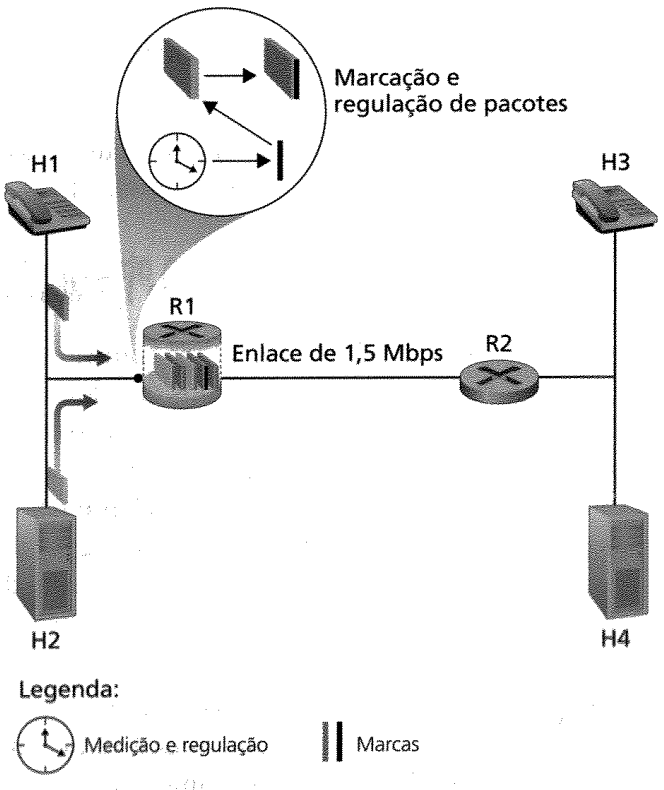


Figura 7.20 Regulação (e marcação) de fluxos de áudio e FTP

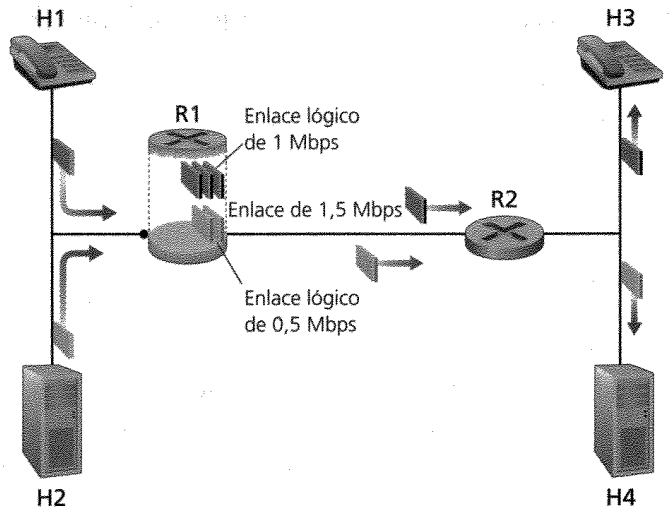


Figura 7.21 Isolamento lógico de fluxos de aplicações de áudio e de FTP

Uma abordagem alternativa para prover isolamento entre os fluxos de tráfego é deixar para o mecanismo de programação de pacotes na camada de enlace a tarefa de alocar explicitamente uma porção fixa da largura de banda de enlace a cada fluxo de aplicação. Por exemplo, poderia ser alocado em R_1 1 Mbps ao fluxo de áudio e 0,5 Mbps ao fluxo FTP. Nesse caso, os fluxos de áudio e FTP perceberiam um enlace lógico com capacidade de 1 Mbps e 0,5 Mbps, respectivamente, como mostra a Figura 7.21.

Com imposição estrita de alocação de largura de banda na camada de enlace, um fluxo pode usar apenas a quantidade de largura de banda que lhe foi alocada; em particular, ele não pode utilizar largura de banda que não está sendo usada correntemente pelas outras aplicações. Por exemplo, se o fluxo de áudio silenciasse (se o interlocutor fizesse uma pausa e não gerasse nenhum pacote de áudio, por exemplo), ainda assim o fluxo FTP não conseguiria transmitir mais do que 0,5 Mbps pelo enlace R1–R2, mesmo que a alocação de largura de banda de 1 Mbps para o fluxo de áudio não estivesse sendo utilizada naquele momento. Por conseguinte, é desejável usar a largura de banda da maneira mais eficiente possível, permitindo que um fluxo utilize a largura de banda não usada por outro fluxo a qualquer instante. Este é o terceiro princípio subjacente ao fornecimento de qualidade de serviço:

Princípio 3: ao fornecer isolamento de fluxos, é desejável que se usem os recursos (por exemplo, largura de banda de enlace e buffers) da maneira mais eficiente possível.

7.6.4 Cenário 4: duas aplicações de áudio de 1 Mbps por um enlace de 1,5 Mbps sobrecarregado

Em nosso último cenário, duas conexões de áudio de 1 Mbps transmitem seus pacotes pelo enlace de 1,5 Mbps, como mostra a Figura 7.22. A taxa combinada de dados dos dois fluxos (2 Mbps) excede a capacidade do enlace. Mesmo com classificação e marcação (Princípio 1), isolamento de fluxos (Princípio 2) e compartilhamento de largura de banda não utilizada (Princípio 3), no caso, inexistente, nosso objetivo é claramente impossível. Simplesmente não há largura de banda suficiente para atender às necessidades das aplicações. Se as duas aplicações compartilhassem eqüitativamente a largura de banda, cada uma receberia apenas 0,75 Mbps. Visto de outro modo, cada aplicação perderia 25 por cento de seus pacotes transmitidos. Esse é um serviço cuja qualidade é tão inaceitavelmente baixa, que a aplicação é totalmente inútil; não é preciso nem mesmo transmitir nenhum pacote de áudio.

Para um fluxo que requer um mínimo de qualidade de serviço para ser considerado útil há duas alternativas: ou a rede permite o fluxo, ou o bloqueia. A rede telefônica é um exemplo de rede que realiza esse bloqueio de chamada — se os recursos exigidos (um circuito fim-a-fim, no caso da rede de telefonia) não puderem ser alocados à chamada, esta será bloqueada (impedida de entrar na rede) e o usuário ouvirá um

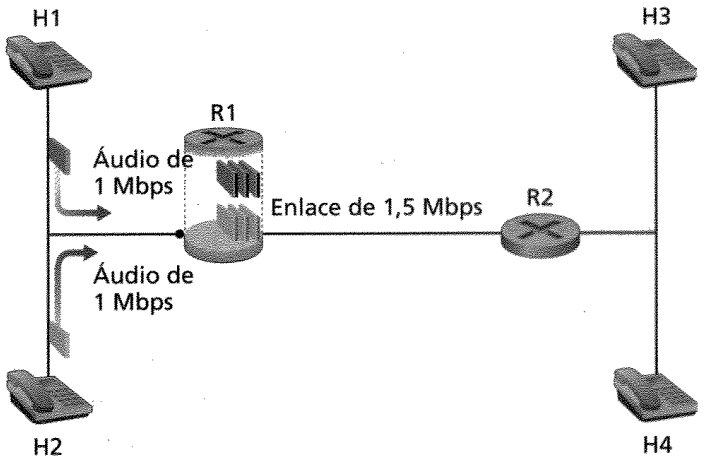


Figura 7.22 Duas aplicações de áudio competindo e sobrecarregando o enlace R1–R2

sinal de ocupado. Em nosso exemplo anterior, não há nada a ganhar ao permitir que um fluxo entre na rede se ele não receber QoS suficiente a ponto de ser considerado útil. Na verdade, há um custo quando se aceita um fluxo que não recebe a QoS de que necessita, pois os recursos da rede estarão sendo usados para suportar um fluxo que nenhuma utilidade trará ao usuário final.

Implícita à necessidade de fornecer QoS garantida a um fluxo está a necessidade de o fluxo declarar suas exigências de QoS. Esse processo de fazer com que um fluxo declare suas exigências de QoS e, em seguida, fazer com que a rede ou aceite o fluxo (à QoS exigida) ou o bloqueie é denominado processo de aceitação de chamada. A necessidade da aceitação de chamada é o quarto princípio subjacente ao fornecimento de garantias de QoS:

> **Princípio 4:** se nem sempre haverá recursos suficientes disponíveis, é preciso um processo de aceitação de chamada pelo qual os fluxos declarem suas exigências de QoS e a rede os aceite (à QoS exigida), ou os bloqueie (se a QoS exigida não puder ser fornecida pela rede).

Nessa nossa discussão, identificamos, até aqui, quatro princípios básicos do fornecimento de garantias de QoS às aplicações de multimídia. Na seção seguinte, consideraremos vários mecanismos de implementação desses princípios. Nas seções subseqüentes a ela, examinaremos os modelos de serviço da Internet propostos para fornecimento de garantias de QoS.

7.7 Mecanismos de escalonamento e regulação

Na seção anterior, identificamos os princípios subjacentes importantes para o fornecimento de QoS às aplicações de rede multimídia. Nesta seção, examinaremos vários mecanismos que são usados para fornecer essas garantias de QoS.

7.7.1 Mecanismos de escalonamento

Lembre-se de que dissemos na Seção 1.6 e na Seção 4.3 que pacotes pertencentes a vários fluxos de rede são multiplexados e enfileirados para transmissão nos buffers de saída associados a um enlace. O modo como os pacotes enfileirados são selecionados para transmissão pelo enlace é conhecido como **disciplina de escalonamento do enlace**. Vimos na seção anterior que a disciplina de escalonamento do enlace desempenha um importante papel no fornecimento de garantias de QoS. Vamos agora examinar mais detalhadamente algumas das mais importantes disciplinas de escalonamento.

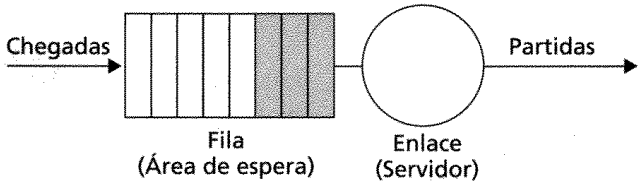


Figura 7.23 Representação do enfileiramento FIFO

Primeiro a entrar/primeiro a sair (FIFO)

A Figura 7.23 mostra a representação do modelo de enfileiramento para a disciplina de escalonamento de enlace primeiro a entrar/primeiro a sair. Pacotes que chegam à fila de saída do enlace esperam pela transmissão se, naquele momento, o enlace estiver ocupado com a transmissão de outro pacote. Se não houver espaço de buffer suficiente para guardar o pacote que chega, a **política de descarte de pacotes** da fila então determinará se o pacote será descartado (perdido) ou se outros pacotes serão retirados da fila para dar espaço ao pacote que está chegando. Em nossa discussão a seguir, vamos ignorar o descarte de pacotes. Quando um pacote é transmitido integralmente pelo enlace de saída (isto é, recebe serviço), ele é retirado da fila.

A disciplina de escalonamento FIFO — também conhecida como FCFS (*first-come-first-served* — primeiro a chegar/primeiro a ser atendido) — seleciona pacotes para transmissão pelo enlace na mesma ordem em que eles chegaram à fila de saída do enlace. Todos estamos familiarizados com as filas FIFO em pontos de ônibus (particularmente na Inglaterra, onde parece que as filas atingiram a perfeição) ou em outras centrais de serviços, onde os clientes que chegam se juntam ao final da fila de espera, permanecem na ordem e então são atendidos quando chegam ao início da fila.

A Figura 7.24 mostra a fila FIFO em operação. As chegadas de pacotes são indicadas por setas numeradas acima da linha temporal superior; os números indicam a ordem em que os pacotes chegaram. As saídas de pacotes individuais são mostradas abaixo da linha temporal inferior. O tempo que um pacote passa no atendimento (sendo transmitido) é indicado pelo retângulo sombreado entre as duas linhas de tempo. Por causa da disciplina FIFO, os pacotes saem na mesma ordem em que chegaram. Note que, após a saída do pacote 4, o enlace permanece ocioso (uma vez que os pacotes 1 a 4 já foram transmitidos e retirados da fila) até a chegada do pacote 5.

Enfileiramento prioritário

Sob a **regra do enfileiramento prioritário**, pacotes que chegam ao enlace de saída são classificados em classes de prioridade na fila de saída, como mostra a Figura 7.25. Como discutimos na seção anterior, a classe de prioridade de um pacote pode depender de uma marca explícita que ele carrega em seu cabeçalho de pacote (por exemplo, o valor dos bits de ToS em um pacote IPv4), do endereço IP de sua origem ou destino, do número de porta de seu destino ou de outro critério. Cada classe de prioridade tipicamente tem sua própria fila. Ao escolher um pacote para transmitir, a disciplina de enfileiramento prioritário transmitirá um pacote da classe de prioridade mais alta cuja fila não esteja vazia (isto é, tenha pacotes esperando transmissão). A escolha entre pacotes da mesma classe de prioridade é feita, tipicamente, pelo método FIFO.

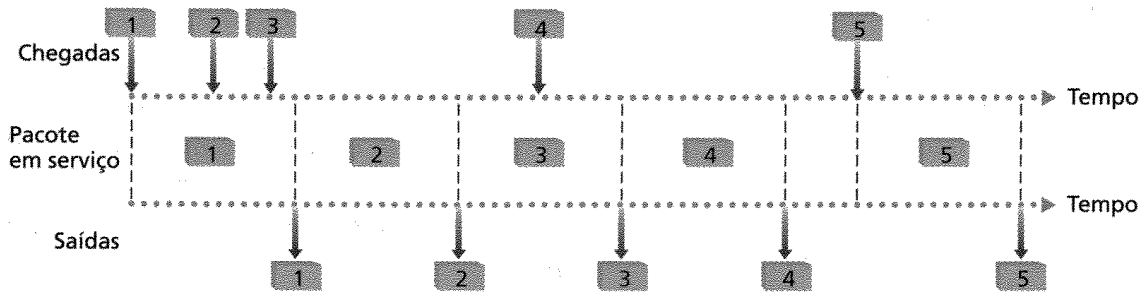


Figura 7.24 A fila FIFO em operação

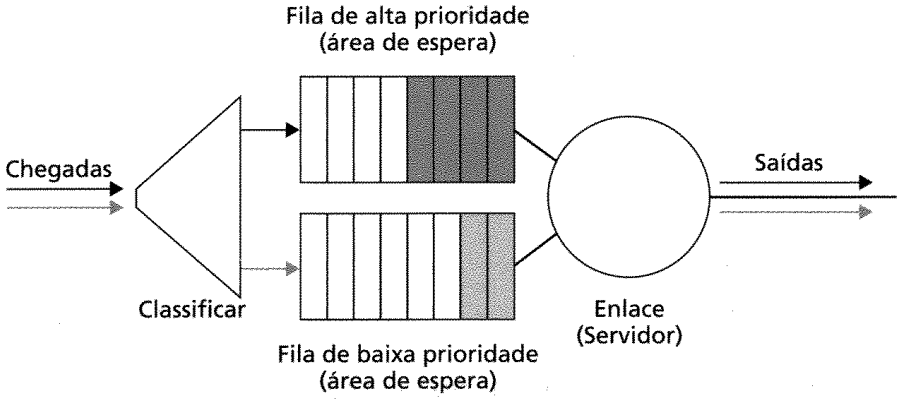


Figura 7.25 Modelo de enfileiramento prioritário

A Figura 7.26 ilustra a operação de uma fila prioritária com duas classes de prioridade. Os pacotes 1, 3 e 4 pertencem à classe de alta prioridade e os pacotes 2 e 5, à classe de baixa prioridade. O pacote 1 chega e, encontrando o enlace vazio, inicia a transmissão. Durante a transmissão do pacote 1, os pacotes 2 e 3 chegam e são colocados nas filas de baixa prioridade e alta prioridade, respectivamente. Após a transmissão do pacote 1, o pacote 3 (um pacote de alta prioridade) é selecionado para transmissão, passando à frente do pacote 2 (que, mesmo tendo chegado mais cedo, é um pacote de baixa prioridade). Ao término da transmissão do pacote 3, começa a transmissão do pacote 2 (um pacote de baixa prioridade). O pacote 4 (um pacote de alta prioridade) chega durante a transmissão do pacote 2 (um pacote de baixa prioridade). Sob uma disciplina de enfileiramento prioritário não preemptiva, a transmissão de um pacote não será interrompida se já tiver começado. Nesse caso, o pacote 4 entra na fila para transmissão e começa a ser transmitido após a conclusão da transmissão do pacote 2.

Varredura cíclica e WQF (enfileiramento justo ponderado)

Sob a **disciplina de enfileiramento por varredura cíclica**, pacotes são classificados do mesmo modo que no enfileiramento prioritário. Contudo, em vez de haver uma prioridade estrita de serviço entre as classes, um escalonador de varredura cíclica alterna serviços entre as classes. Na forma mais simples de escalonamento por varredura cíclica, um pacote de classe 1 é transmitido, seguido por um pacote de classe 2, seguido por um pacote de classe 1, seguido por um pacote de classe 2 e assim por diante. Uma disciplina de enfileiramento de conservação de trabalho nunca permitirá que o enlace fique ocioso enquanto houver pacotes (de qualquer classe) enfileirados para transmissão. Uma **disciplina de varredura cíclica de conservação de trabalho** que procura um pacote de uma dada classe, mas não encontra nenhum, verifica imediatamente a classe seguinte na seqüência da varredura cíclica.

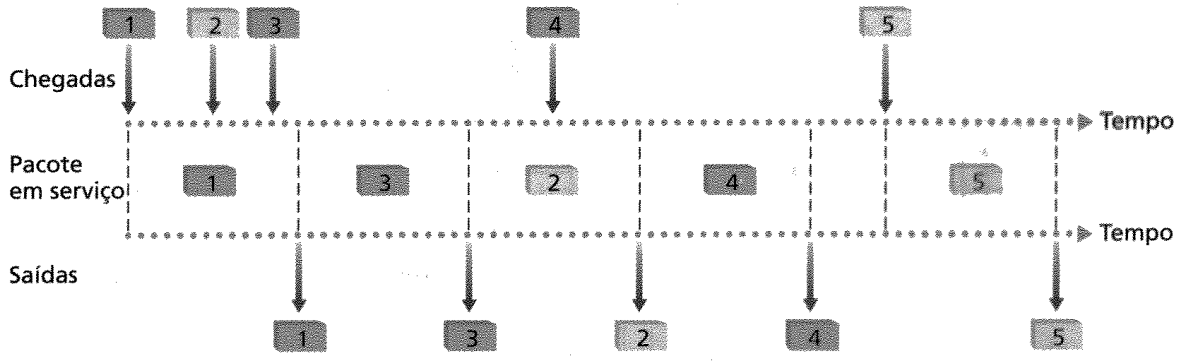


Figura 7.26 Operação do enfileiramento prioritário

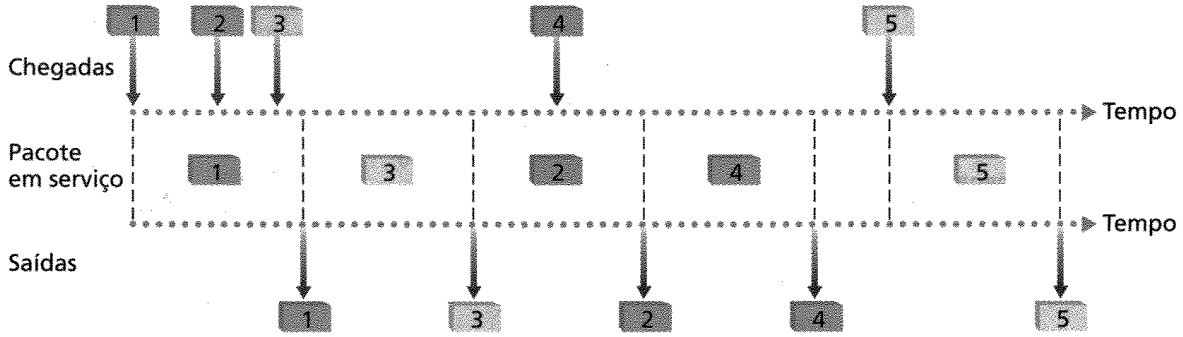


Figura 7.27 Operação de enfileiramento de duas classes por varredura cíclica

A Figura 7.27 ilustra a operação de uma fila de duas classes por varredura cíclica. Nesse exemplo, os pacotes 1, 2 e 4 pertencem à classe 1 e os pacotes 3 e 5, à classe 2. O pacote 1 inicia a transmissão imediatamente após sua chegada à fila de saída. Os pacotes 2 e 3 chegam durante a transmissão do pacote 1 e, assim, entram na fila para transmissão. Após a transmissão do pacote 1, o escalonador de enlace procura um pacote de classe 2 e, então, transmite o pacote 3. Após a transmissão do pacote 3, o escalonador procura um pacote de classe 1 e, então, transmite o pacote 2. Após a transmissão do pacote 2, o pacote 4 é o único na fila; assim, ele é transmitido imediatamente após o pacote 2.

Uma abstração generalizada do enfileiramento por varredura cíclica que encontrou considerável utilização nas arquiteturas com QoS é a denominada disciplina de **enfileiramento justo ponderado** (*weighted fair queuing* — WFQ) [Demers, 1990; Parekh, 1993]. A WFQ é ilustrada na Figura 7.28. Os pacotes que chegam são classificados e enfileirados por classe em suas áreas de espera apropriadas. Como acontece no escalonamento por varredura cíclica, um programador WFQ atende às classes de modo cíclico — atende primeiramente à classe 1, depois à classe 2 e, em seguida, à classe 3; e então (admitindo que haja três classes) repete o esquema de serviço. A WFQ também é uma disciplina de enfileiramento de conservação de trabalho; assim, ao encontrar uma classe de fila vazia, ela imediatamente passa para a classe seguinte na seqüência de atendimento.

A WFQ é diferente da varredura cíclica, pois cada classe pode receber uma quantidade de serviço diferenciado a qualquer intervalo de tempo. Em particular, a cada classe i é atribuído um peso w_i. A WFQ garante que, em qualquer intervalo de tempo durante o qual houver pacotes da classe i para transmitir, a classe i receberá uma fração de serviço igual a $w_i/(\Sigma w_j)$, onde o denominador é a soma de todas as classes que também têm pacotes enfileirados para transmissão. No pior caso, mesmo que todas as classes tenham pacotes na fila, a classe i ainda terá garantido o recebimento de uma fração $w_i/(\Sigma w_j)$ da largura de banda. Assim, para um enlace com taxa de transmissão R, a classe i sempre conseguirá uma vazão de, no mínimo, $R \cdot w_i/(\Sigma w_j)$.

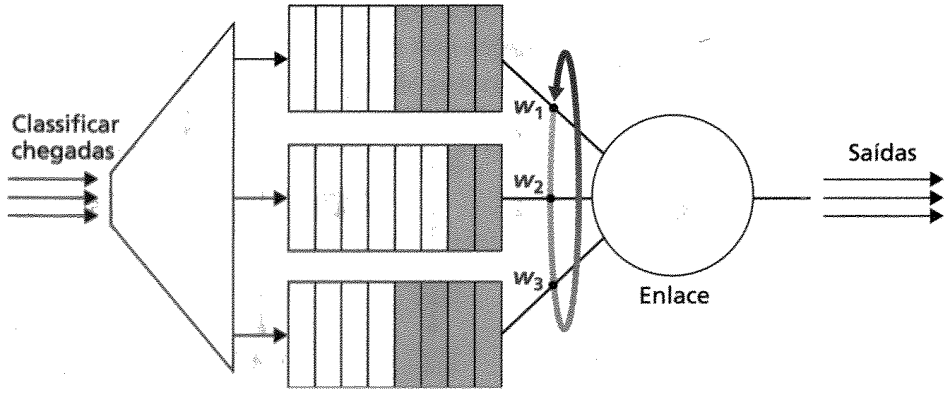


Figura 7.28 Enfileiramento justo ponderado (WFQ)

Descrevemos a WFQ em condições ideais, pois não consideramos o fato de que os pacotes são unidades discretas de dados e que a transmissão de um pacote não será interrompida para dar início à transmissão de outro; [Demers, 1990] e [Parekh, 1993] discutem essa questão do empacotamento. Como veremos nas seções seguintes, a WFQ tem um papel fundamental nas arquiteturas com QoS. Ela também está disponível nos roteadores fabricados atualmente [Cisco QoS, 2002]. (Intranets que usam roteadores que dispõem de WFQ podem, portanto, fornecer QoS a seus fluxos internos.)

7.7.2 Regulação: o leaky bucket

Na Seção 7.6, identificamos a regulação, o ajuste da taxa com a qual é permitido que um fluxo injete pacotes na rede, como uma das pedras fundamentais de qualquer arquitetura com QoS. Mas quais características da taxa de pacotes de um fluxo devem ser reguladas? Podemos identificar três critérios importantes de regulação diferentes entre si pela escala de tempo durante a qual o pacote é regulado:

- *Taxa média.* A rede pode desejar limitar a taxa média durante um período de tempo (pacotes por intervalo de tempo) com a qual os pacotes de um fluxo podem ser enviados para dentro da rede. Uma questão crucial aqui é o intervalo de tempo durante o qual a taxa média será regulada. Um fluxo cuja taxa média está limitada a 100 pacotes por segundo é mais restringido do que uma fonte limitada a 6.000 pacotes por minuto, mesmo que ambos tenham a mesma taxa média durante um intervalo de tempo suficientemente longo. Por exemplo, a última limitação permitiria que um fluxo enviasse 1.000 pacotes em um dado intervalo de tempo de um segundo de duração (sujeito à limitação de que a taxa seja menor do que 6.000 pacotes durante um intervalo de 1 minuto de duração que contenha esses 1.000 pacotes), enquanto a limitação anterior desautorizaria esse comportamento de envio.
- *Taxa de pico.* Enquanto a limitação da taxa média restringe a quantidade de tráfego que pode ser enviada para dentro da rede durante um período de tempo relativamente longo, uma limitação de taxa de pico restringe o número máximo de pacotes que podem ser enviados durante um período de tempo mais curto. Usando o mesmo exemplo anterior, a rede pode regular um fluxo a uma taxa média de 6.000 pacotes por minuto e, ao mesmo tempo, limitar a taxa de pico do fluxo a 1.500 pacotes por segundo.
- *Tamanho da rajada.* A rede também pode limitar o número máximo de pacotes (a 'rajada' de pacotes) que podem ser enviados para dentro dela durante um intervalo de tempo extremamente curto. No limite, à medida que o comprimento do intervalo se aproxima de zero, o tamanho da rajada limita o número de pacotes que podem ser enviados instantaneamente para dentro da rede. Mesmo que seja impossível em termos físicos enviar vários pacotes para dentro da rede instantaneamente (afinal, cada enlace tem uma taxa de transmissão que não pode ser ultrapassada!), a abstração de um tamanho máximo de rajada é útil.

O mecanismo leaky bucket é uma abstração que pode ser usada para caracterizar esses limites da regulação. Como mostra a Figura 7.29, um leaky bucket é um balde que pode conter até b permissões. As permissões são adicionadas ao balde como segue. Novas permissões, que potencialmente podem ser adicionadas ao balde, estão sempre sendo geradas a uma taxa de r permissões por segundo. (Para facilitar, nesse caso admitimos que a unidade de tempo seja 1 segundo.) Se o balde estiver cheio com menos de b permissões quando é gerada uma permissão, a permissão recém-gerada será adicionada ao balde; caso contrário, ela será ignorada e o balde permanecerá cheio com as b permissões.

Vamos considerar agora como o leaky bucket pode ser usado para regular um fluxo de pacotes. Suponha que, antes de um pacote ser transmitido para dentro da rede, ele tenha de retirar uma permissão de dentro do balde de permissões. Se o balde de permissões estiver vazio, o pacote terá de esperar por uma permissão. (Uma alternativa é o pacote ser descartado, mas não vamos considerar essa opção aqui.) Vamos considerar agora como esse comportamento regula o fluxo de tráfego. Como pode haver no máximo b per-

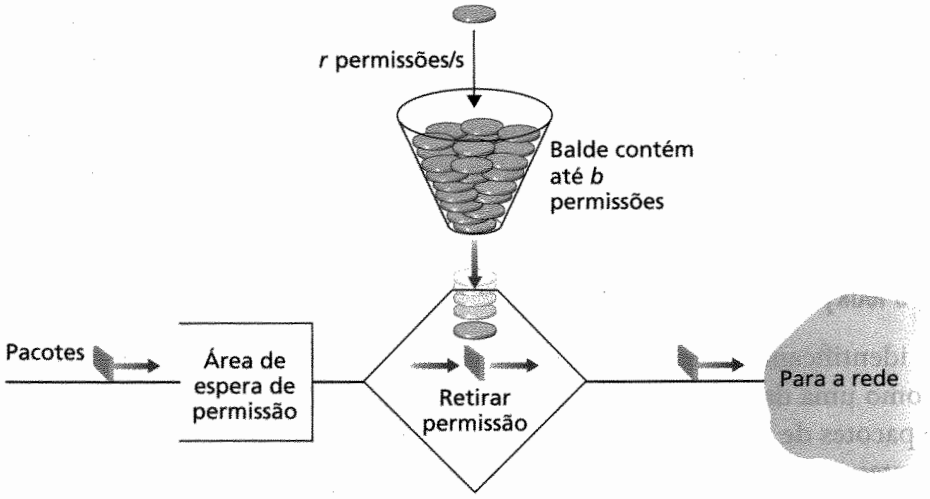


Figura 7.29 Regulação do leaky bucket

missões no balde, o tamanho máximo da rajada para um fluxo regulado pela técnica do leaky bucket é b pacotes. Além disso, como a taxa de geração de permissões é r, o número máximo de pacotes que pode entrar na rede para qualquer intervalo de tempo t é $rt + b$. Assim, a taxa de geração de permissões, r, serve para limitar a taxa média de longo período com a qual pacotes podem entrar na rede. Também é possível usar leaky buckets (especificamente dois baldes em série) para regular a taxa de pico de um fluxo e a taxa média por um período de tempo; veja os exercícios ao final deste capítulo.

Leaky bucket e enfileiramento justo ponderado fornecem máximo atraso provável em uma fila

Na Seção 7.8, examinaremos as abordagens denominadas Intserv e Diffserv para fornecimento de qualidade de serviço na Internet. Veremos que tanto a regulação do leaky bucket quanto o escalonamento WFQ podem desempenhar um papel importante. Vamos fechar esta seção considerando um enlace de saída de roteador que multiplexa n fluxos, cada um regulado por um leaky bucket com parâmetros b_i, e r_i, $i = 1, ..., n$, usando escalonamento WFQ. Usamos aqui o termo 'fluxo' de um modo um pouco impreciso para denominar conjuntos de pacotes que não são distinguidos uns dos outros pelo escalonador. Na prática, um fluxo pode conter o tráfego de uma única conexão fim-a-fim (como na abordagem Intserv) ou um conjunto de muitas conexões desse tipo (como na abordagem Diffserv); veja a Figura 7.30.

Lembre-se de que mencionamos, em nossa discussão sobre a WFQ, que cada fluxo i tem a garantia de receber uma parcela da largura de banda do enlace igual a, no mínimo, $R \cdot w_i/(\Sigma w_j)$, onde R é a taxa de transmissão do enlace em pacotes por segundo. Qual é, então, o atraso máximo que sofrerá um pacote enquanto espera ser atendido na WFQ (isto é, após ter passado pelo balde)? Vamos considerar o fluxo 1. Suponha que o balde de permissões do fluxo 1 esteja inicialmente cheio. Uma rajada de b_1 pacotes então chega ao regulador de leaky bucket para o fluxo 1. Esses pacotes retiram todas as permissões do balde (sem esperar) e, em seguida, juntam-se à área de espera WFQ para o fluxo 1. Como esses b_1 pacotes são atendidos a uma taxa de, no mínimo, $R \cdot w_1/(\Sigma w_j)$ pacotes por segundo, o último desses pacotes sofrerá um atraso máximo, $d_{máx}$, até que sua transmissão seja concluída, onde

$$d_{máx} = \frac{b_1}{R \cdot w_1/\Sigma w_j}$$

A razão dessa fórmula é que, se houver b_1 pacotes na fila e os pacotes estiverem sendo atendidos (retirados) na fila a uma taxa de, no mínimo, $R \cdot w_i/(\Sigma w_j)$ por segundo, então a quantidade de tempo até que o último bit do último pacote seja transmitido não poderá ser maior do que $b_1/(R \cdot w_i/(\Sigma w_j))$. Um exercício ao final deste capítulo solicita que você demonstre que, se $r_1 < R \cdot w_i/(\Sigma w_j)$, então $d_{máx}$ é o atraso máximo que qualquer pacote do fluxo 1 pode sofrer na fila WFQ.

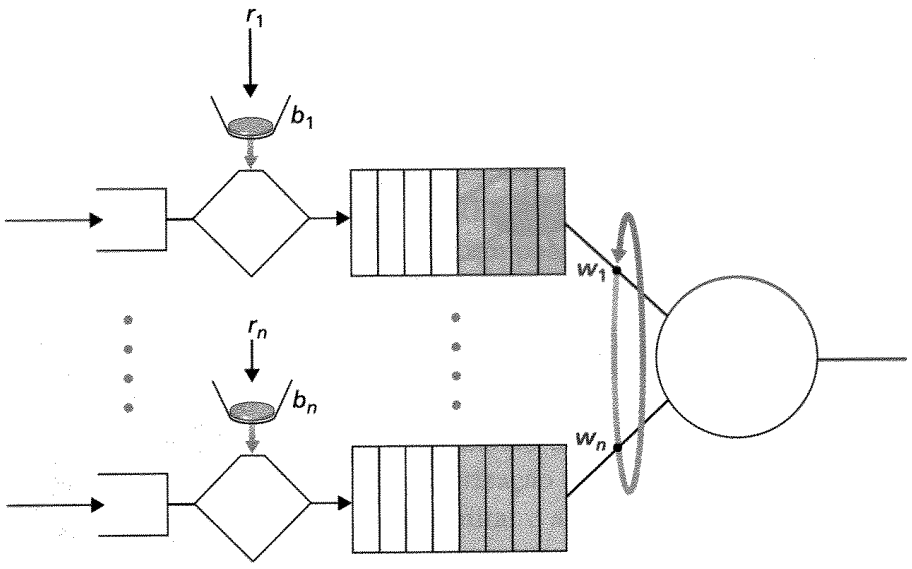


Figura 7.30 *n* fluxos multiplexados com leaky bucket com escalonamento WFQ

7.8 Serviços integrados e serviços diferenciados

Nas seções anteriores, identificamos os princípios e os mecanismos usados para prover qualidade de serviço na Internet. Nesta seção, vamos considerar como essas idéias são utilizadas em duas arquiteturas específicas propostas para fornecer qualidade de serviço à Internet — a arquitetura de serviços integrados (**Intserv**) e a arquitetura de serviços diferenciados (**Diffserv**). Como veremos, Intserv é uma estrutura desenvolvida pela IETF para fornecer garantias de qualidade de serviço específicas às sessões de aplicações individuais. O objetivo da Diffserv é prover a capacidade de manipular diferentes classes de tráfego de modos diferentes dentro da Internet. Embora nenhuma dessas arquiteturas tenha alcançado um sucesso incontestável nem tenha sido amplamente adotada (por razões que serão discutidas), elas representam os padrões correntes da IETF para prover qualidade e garantia de serviço. Como incorporam os princípios e os mecanismos que estudamos nas seções anteriores, elas são de nosso interesse.

7.8.1 Intserv

A arquitetura Intserv tem duas características fundamentais:

- *Recursos reservados.* Um roteador tem de saber que quantidade de seus recursos (buffers, largura de banda de enlace) já está reservada para sessões em andamento.
- *Estabelecimento de chamada.* Uma sessão que exige garantias de QoS deve primeiramente estar habilitada a reservar recursos suficientes em cada roteador da rede em seu trajeto fonte–destino para garantir que suas exigências de QoS fim-a-fim sejam cumpridas. Esse processo de estabelecimento de chamada (também conhecido como aceitação de chamada) exige a participação de cada roteador no trajeto. Cada roteador deve determinar os recursos locais exigidos pela sessão, considerar a quantidade de seus recursos que já estão comprometidos com outras sessões em andamento e determinar se tem recursos suficientes para satisfazer as exigências de QoS por salto da sessão naquele roteador sem violar as garantias de QoS dadas a uma sessão que já foi aceita.

A Figura 7.31 demonstra o processo de estabelecimento de chamada. Vamos agora considerar com mais detalhes as etapas envolvidas na aceitação da chamada:

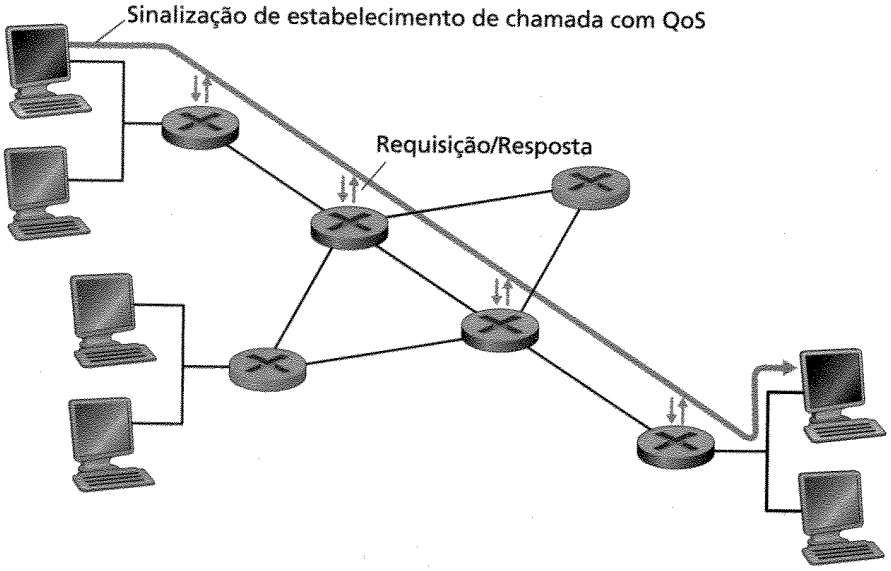


Figura 7.31 O processo de estabelecimento de chamada

1. *Caracterização do tráfego e especificação da QoS desejada.* Para que um roteador determine se seus recursos são suficientes ou não para atender às exigências de QoS de uma sessão, esta deve, primeiramente, declarar suas exigências de QoS, bem como caracterizar o tráfego que estará enviando à rede e para o qual requer uma garantia de QoS. Na arquitetura Intserv, a Rspec (R de reserva) define a QoS específica que está sendo requisitada por uma conexão; a Tspec (T de tráfego) caracteriza o tráfego que o remetente estará enviando à rede ou que o destinatário estará recebendo da rede. A forma específica da Rspec e da Tspec variará dependendo do serviço requisitado, como discutido a seguir. A Tspec e a Rspec são definidas em parte no RFC 2210 e no RFC 2215.
2. *Sinalização para o estabelecimento da chamada.* A Tspec e a Rspec de uma sessão devem ser transportadas para os roteadores nos quais serão reservados os recursos para essa sessão. Na Internet, o protocolo RSVP, que será discutido detalhadamente em breve, é atualmente o protocolo de sinalização preferido. O RFC 2210 descreve o uso do protocolo de reserva de recursos RSVP com a arquitetura Intserv.
3. *Aceitação de chamada por elemento.* Tão logo um roteador receba a Tspec e a Rspec para uma sessão que requisita garantia de QoS, ele pode determinar se pode ou não aceitar a chamada. Essa decisão de aceitação da chamada dependerá da especificação do tráfego, do tipo de serviço exigido e do comprometimento de recursos existentes já realizado pelo roteador para sessões em andamento. A aceitação de chamada por elemento é mostrada na Figura 7.32.

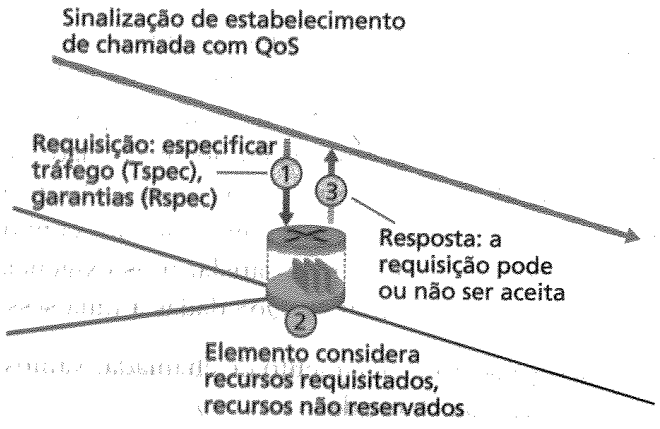


Figura 7.32 Comportamento da chamada por elemento

A arquitetura Intserv define duas grandes classes de serviços: serviço garantido e serviço de carga controlada. Veremos a seguir que cada uma dessas classes fornece um modelo muito diferente de garantia de qualidade de serviço.

Serviço de qualidade garantida

A especificação do serviço garantido, definida no RFC 2212, estabelece limites rígidos (que podem ser provados matematicamente) para os atrasos de fila que um pacote sofrerá em um roteador. Embora os detalhes subjacentes ao serviço garantido sejam bastante complicados, na realidade, a idéia básica é muito simples. Como uma primeira aproximação, a caracterização do tráfego de uma fonte é dada por um leaky bucket (veja a Seção 7.7) com parâmetros (r, b) e o serviço requisitado é caracterizado por uma taxa de transmissão R à qual os pacotes serão transmitidos. Em essência, o que a sessão requisitante do serviço garantido quer é que seja garantida uma taxa de transmissão de R bits por segundo aos bits de seu pacote. Dado que o tráfego é especificado usando-se a caracterização por meio do leaky bucket e que está sendo requisitada uma taxa R garantida, também é possível limitar o atraso máximo de fila no roteador. Lembre-se de que, com a caracterização de tráfego por leaky bucket, a quantidade de tráfego (em bits) gerada em qualquer intervalo de duração t é limitada por $rt + b$. Lembre-se também que, quando uma fonte leaky bucket é alimentada para uma fila que garante que o tráfego enfileirado será atendido a uma taxa de, no mínimo, R bits por segundo (veja a Seção 7.7), o atraso máximo de fila que qualquer pacote sofrerá será limitado por b/R, contanto que R seja maior do que r. O limite real de atraso garantido sob a definição do serviço garantido é um pouco mais complicado, devido aos efeitos do empacotamento (o limite simples b/R admite que os dados estejam sob a forma de um fluxo fluido, e não como pacotes discretos), ao fato de que o processo de chegada do tráfego está sujeito à limitação de taxa de pico do enlace de entrada (o limite simples b/R admite que uma rajada de b bits pode chegar em tempo zero) e às possíveis variações adicionais no tempo de transmissão de um pacote.

Serviço de rede de carga controlada

Uma sessão que está recebendo serviço de carga controlada receberá "uma qualidade de serviço que se aproxima muito da QoS que o mesmo fluxo receberia de um elemento de rede que não tivesse carga" [RFC 2211]. Em outras palavras, a sessão pode admitir que uma 'porcentagem muito alta' de seus pacotes passará com sucesso pelo roteador sem ser descartada e sofrerá atraso próximo de zero na fila do roteador. É interessante notar que o serviço de carga controlada não dá garantias quantitativas de desempenho — não especifica o que constitui uma porcentagem muito alta de pacotes nem qual é a qualidade de serviço que fica muito próxima à de um elemento de rede que não tem carga.

O serviço de carga controlada tem como alvo as aplicações de multimídia que têm sido desenvolvidas para a Internet de hoje. Como temos visto, essas aplicações funcionam muito bem quando a rede não está carregada, mas seu desempenho decai rapidamente à medida que a rede fica mais carregada.

7.8.2 Diffserv

A capacidade de requisitar e reservar recursos por fluxo, por sua vez, faz com que seja possível que a estrutura Intserv forneça qualidade de serviço a fluxos individuais. No entanto, à medida que progrediam os trabalhos com o Intserv, os pesquisadores envolvidos começaram a avaliar algumas dificuldades associadas a esse modelo e à reserva de recursos por fluxo:

- *Escalabilidade.* A reserva de recursos por fluxo implica a necessidade de um roteador para processar reservas de recursos e para manter estado de fluxo para cada fluxo que passa pelo roteador. Assim, o processamento de reserva por fluxo em um roteador de backbone pode representar uma sobrecarga potencialmente significativa em redes de grande porte.

- *Modelos de serviço flexíveis.* A estrutura Intserv atende a um pequeno número de classes de serviços pré-especificadas. Esse conjunto específico de classes de serviço não comporta definições mais qualitativas ou relativas para as diferenças entre as classes (por exemplo, "O serviço de classe A receberá tratamento preferencial em relação ao serviço de classe B"). Essas definições mais qualitativas poderiam se ajustar melhor à nossa idéia intuitiva do que seja diferença de serviço (por exemplo, primeira classe versus classe econômica em viagens de avião; 'cartão platina' versus 'cartão ouro' versus 'cartão normal' quando se trata de cartões de crédito).

Essas considerações levaram à atividade Diffserv na Internet Engineering Task Force. A arquitetura Diffserv visa a prover diferenciação de serviço escalável e flexível — isto é, a capacidade de manipular diferentes 'classes' de tráfego de maneiras diferentes dentro da Internet. A necessidade de escalabilidade surge do fato de que centenas de milhares de fluxos de tráfego fonte–destino simultâneos podem estar presentes em um roteador de backbone da Internet. Veremos em breve que essa necessidade é satisfeita pela introdução de uma funcionalidade simples no núcleo da rede, com a implementação de operações de controle mais complexas na borda da rede. A necessidade de flexibilidade surge do fato de que novas classes de serviço podem surgir e velhas classes de serviço podem se tornar obsoletas. A arquitetura Diffserv é flexível, no sentido de que não define serviços específicos nem classes de serviço específicas (como é o caso, por exemplo, da arquitetura Intserv). Em vez disso, ela fornece os componentes funcionais, isto é, as peças de uma arquitetura de rede com as quais esses serviços podem ser montados. Vamos examinar esses componentes detalhadamente.

Serviços diferenciados: um cenário simples

Com a finalidade de estabelecer uma estrutura para definir os componentes arquitetônicos do modelo de serviço diferenciado, vamos começar com a rede simples mostrada na Figura 7.33. Nesta seção, descreveremos um possível uso dos componentes da arquitetura Diffserv. Muitas outras variações são possíveis, como descreve o RFC 2475. Nosso objetivo aqui é apresentar uma introdução aos aspectos fundamentais do Diffserv, e não descrever o modelo arquitetônico com detalhes exaustivos. O leitor interessado em aprender mais sobre Diffserv pode consultar o livro muito abrangente de [Kilkki, 1999].

A arquitetura Diffserv consiste em dois conjuntos de elementos funcionais:

- *Funções de borda: classificação de pacotes e condicionamento de tráfego.* Na borda de entrada da rede (isto é, ou em um hospedeiro habilitado a Diffserv que gera o tráfego, ou no primeiro roteador habilitado a Diffserv pelo qual o tráfego passa), os pacotes que chegam são marcados. Mais especificamente, o campo Differentiated Service (DS) do cabeçalho do pacote é configurado para algum valor. Por exemplo, na Figura 7.33, os pacotes que estão sendo enviados de H1 para H3 poderiam ser marcados em R1, ao passo que os pacotes enviados de H2 para H4 poderiam ser marcados em R2. A marca que um pacote recebe identifica a classe de tráfego à qual ele pertence. Assim, diferentes classes de tráfego receberão serviços diferenciados dentro do núcleo da rede.
- *Função central: envio.* Quando um pacote marcado com DS chega a um roteador habilitado para Diffserv, ele é repassado até seu próximo salto de acordo com o **comportamento por salto** associado à classe do pacote. O comportamento por salto influencia a maneira pela qual os buffers e a largura de banda de um roteador são compartilhados entre as classes de tráfego concorrentes. Um dogma crucial da arquitetura Diffserv é que o comportamento por salto do roteador se baseará somente nas marcas dos pacotes, isto é, na classe de tráfego a que o pacote pertence. Assim, se os pacotes que estão sendo enviados de H1 para H3 na Figura 7.33 receberem a mesma marca dos que estão sendo enviados de H2 para H4, os roteadores da rede tratarão esses pacotes como um agregado, sem distinguir se eles se originam de H1 ou H2. Por exemplo, R3 não faria nenhuma distinção entre pacotes de H1 e H2 ao transmiti-los a R4. Portanto, a arquitetura de serviço diferenciado evita a necessidade de manter o estado do roteador para pares fonte–destino individuais — uma consideração importante para atender aos requisitos de escalabilidade discutidos no início desta seção.

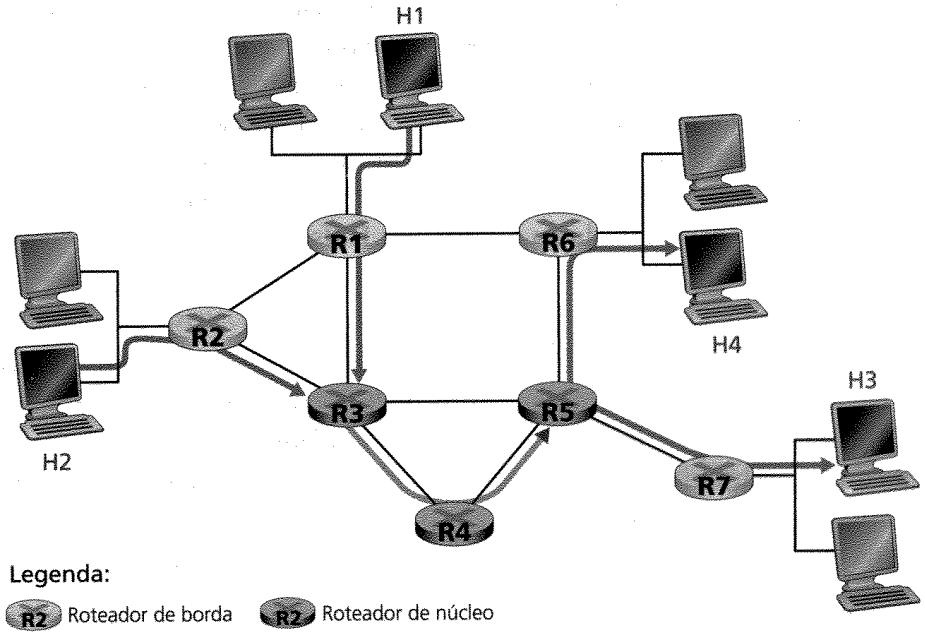


Figura 7.33 Um exemplo simples de rede Diffserv

Uma analogia poderia ser útil nesse ponto. Em muitos eventos sociais de grande escala (por exemplo, uma recepção com muitos convidados, uma boate ou discoteca, um concerto ou uma partida de futebol), as pessoas que participarão do evento adquirem algum tipo de ingresso. Há ingressos VIP para pessoas muito importantes; há ingressos para maiores de 18 anos (por exemplo, para eventos em que serão servidas bebidas alcoólicas), há credenciais que dão direito a visitar o camarim dos artistas; há ingressos especiais para repórteres e imprensa em geral; há até mesmo um ingresso comum para o cidadão comum. Esses ingressos são, em geral, adquiridos em bilheterias, isto é, na borda do evento. E é aqui, na borda, que são realizadas as operações que requerem intensa atividade computacional, como pagar pelo ingresso, verificar o tipo adequado de ingresso, comparar o ingresso com um documento de identidade. Além disso, pode haver um limite para o número de pessoas de um dado tipo que poderão entrar no evento. Se houver esse limite, as pessoas poderão ter de esperar antes de entrar. Uma vez lá dentro, o tipo de ingresso determina o tipo de serviço diferenciado recebido nos diversos locais do evento — um VIP recebe bebidas gratuitas, uma mesa melhor, refeição gratuita, dispõe de salas exclusivas e recebe serviço especial, ao passo que uma pessoa comum não pode entrar em certas áreas, paga sua bebida e recebe apenas o serviço básico. Em ambos os casos, o serviço recebido no evento depende exclusivamente do tipo de ingresso da pessoa. Além disso, todas as pessoas pertencentes à mesma classe recebem tratamento igual.

Classificação e condicionamento de tráfego do Diffserv

A Figura 7.34 apresenta uma visão lógica da função de classificação e marcação no roteador de borda. Os pacotes que chegam ao roteador de borda são primeiramente classificados. O classificador seleciona os pacotes com base nos valores de um ou mais campos de cabeçalho de pacote (por exemplo, endereço da fonte, endereço de destino, porta da fonte, porta de destino e protocolo ID) e dirige o pacote à função de marcação apropriada. A marca de um pacote é carregada dentro do campo DS [RFC 3260] no cabeçalho de pacote IPv4 ou IPv6. A intenção é que a definição do campo DS substitua as definições anteriores do campo de tipo de serviço do IPv4 e dos campos de classe de tráfego do IPv6 que discutimos no Capítulo 4.

Em alguns casos, um usuário final pode ter concordado em limitar sua taxa de envio de pacotes conforme um **perfil de tráfego** declarado. O perfil de tráfego poderia conter um limite à taxa de pico, bem como às rajadas do fluxo de pacotes, como vimos na Seção 7.7 quando tratamos do mecanismo leaky bucket. Enquanto o usuário enviar pacotes para dentro da rede de um modo que esteja de acordo com o perfil

de tráfego negociado, os pacotes receberão sua marca de prioridade e serão transmitidos ao longo de sua rota até o destino. Por outro lado, se o perfil de tráfego for violado, pacotes que estão fora do perfil poderão ser marcados de modo diferente, ajustados (por exemplo, atrasados de modo que seja observada a limitação da taxa máxima) ou descartados na borda da rede. O papel da **função de medição** mostrada na Figura 7.34 é comparar o fluxo de entrada de pacotes com o perfil de tráfego negociado. A decisão de imediatamente remarcar, transmitir, atrasar ou descartar um pacote é uma questão de política determinada pelo administrador da rede e não está especificada na arquitetura Diffserv.

Comportamentos por salto

Até aqui, examinamos as funções de borda da arquitetura Diffserv. O segundo componente fundamental dessa arquitetura envolve o comportamento por salto (PHB — *per-hop behavior*) realizado pelos roteadores habilitados para Diffserv. O PHB é definido de maneira cuidadosa, se bem que um tanto enigmática, como "uma descrição do comportamento de envio de um nó Diffserv que possa ser observado externamente aplicado a um comportamento agregado Diffserv em particular" [RFC 2475]. Explorando essa definição mais a fundo, podemos ver diversas considerações importantes nela embutidas:

- Um PHB pode resultar no recebimento de diferentes desempenhos por diferentes classes de tráfego (isto é, comportamentos de envio diferentes que possam ser observados externamente).
- Conquanto um PHB defina diferenças de desempenho (comportamento) entre classes, ele não determina nenhum mecanismo específico para conseguir esses comportamentos. Desde que os critérios de desempenho observados externamente sejam cumpridos, quaisquer mecanismos de implementação e quaisquer políticas de alocação de buffer/largura de banda podem ser usados. Por exemplo, um PHB não exigiria que fosse utilizada uma disciplina de enfileiramento de pacotes em particular (por exemplo, um enfileiramento prioritário *versus* um enfileiramento justo ponderado *versus* uma fila primeiro a chegar/primeiro a ser atendido) para atingir um determinado comportamento. O PHB é o fim para o qual os mecanismos de alocação e implementação de recursos são os meios.
- As diferenças de comportamento devem ser observáveis e, por conseguinte, mensuráveis.

Até agora foram definidos dois PHBs: um PHB de repasse acelerado (*expedited forwarding* — EF) [RFC 3246] e um PHB de repasse assegurado (*assured forwarding* — AF) [RFC 2597]:

- O PHB de **repasse acelerado** especifica que a taxa de partida de uma classe de tráfego de um roteador deve ser igual ou maior do que uma taxa configurada. Isto é, durante qualquer intervalo de tempo, fica garantido que a classe de tráfego receba largura de banda suficiente de modo que a taxa de saída do tráfego seja igual ou maior do que essa taxa mínima configurada. Note que o

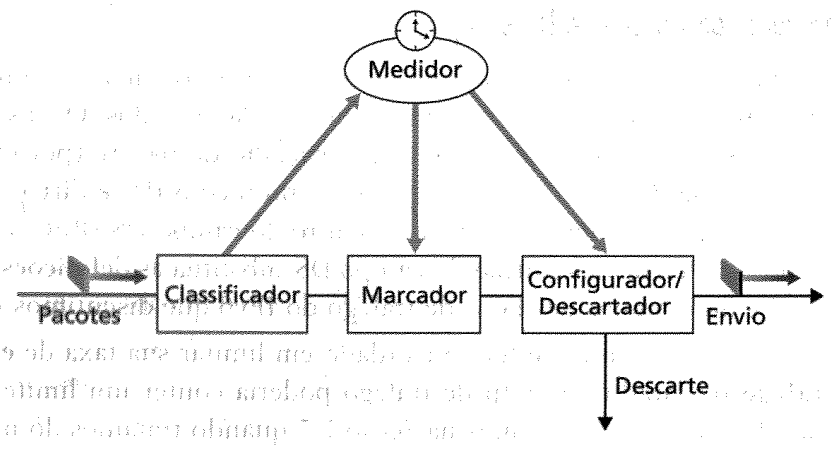


Figura 7.34 Visão lógica da classificação de pacotes e do condicionamento de tráfego no roteador de borda

comportamento por salto EF implica algum tipo de isolamento entre as classes de tráfego, pois essa garantia vigora independentemente da intensidade do tráfego de quaisquer outras classes que estejam chegando a um roteador. Assim, mesmo que as outras classes de tráfego estejam sobrecarregando os recursos do roteador e do enlace, recursos suficientes deverão ser mantidos disponíveis para a classe, para assegurar que ela receba sua taxa mínima garantida. Portanto, o EF fornece a uma classe uma abstração simples de um enlace com largura de banda de enlace mínima garantida.

O PHB de **envio assegurado** é mais complexo. O AF divide o tráfego em quatro classes e garante, a cada classe AF, o fornecimento de alguma quantidade mínima de largura de banda e de buffer. Dentro de cada classe, os pacotes ainda são repartidos em uma de três categorias de descarte preferencial. Quando ocorre um congestionamento dentro de uma classe AF, um roteador pode então descartar pacotes com base em seus valores de descarte preferencial. Para mais detalhes, consulte o RFC 2597. Variando a quantidade de recursos alocados a cada classe, um ISP pode fornecer diferentes níveis de desempenho às diversas classes de tráfego AF.

Retrospectiva do Intserv e do Diffserv

Durante os últimos 20 anos houve numerosas tentativas malsucedidas de introduzir QoS em redes de comutação de pacotes. As várias tentativas falharam, até agora, mais por razões econômicas e de compatibilidade do que por razões técnicas. Entre elas estão redes ATM fim-a-fim e redes TCP/IP que oferecem Intserv ou Diffserv. Esses esforços alcançarão sucesso, ou ao menos sucesso parcial, no longo prazo? Vamos examinar algumas das questões envolvidas.

Até aqui, admitimos que Intserv e Diffserv eram disponibilizados dentro de um único domínio administrativo. O caso mais típico é o do serviço fim-a-fim, que tem de ser configurado para várias ISPs localizadas entre os sistemas finais comunicantes. Para prover serviço Intserv ou Diffserv fim-a-fim, todas as ISPs entre o sistema final devem não apenas prover esse serviço, mas, acima de tudo, cooperar e fazer acordos para oferecer aos clientes finais um serviço verdadeiramente fim-a-fim. Sem esse tipo de cooperação, ISPs que vendem serviços Intserv ou Diffserv diretamente a clientes terão sempre de se justificar perante eles dizendo: "Sim, sabemos que você pagou um extra, mas não temos um acordo de serviço com suas ISPs de nível mais alto. Pedimos desculpas pelas muitas lacunas na sua chamada de voz sobre IP!"

Uma outra preocupação com esses serviços avançados é que eles precisam monitorar e possivelmente condicionar o tráfego, o que pode se revelar uma operação complexa e cara. Também é preciso cobrar pelo serviço de modo diferenciado, mais provavelmente por volume, e não por uma taxa mensal fixa, como fazem atualmente a maioria dos ISPs — um outro requisito caro para o ISP. Finalmente, se o Intserv ou o Diffserv estivessem realmente disponíveis e a rede executasse somente sob carga moderada, na maior parte do tempo não se perceberia nenhuma diferença entre um serviço de melhor esforço e um serviço Intserv/Diffserv. Na verdade, hoje, atraso fim-a-fim é usualmente dominado pelas taxas de acesso e saltos de roteadores e não por atrasos de fila em roteadores. Imagine o infeliz cliente do Intserv/Diffserv que pagou a mais por um serviço diferenciado, mas descobre que o serviço de melhor esforço oferecido por outros quase sempre apresenta o mesmo desempenho de um serviço diferenciado.

7.9 RSVP

Vimos na Seção 7.8 que, para que uma rede forneça garantias de QoS, é preciso que haja um protocolo de sinalização que permita que as aplicações que rodam em hospedeiros reservem recursos na Internet. O recurso ReSerVation Protocol (Protocolo de Reserva de Largura de Banda — RSVP) [RFC 2205; Zhang, 1993] é esse protocolo de sinalização para a Internet.

Quando as pessoas falam sobre recursos no contexto da Internet, elas comumente estão se referindo à largura de banda de enlace e a buffers de roteadores. Contudo, para manter a discussão centrada e em ter-

reno concreto, vamos admitir que a palavra recurso seja sinônimo de largura de banda. Observamos que o RSVP tem sido estendido e utilizado como um protocolo de sinalização em outras circunstâncias, das quais a mais notável talvez seja sob a forma do RSVP-TE [RFC 3209] para sinalização MPLS, como discutido na Seção 5.8.2. Porém, aqui, focalizaremos a utilização do RSVP para reservar recursos (largura de banda).

7.9.1 A essência do RSVP

O protocolo RSVP permite que aplicações reservem largura de banda para seus fluxos de dados. Ele é usado por um hospedeiro, em nome do fluxo de dados de uma aplicação, para requisitar uma quantidade específica de largura de banda da rede. O RSVP também é usado pelos roteadores para repassar requisições de reserva de largura de banda. Para implementar o RSVP, um software RSVP deve estar presente nos receptores, remetentes e roteadores. As duas características principais do RSVP são:

1. Ele fornece **reservas de largura de banda em árvores *multicast*** (o *unicast* é tratado como um caso de *multicast* degenerado).
2. Ele é **orientado para o receptor**, isto é, o receptor do fluxo de dados é quem inicia e mantém a reserva de recursos usada para aquele fluxo.

Essas duas características são ilustradas na Figura 7.35. O diagrama mostra uma árvore *multicast* com dados fluindo do topo da árvore aos hospedeiros na base da árvore. Embora os dados se originem do remetente, as mensagens de reserva se originam dos receptores. Quando um roteador transmite uma mensagem de reserva na direção do remetente, ele pode consolidar a mensagem de reserva com outras mensagens de reserva que estão chegando da corrente inferior.

Antes de discutir o RSVP mais detalhadamente, precisamos considerar a idéia de **sessão**. Como acontece com o RTP, uma sessão pode consistir em vários fluxos de dados *multicast*. Cada remetente em uma sessão é a fonte de um ou mais fluxos de dados; por exemplo, um remetente pode ser a fonte de um fluxo de dados de vídeo e de um fluxo de dados de áudio. Cada fluxo de dados de uma sessão tem o mesmo endereço *multicast*. Para manter a discussão em terreno concreto, admitimos que roteadores e hospedeiros identifiquem a sessão à qual um pacote pertence pelo endereço *multicast* do pacote. Essa pressuposição é, de certa forma, restritiva; a especificação do RSVP, na realidade, permite métodos mais gerais para identificar uma sessão. Dentro de uma sessão, o fluxo de dados ao qual um pacote pertence também precisa ser identificado, o que poderia ser feito, por exemplo, pelo campo de identificação de fluxo no IPv6.

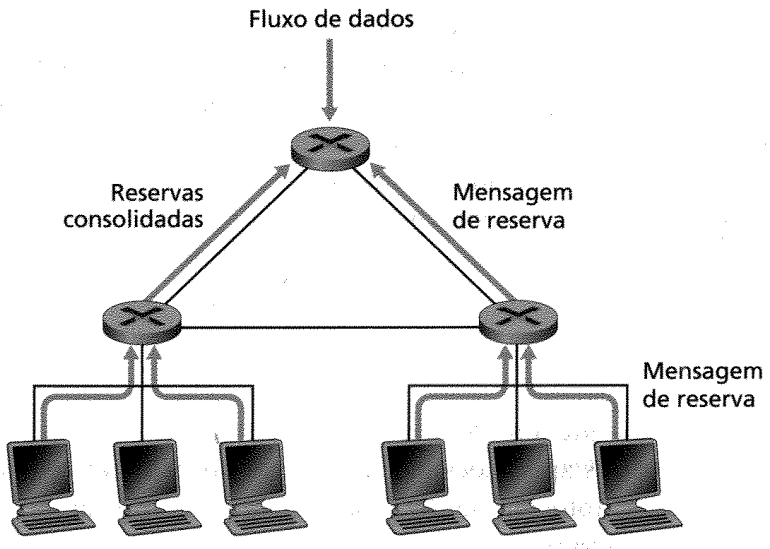


Figura 7.35 RSVP: orientado para multicast e para o receptor

O que o RSVP não faz

Enfatizamos que o padrão RSVP [RFC 2205] não especifica *como* a rede fornece largura de banda reservada aos fluxos de dados. Ele é meramente um protocolo que permite que as aplicações reservem a necessária largura de banda de enlace. Uma vez colocadas as reservas, o fornecimento da largura de banda reservada aos fluxos de dados passa a ser tarefa dos roteadores da Internet. Provavelmente, esse fornecimento seria feito por meio dos mecanismos de escalonamento (escalonamento prioritário, enfileiramento justo ponderado etc.), discutidos na Seção 7.7.

É também importante entender que o RSVP não é um protocolo de roteamento — ele não determina os enlaces nos quais as reservas devem ser feitas. Em vez disso, ele depende de um protocolo de roteamento subjacente (*unicast* ou *multicast*) para determinar as rotas dos fluxos. Tão logo as rotas estejam determinadas, o RSVP pode reservar largura de banda nos enlaces ao longo dessas rotas. (Veremos em breve que, quando uma rota muda, o RSVP atualiza a reserva de recursos.) Assim que as reservas estejam em vigor, os escalonadores de pacotes do roteador devem realmente fornecer aos fluxos de dados a largura de banda reservada. Assim, o RSVP é apenas uma peça — embora uma peça importante — do quebra-cabeça da garantia de QoS.

O RSVP é às vezes denominado protocolo de sinalização. Isso quer dizer que ele é um protocolo que permite que os hospedeiros estabeleçam e encerrem reservas para fluxos de dados. A expressão *protocolo de sinalização* vem do jargão da comunidade de telefonia por comutação de circuitos.

Receptores heterogêneos

Considere uma rede na qual alguns receptores podem receber um fluxo à taxa de 28,8 kbps; outros, à taxa de 128 kbps, e ainda outros, à taxa de 10 Mbps ou até mais alta. Essa heterogeneidade dos receptores propõe uma questão interessante. Se um remetente estiver enviando vídeo em *multicast* para um grupo de receptores heterogêneos, ele deverá codificar o vídeo para má qualidade a 28,8 kbps, para qualidade média a 128 kbps ou para alta qualidade a 10 Mbps? Se o vídeo for codificado a 10 Mbps, então somente os usuários com acesso de 10 Mbps poderão assistir a ele. Por outro lado, se o vídeo for codificado a 28,8 kbps, então os usuários de 10 Mbps serão obrigados a ver imagens de baixa qualidade sabendo que poderiam estar vendo algo muito melhor.

Para resolver esse dilema, a sugestão mais freqüente é que áudio e vídeo sejam codificados em duas camadas. Por exemplo, um vídeo poderia ser codificado em duas camadas: uma camada de base e uma camada de realce. A camada de base poderia ter uma taxa de 20 kbps e a camada de realce, uma taxa de 100 kbps. Desse modo, receptores com acessos de 28,8 kbps poderiam receber a imagem de baixa qualidade da camada de base, e receptores com acessos de 128 kbps poderiam receber ambas as camadas e fundi-las, resultando em uma imagem de alta qualidade.

Observamos que o remetente não precisa conhecer as taxas de recepção de todos receptores. Ele precisa apenas saber qual é a taxa máxima de todos os seus receptores. O remetente codifica o áudio e o vídeo em várias camadas e envia todas elas para a árvore *multicast* à taxa máxima. Os receptores escolhem as camadas que são adequadas às suas taxas de recepção. Para não desperdiçar excessivamente a largura de banda nos enlaces da rede, os receptores heterogêneos devem comunicar à rede as taxas que podem manusear. Veremos que o RSVP dá atenção prioritária à questão de reservas de recursos para receptores heterogêneos.

7.9.2 Alguns exemplos simples

Primeiramente vamos descrever o RSVP no contexto de um exemplo concreto de *multicast* um-para-muitos. Suponha que haja uma fonte transmitindo vídeo de um importante evento esportivo pela Internet. Essa sessão recebeu um endereço *multicast* e a fonte marca todos os pacotes que saem com esse endereço *multicast*. Suponha também que um protocolo de roteamento *multicast* subjacente tenha estabelecido uma árvore *multicast* do remetente até quatro receptores, como mostra a Figura 7.36; os números ao lado dos

receptores são as taxas com as quais eles querem receber dados. Vamos admitir ainda que o vídeo esteja em camadas e codificado para atender a essa heterogeneidade de taxas dos receptores.

Em termos simples, o RSVP opera em duas fases, como ilustrado no exemplo simples a seguir. Uma fonte transmissora anuncia seu conteúdo enviando **mensagens de caminho RSVP** por meio de uma árvore *multicast*, indicando a largura de banda requerida pelo conteúdo, a duração da temporização do estado soft (veja o quadro O princípio do estado soft) e as informações sobre o caminho ascendente até o remetente. Cada receptor envia uma **mensagem de reserva RSVP** à árvore *multicast* pela corrente ascendente. Essa mensagem de reserva especifica a taxa à qual o receptor gostaria de receber os dados que vêm da fonte. Quando a mensagem de reserva chega a um roteador, este ajusta seu escalonador de pacotes para atender àquela reserva. Em seguida, envia uma reserva pela corrente ascendente. A quantidade de largura de banda reservada na corrente ascendente a partir do roteador depende das larguras de banda reservadas na corrente descendente. No exemplo da Figura 7.36, os receptores R1, R2, R3 e R4 reservam 20 kbps, 100 kbps, 3 Mbps e 3 Mbps, respectivamente. Assim, os receptores da corrente descendente que parte do roteador D requisitam, no máximo, 3 Mbps. Para essa transmissão um-para-muitos, o roteador D envia uma mensagem de reserva ao roteador B requisitando que este reserve 3 Mbps no enlace entre os dois roteadores. Note que somente 3 Mbps são reservados, e não 3 + 3 = 6 Mbps; isso ocorre porque os receptores R3 e R4 estão assistindo ao mesmo evento esportivo; portanto, suas reservas podem ser consolidadas. De maneira semelhante, o roteador C solicita que o roteador B reserve 100 kbps no enlace entre os roteadores B e C; a codificação em camadas assegura que a corrente de 20 kbps de R1 seja incluída na corrente de 100 kbps. Assim que o roteador B recebe a mensagem de reserva dos roteadores abaixo dele e passa a reserva a seus escalonadores, ele envia uma nova mensagem de reserva ao roteador da corrente ascendente, roteador A. Essa mensagem reserva 3 Mbps de largura de banda no enlace entre o roteador A e o roteador B, que corresponde, novamente, à reserva máxima na corrente descendente.

Por esse primeiro exemplo, vemos que o RSVP é **orientado para receptor**, isto é, o receptor de um fluxo de dados inicia e mantém a reserva de recursos usada para aquele fluxo. Note que cada roteador recebe uma mensagem de reserva de cada um de seus enlaces na corrente descendente da árvore *multicast* e envia apenas uma mensagem de reserva ao enlace na corrente ascendente.

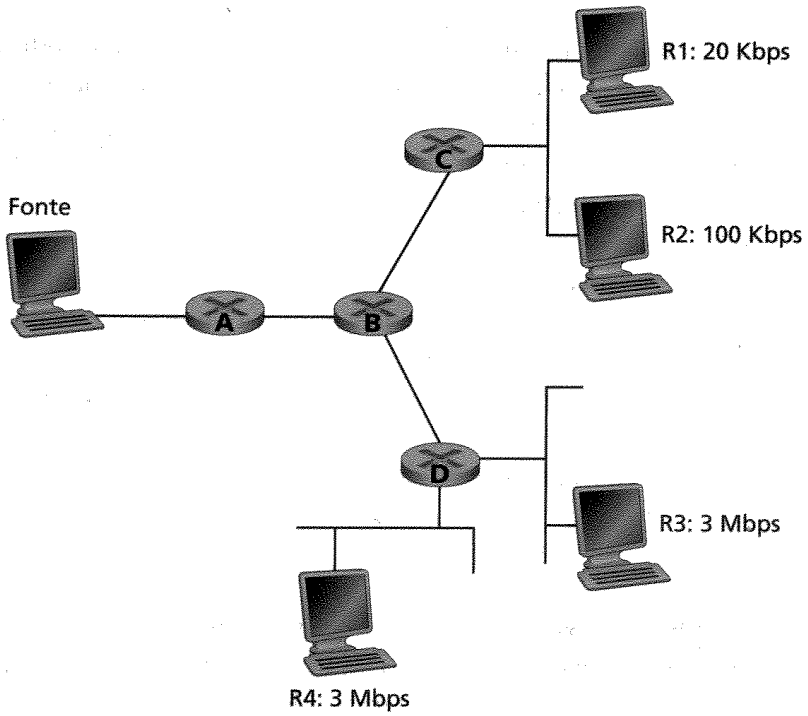


Figura 7.36 Um exemplo de RSVP

Como outro exemplo, suponha que quatro pessoas estejam participando de uma videoconferência, como mostra a Figura 7.37. Cada pessoa tem três janelas abertas em seu computador para poder ver as outras três pessoas. Suponha também que o protocolo de roteamento subjacente tenha estabelecido a árvore *multicast* entre os quatro hospedeiros. Finalmente, suponha que cada pessoa queira ver cada um dos vídeos a 3 Mbps. Então, em cada um dos enlaces dessa árvore *multicast*, o RSVP reservaria 9 Mbps em uma direção e 3 Mbps na outra direção. Note que o RSVP não consolida reservas nesse exemplo, pois cada pessoa quer receber três correntes distintas.

Considere, agora, uma audioconferência entre as mesmas quatro pessoas pela mesma árvore *multicast*. Suponha que sejam necessários b bps para uma corrente isolada de áudio. Como em uma audioconferência é raro que mais de duas pessoas falem ao mesmo tempo, não é necessário reservar $3 \cdot b$ bps em cada receptor; $2 \cdot b$ deverão ser suficientes. Assim, essa última aplicação pode economizar largura de banda consolidando reservas.

Aceitação de chamadas

Da mesma maneira que o gerente de restaurante da Seção 1.3.1 não deve aceitar reservas para um número maior de mesas do que o restaurante dispõe, a quantidade de largura de banda que um roteador reserva sobre um enlace não deve ultrapassar a capacidade do enlace. Assim, sempre que um roteador receber uma nova mensagem de reserva, ele deve, primeiramente, determinar se os enlaces descendentes da árvore *multicast* podem aceitar a reserva. Esse **teste de aceitação** é realizado sempre que um roteador recebe uma mensagem de reserva. Se o teste de aceitação falhar, o roteador rejeitará a reserva e devolverá uma mensagem de erro aos receptores apropriados.

O RSVP não define o teste de aceitação, mas admite que os roteadores realizem o teste e que o RSVP possa interagir com esse teste. Se o leitor quiser saber mais sobre aceitação de chamada, pode consultar [Jamin 1996; Breslau 2000; Roberts 2004].

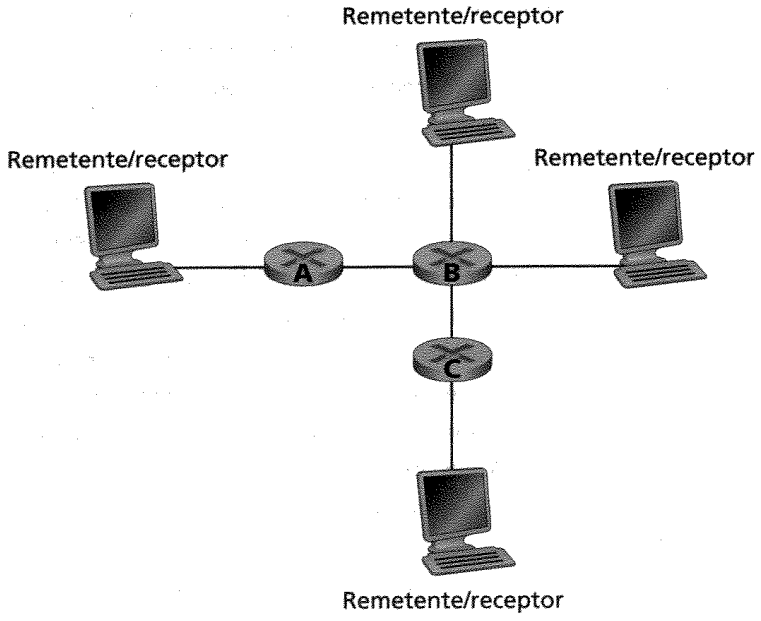


Figura 7.37 Um exemplo de videoconferência RSVP

Princípios na prática

O princípio do estado soft

O RSVP é usado para criar um estado (reservas de largura de banda) em roteadores, e é conhecido como um protocolo de *estado soft*. Em termos gerais, associamos o termo estado soft com abordagens de sinalização nas quais o estado criado se extingue (e é removido) por temporização, a menos que seja renovado periodicamente pelo recebimento de uma mensagem de sinalização (caracteristicamente da entidade que criou o estado no início) indicando que o estado deve continuar e permanecer instalado. Como um estado não renovado será extinto a qualquer momento, a sinalização de *estado soft* não requer remoção explícita de estado nem um procedimento para remover estados órfãos caso o criador de estados falhe. De modo semelhante, visto que criação de estado e mensagens de renovação serão seguidas por mensagens de renovação periódicas subseqüentes, não é necessário sinalização confiável. O termo *estado soft* foi cunhado por Clark [Clark, 1998], que descreveu a noção de mensagens periódicas de renovação de estado enviadas por um sistema final e sugeriu que, com essas mensagens de renovação, o estado podia ser perdido devido a uma falha e então restaurado automaticamente por mensagens de renovação subseqüentes — tudo transparente para o sistema final, e sem invocar nenhum procedimento de recuperação de falha:

> "... a informação de estado não seria crítica para a manutenção do tipo de serviço desejado associado com o fluxo. Em vez disso, aquele tipo de serviço seria imposto pelas extremidades, que enviariam mensagens periodicamente para assegurar que o tipo de serviço adequado está sendo associado com o fluxo. Dessa maneira, a informação de estado associada com o fluxo poderia ser perdida devido a uma falha sem causar a disrupção permanente das características de serviço que estão sendo usadas. Eu denomino esse conceito 'estado soft', e ele pode perfeitamente permitir que alcancemos nossas metas primárias de resiliência e flexibilidade..."

Grosseiramente falando, a essência de uma abordagem de estado soft é a utilização pelo criador de estado do serviço de melhor esforço para criação e renovação de estado e remoção de estado por esgotamento de temporizador no mantenedor do estado. Abordagens de estado soft foram adotadas em inúmeros protocolos, entre eles RSVP, PIM (Seção 4.7), SIP (Seção 7.4) e IGMP (Seção 4.7) e em tabelas de repasse e pontes transparentes (Seção 5.6).

Sinalização de *estado hard* adota a abordagem inversa do estado soft — o estado criado permanece efetivo a menos que explicitamente removido pelo recebimento de uma mensagem de remoção de estado enviada pelo criador de estado. Visto que o estado permanece efetivo a menos que explicitamente removido, a sinalização de estado hard requer um mecanismo para remover um estado órfão que permanece após um criador de estado falhar ou sair sem ter removido o estado. De modo semelhante, uma vez que criação e remoção de estado são realizadas apenas uma vez (e sem renovação de estado ou esgotamento de temporização de estado), é importante que o criador de estado saiba quando o estado foi criado ou removido. Assim, protocolos de sinalização confiáveis (e não de melhor esforço) são comumente associados com protocolos de estado hard. Então, grosseiramente falando, a essência de uma abordagem de estado hard é a criação e remoção confiáveis e explícitas de informação de estado. Abordagens de estado hard foram adotadas em protocolos como ST-II [Partridge, 1992; RFC 1990] e Q.293 [ITU-T Q.293, 1994].

O RSVP oferece remoção de reserva explícita (se bem que opcional) desde que foi concebido.

Sinalização confiável baseada em ACK foi introduzida como uma extensão do RSVP no [RFC 2961] e também foi sugerida em [Pan, 1997]. Assim, o RSVP adotou, por opção, alguns elementos de uma abordagem de sinalização de estado hard. Para uma discussão e comparação de protocolos de estado soft *versus* estado hard, veja [Ji, 2003].

7.10 Resumo

A rede multimídia talvez seja um dos desenvolvimentos mais interessantes (com um potencial que ainda não se realizou totalmente) da Internet de hoje. Pessoas em todo o mundo estão passando menos tempo diante de seus aparelhos de rádio ou televisão e usando mais a Internet para receber transmissões de áudio e vídeo, ao vivo ou pré-gravadas. À medida que o acesso de alta velocidade for invadindo mais e mais residências, essa tendência continuará — os habituados aos sofás de todo o mundo acessarão seus programas favoritos em vídeo por meio da Internet, e não pelos canais tradicionais de distribuição *broadcast*. Além da distribuição de áudio e vídeo, a Internet está sendo usada para transmitir chamadas telefônicas. De fato, nos próximos dez anos ela poderá vir a substituir o quase obsoleto sistema de telefonia de comutação de circuitos em muitos países. A Internet não somente proverá serviço telefônico mais barato, mas também oferecerá numerosos serviços de valor agregado, como videoconferência, lista telefônica on-line, serviço de mensagens de viva-voz e integração com a Web.

Na Seção 7.1, classificamos aplicações de multimídia em três categorias: áudio e vídeo armazenados de fluxo contínuo; transmissão um-para-muitos de áudio e vídeo em tempo real e áudio e vídeo interativos em tempo real. Enfatizamos que aplicações multimídia são sensíveis a atraso e tolerantes à perda — características muito diferentes das aplicações de conteúdo estático, que são tolerantes ao atraso e intolerantes à perda. Discutimos também alguns obstáculos que o serviço de melhor esforço da Internet apresenta às aplicações de multimídia. Avaliamos diversas propostas para superar esses obstáculos, incluindo o simples aprimoramento da infra-estrutura de rede existente (adicionando mais largura de banda, mais memórias temporárias de rede e mais nós CDN e disponibilizando o *multicast*), agregar mais funcionalidades à Internet, de modo que as aplicações possam reservar recursos fim-a-fim (para que a Internet possa honrar essas reservas) e, finalmente, introduzir classes de serviços para prover diferenciação de serviços.

Da seção 7.2 à 7.4, examinamos arquiteturas e mecanismos para rede multimídia com serviço de melhor esforço. Na Seção 7.2, avaliamos diversas arquiteturas para fluxo contínuo de áudio e vídeo armazenados. Discutimos interação com o usuário — como pausa/reinício, reposicionamento e avanço rápido visual — e apresentamos uma introdução ao RTSP, um protocolo que provê interação cliente-servidor para aplicações de fluxo contínuo. Na Seção 7.3, examinamos como aplicações interativas em tempo real podem ser projetadas para rodar sobre uma rede de serviço de melhor esforço. Vimos como uma combinação de buffers clientes, números de seqüência de pacotes e marcas de tempo pode aliviar os efeitos da variação de atraso induzida pela rede. Na Seção 7.4, estudamos protocolos para multimídia interativa em tempo real, incluindo RTP, SIP e H.323. Na Seção 7.5, investigamos como redes de distribuição de conteúdo levam conteúdo de multimídia a usuários.

Da seção 7.6 à 7.9 vimos como a Internet pode evoluir para fornecer QoS garantida às suas aplicações. Na Seção 7.6, identificamos diversos princípios do fornecimento de QoS às aplicações de multimídia. Entre eles estão a marcação e a classificação de pacotes, o isolamento de fluxos de pacotes, o uso eficiente de recursos e a aceitação de chamada. Na Seção 7.8, avaliamos várias políticas de escalonamento e mecanismos de regulação que podem prover a fundação para uma arquitetura de rede com QoS. Entre as políticas de programação estão o escalonamento prioritário, o escalonamento por varredura cíclica e o enfileiramento justo ponderado. Em seguida, examinamos mecanismos de regulação leaky bucket. Na Seção 7.9, mostramos como esses princípios e mecanismos levaram à definição de novos padrões Intserv e Diffserv para o fornecimento de QoS na Internet e de um protocolo de sinalização de reservas na Internet denominado RSVP.

Agora que terminamos nosso estudo sobre rede multimídia, é hora de passar para outro tópico interessante: a segurança na rede. Progressos recentes em rede multimídia podem deslocar a distribuição de informações por áudio e vídeo para a Internet. Como veremos no próximo capítulo, os recentes avanços em segurança de rede podem muito bem ajudar a transferir a maioria das transações financeiras para a Internet.

Exercícios de fixação

Capítulo 7 Questões de revisão

Seções 7.1 a 7.2

1. O que significa interatividade no caso de fluxo contínuo de áudio/vídeo? E no caso de áudio/vídeo interativo em tempo real?
2. Foram discutidos três campos para aprimorar a Internet de modo que ela possa dar melhor suporte às aplicações de multimídia. Faça um breve resumo das visões de cada campo. Qual deles você defende?
3. As figuras 7.1, 7.2 e 7.3 apresentam três esquemas para fluxo contínuo de mídia armazenada. Quais as vantagens e as desvantagens de cada esquema?

Seções 7.3 a 7.4

4. Qual é a diferença entre atraso fim-a-fim e variação de atraso? Quais são as causas da variação de atraso?
5. Por que um pacote recebido após seu tempo de reprodução programado é considerado perdido?
6. Na Seção 7.3, descrevemos dois esquemas FEC. Faça um pequeno resumo deles. Ambos os esquemas aumentam a taxa de transmissão da corrente adicionando sobrecarga. A intercalação também aumenta a taxa de transmissão?
7. Como as diferentes correntes RTP em sessões diferentes são identificadas por um receptor? Como as diferentes correntes internas à mesma sessão são identificadas? Como são distinguidos os pacotes RTP e RTCP (como parte da mesma sessão)?
8. Três tipos de pacotes RTCP foram descritos na Seção 7.4. Faça um pequeno resumo da informação contida em cada um desses tipos de pacotes.

Seções 7.6 a 7.9

9. Na Seção 7.6, discutimos o enfileiramento prioritário não preemptivo. O que seria um enfileiramento prioritário preemptivo? O enfileiramento prioritário preemptivo teria sentido em redes de computadores?
10. Dê um exemplo de disciplina de escalonamento que não é conservadora de trabalho.
11. Cite algumas dificuldades associadas ao modelo Intserv e à reserva de recursos por fluxo.

Problemas

1. Navegue na Web e ache dois produtos para fluxo contínuo de áudio e/ou vídeo armazenado(s). Para cada produto, determine:
   a. se são usados metarquivos;
   b. se o áudio/vídeo é enviado por UDP ou TCP;
   c. se é usado RTP;
   d. se é usado RTSP.
2. Escreva um poema, um conto, uma redação sobre férias recentes ou qualquer outro tipo de texto cuja leitura demore de 2 a 5 minutos. Leia e grave seu texto. Converta sua gravação para um dos formatos de áudio da RealNetworks ou da Microsoft usando um dos codificadores gratuitos. Carregue o arquivo no mesmo servidor que contém sua home page pessoal. Carregue também o metarquivo correspondente no servidor. Finalmente, crie um ponteiro entre sua home page e o metarquivo.
3. Considere o buffer cliente mostrado na Figura 7.4. Suponha que o sistema de transmissão use a terceira opção, isto é, o servidor lança a mídia na porta o mais rapidamente possível. Suponha que a largura

de banda TCP disponível seja >> d na maior parte do tempo. Suponha também que o buffer cliente pode conter somente cerca de um terço da mídia. Descreva como $x(t)$ e o conteúdo do buffer cliente evoluirão com o tempo.

4. O buffer de recebimento TCP e o buffer cliente do transdutor são a mesma coisa? Se não são, como eles interagem?

5. No exemplo de telefone por Internet da Seção 7.3, seja h o número total de bytes de cabeçalho adicionado a cada porção, incluindo os cabeçalhos UDP e IP.

   a. Admitindo que um datagrama IP seja emitido a cada 20 milissegundos, qual é a taxa de transmissão em bits por segundo para os datagramas gerados por um lado dessa aplicação?

   b. Qual é um valor típico de h quando é usado RTP?

6. Considere o procedimento descrito na Seção 7.3 para estimar o atraso médio d_i. Suponha que $u = 0{,}1$. Seja $r_1 - t_1$ o atraso da amostra mais recente, seja $r_2 - t_2$ o atraso da amostra mais recente seguinte e assim por diante.

   a. Para uma dada aplicação de áudio, suponha que quatro pacotes tenham chegado ao receptor com atrasos de amostra de $r_4 - t_4$, $r_3 - t_3$, $r_2 - t_2$, $r_1 - t_1$. Expresse a estimativa de atraso d em termos das quatro amostras.

   b. Generalize sua fórmula para n atrasos de amostra.

   Para a fórmula da parte (b), suponha que n tenda ao infinito e dê a fórmula resultante. Comente por que esse procedimento de determinação de média é denominado média móvel exponencial.

7. Repita as partes (a) e (b) da questão anterior para a estimativa do desvio médio do atraso.

8. No exemplo do telefone por Internet na Seção 7.3, apresentamos um procedimento on-line (média móvel exponencial) para estimar atraso. Neste problema examinaremos um procedimento alternativo. Seja t_i a marca de tempo do i-ésimo pacote recebido; seja r_i o tempo em que o i-ésimo pacote é recebido. Seja d_n nossa estimativa do atraso médio após o recebimento do *enésimo* pacote. Após a recepção do primeiro pacote, estabelecemos a estimativa de atraso como $d_1 = r_1 - t_1$.

   a. Suponha que gostaríamos que $d_n = (r_1 - t_1 + r_2 - t_2 + \ldots + r_n - t_n)/n$ para todo n. Deduza uma fórmula recursiva para d_n em termos de d_{n-1}, r_n, e t_n.

   b. Descreva por que, para a telefonia por Internet, a estimativa de atraso descrita na Seção 7.3 é mais apropriada do que a estimativa esboçada na Parte a.

9. Compare o procedimento descrito na Seção 7.3 para a estimativa do desvio médio do atraso com o procedimento descrito na Seção 3.5 para a estimativa do tempo de ida e volta. O que os procedimentos têm em comum? Em que são diferentes?

10. Considere a estratégia de reprodução adaptativa descrita na Seção 7.3.

    a. Como dois pacotes sucessivos recebidos no destino podem ter marcas de tempo que diferem em mais de 20 milissegundos se os dois pacotes pertencem à mesma rajada de voz?

    b. Como o receptor pode usar números de seqüência para determinar se um pacote é o primeiro de uma rajada de voz? Seja objetivo em sua resposta.

11. Recorde os dois esquemas FEC para telefone por Internet descritos na Seção 7.3. Suponha que o primeiro esquema gere uma porção redundante para cada quatro porções originais. Suponha que o segundo esquema use uma codificação de baixa taxa de bits, cuja taxa de transmissão seja 25 por cento da taxa de transmissão da corrente nominal.

    a. Quanta largura de banda adicional cada esquema requer? E quanto atraso de reprodução cada esquema adiciona?

    b. Como os dois esquemas funcionarão se em cada grupo de cinco pacotes o primeiro for perdido? Qual esquema terá melhor qualidade de áudio?

c. Como os dois esquemas funcionarão se em cada grupo de dois pacotes o primeiro for perdido? Qual esquema terá melhor qualidade de áudio?

12. Como é calculada a variação do atraso entre chegadas no relatório de recepção RTCP? (*Dica*: leia o RFC do RTP.)

13. a. Suponha que enviamos dois datagramas IP para a Internet, cada um portando um segmento UDP diferente. O primeiro datagrama tem endereço IP de fonte A1, endereço IP de destino B, porta de fonte P1 e porta de destino T. O segundo datagrama tem endereço IP de fonte A2, endereço IP de destino B, porta de fonte P2 e porta de destino T. Suponha que A1 é diferente de A2 e P1 é diferente de P2. Admitindo que ambos os datagramas cheguem a seu destino final, os dois datagramas UDP serão recebidos pelo mesmo socket? Justifique sua resposta.

b. Suponha que Alice, Bob e Claire querem fazer uma audioconferência usando SIP e RTP. Para Alice enviar e receber pacotes RTP de e para Bob e Claire, somente uma porta UDP é suficiente (além da porta necessária para as mensagens SIP)? Caso a resposta seja positiva, então como o cliente SIP de Alice distingue entre os pacotes RTP recebidos de Bob e Claire?

14. Considere uma sessão RTP com quatro usuários, todos eles enviando e recebendo pacotes RTP para o mesmo endereço *multicast*. Cada usuário envia vídeo a 100 kbps.

a. O RTCP limitará seu tráfego a qual taxa?

b. Quanta largura de banda RTCP será alocada a um determinado receptor?

c. Quanta largura de banda RTCP será alocada a um determinado remetente?

15. a. Quais as semelhanças entre o RTSP e o HTTP? O RSTP tem métodos? O HTTP pode ser usado para requisitar uma corrente de dados?

b. Quais as diferenças entre RTSP e HTTP? Por exemplo, o HTTP é dentro da banda ou fora da banda? O RTSP mantém informação de estado sobre o cliente (considere a função pausa/reinício)?

16. Quais são os produtos atuais da Microsoft para áudio/videoconferência em tempo real? Esses produtos usam algum dos protocolos discutidos neste capítulo (por exemplo, RTP ou RTSP)?

17. Verdadeiro ou falso:

a. Se um vídeo armazenado é entregue diretamente de um servidor Web a um transdutor, então a aplicação está usando TCP como protocolo de transporte subjacente.

b. Ao usar RTP, é possível que um remetente mude a codificação no meio de uma sessão.

c. Todas as aplicações que usam RTP devem usar a porta 87.

d. Suponha que uma sessão RTP tenha uma corrente separada de áudio e vídeo para cada remetente. Então, as correntes de áudio e vídeo usam o mesmo SSRC.

e. Em serviços diferenciados, ainda que o comportamento por salto defina diferenças de desempenho entre classes, ele não impõe nenhum mecanismo particular para alcançar esses desempenhos.

f. Suponha que Alice quer estabelecer uma sessão SIP com Bob. Ela inclui, na sua mensagem INVITE, a linha m=audio 48753 RTP/AVP 3 (AVP 3 denota áudio GSM). Portanto, Alice indicou em sua mensagem que ela deseja enviar áudio GSM.

g. Com referência à declaração anterior, Alice indicou em sua mensagem INVITE que enviará áudio para a porta 48753.

h. Mensagens SIP são enviadas tipicamente entre entidades SIP usando um número default para a porta SIP.

i. Para manter seu registro, clientes SIP têm de enviar mensagens REGISTER periodicamente.

j. SIP impõe que todos os clientes SIP suportem codificação de áudio G.711.

18. Suponha que a política de escalonamento WFQ seja aplicada a um buffer que suporta três classes; suponha que os pesos para essas três classes sejam 0,5, 0,25 e 0,25.

a. Suponha que cada classe tenha um grande número de pacotes no buffer. Em que seqüência poderiam ser atendidas essas três classes para atingir os pesos WFQ descritos? (Para escalonamento por varredura cíclica, uma seqüência natural é 123123123. . .)

b. Suponha que as classes 1 e 2 tenham um grande número de pacotes no buffer e que não haja pacotes de classe 3 no buffer. Em que seqüência as três classes poderiam ser atendidas para alcançar os pesos WFQ descritos?

19. Considere o sistema de leaky bucket (discutido na Seção 7.7), que regula a taxa média e o tamanho da rajada de um fluxo de pacotes. Queremos agora também regular a taxa de pico p. Mostre como a saída desse regulador leaky bucket pode ser alimentada para um segundo regulador de leaky bucket, de modo que os dois, em série, regulem a taxa média, a taxa de pico e o tamanho da rajada. Não esqueça de atribuir um tamanho e uma taxa de geração de permissão ao balde do segundo regulador.

20. Diz-se que um fluxo de pacote está de acordo com a especificação de um leaky bucket (r,b) com tamanho de rajada b e taxa média r se o número de pacotes que chega ao leaky bucket for menor do que $rt + b$ pacotes a cada intervalo de tempo de duração t para todo t. Um fluxo de pacotes que esteja de acordo com a especificação do leaky bucket (r,b) alguma vez terá de esperar em um regulador leaky bucket com parâmetros r e b? Justifique sua resposta.

21. Demonstre que, enquanto $r_1 < R \cdot w_1/(\Sigma w_j)$, então $d_{máx}$ é, na verdade, o atraso máximo que qualquer pacote do fluxo 1 sofrerá na fila WFQ.

Questões dissertativas

1. Como um hospedeiro pode usar realimentação de informação RTCP para determinar se os problemas são locais, regionais ou globais?

2. Na sua opinião, é melhor transmitir fluxo contínuo de áudio/vídeo por TCP ou por UDP?

3. Escreva um relatório sobre os produtos SIP da Cisco.

4. O problema de fornecimento de garantias de QoS pode ser resolvido simplesmente fornecendo largura de banda suficiente, isto é, apenas aumentando todas as capacidades de enlace, de modo que as limitações de largura de banda deixem de ser uma preocupação?

5. Um interessante mercado emergente está usando o telefone por Internet e a LAN de alta velocidade de uma empresa para substituir seu PABX (*private branch exchange* — central privada de comutação telefônica). Faça um relatório de uma página sobre esse assunto que inclua as seguintes questões:

a. O que é um PABX tradicional? Quem o utiliza?

b. Considere uma chamada entre um usuário na empresa e outro usuário fora da empresa realizada pela rede de telefonia tradicional. Que tipo de tecnologia é necessária na interface entre a LAN e a rede de telefonia tradicional?

c. Além do software do telefone por Internet e da interface da parte (b), o que mais é preciso para substituir o PABX?

6. Considere os quatro princípios de fornecimento de suporte de QoS discutidos na Seção 7.6. Descreva as circunstâncias, se houver, em que cada um desses princípios não precisa ser seguido.

Tarefa de programação

Neste laboratório você implementará um servidor e um cliente de vídeo de fluxo contínuo. O cliente usará o protocolo de fluxo contínuo em tempo real (RTSP) para controlar as ações do servidor. O servidor usará o protocolo de tempo real (RTP) para empacotar o vídeo para transporte por UDP.

Você receberá um código Java que implementará parcialmente RTSP e RTP no cliente e no servidor. Sua tarefa será concluir o código para o cliente e também para o servidor. Quando terminar, você terá criado uma aplicação cliente-servidor que faz o seguinte:

- O cliente envia comandos RTSP SETUP, PLAY, PAUSE e TEARDOWN e o servidor responde aos comandos.
- Quando o servidor estiver no estado de reprodução, ele pega periodicamente um quadro JPEG armazenado, empacota o quadro com RTP e envia o pacote RTP para um socket UDP.
- O cliente recebe os pacotes RTP, extrai os quadros JPEG, descomprime os quadros e os apresenta no seu monitor.

O código que você receberá implementa o protocolo RTSP no servidor e o desempacotamento RTP no cliente e também cuida da apresentação do vídeo transmitido. Você precisará implementar RTSP no cliente e RTP no servidor.

Esta tarefa de programação aprimorará significativamente a compreensão de RTP, RTSP e vídeo em fluxo contínuo para o aluno. Recomendamos veementemente que ela seja executada. A tarefa também sugere vários exercícios opcionais, incluindo implementação do comando RTSP DESCRIBE no cliente e também no servidor. Você encontrará informações completas sobre a tarefa, bem como snippets importantes de código Java, no Companion Website deste livro, `www.aw.com/kurose_br`.

Entrevista

Henning Schulzrinne

Henning Schulzrinne é professor associado e chefe do Internet Real-Time Laboratory na Universidade Columbia. É co-autor dos protocolos RTSP, RTP e SIP, protocolos fundamentais para comunicação de áudio e vídeo pela Internet. É graduado em engenharia elétrica e industrial pela Universidade de Darmstadt, na Alemanha, mestre em engenharia elétrica e de computação pela Universidade de Cincinnati e doutor em engenharia elétrica pela Universidade de Massachusetts, em Amherst.

O que o fez se decidir pela especialização em tecnologia de rede multimídia?

Aconteceu quase por acidente. Quando fazia doutorado, acabei me envolvendo com a DARTnet, uma rede experimental que abrangia os Estados Unidos com linhas T1. A DARTnet era usada como campo de prova para ferramentas *multicast* e de Internet em tempo real. Isso me levou a implementar minha primeira ferramenta de áudio, a NeVoT. Por intermédio de alguns participantes da DARTnet, acabei me envolvendo com a IETF, com o grupo de trabalho Audio Video Transport, que se formara havia pouco tempo naquela época. O grupo terminou por padronizar o RTP.

Qual foi o seu primeiro emprego no setor de computação? O que implicava?

No meu primeiro emprego no setor de computação fiz a soldagem das peças de um computador Altair. Eu ainda era estudante do ensino médio em Livermore, na Califórnia. Quando voltei à Alemanha, montei uma pequena empresa de consultoria que projetou um programa de gerenciamento de endereços para uma agência de viagens — armazenando dados em fitas cassete para nosso TRS-80 e usando uma máquina de escrever elétrica Selectric da IBM com uma interface de hardware feita em casa servindo de impressora.

Meu primeiro emprego verdadeiro foi no AT&T Bell Laboratories, no desenvolvimento de um emulador de rede para a construção de redes experimentais em ambiente de laboratório.

Quais os objetivos do Internet Real-Time Laboratory?

Nosso objetivo é fornecer peças para que a infra-estrutura da Internet possa ser a futura plataforma única de comunicações. Isso inclui o desenvolvimento de protocolos de sinalização, como o SIP e o RTSP, e protocolos de reserva de recursos, como o RNAP, o Yessir e o BGRP, a medição do desempenho de protocolos e aplicações da

Internet e o desenvolvimento de algoritmos para aprimorar a qualidade dos serviços na Internet. Construímos protótipos de aplicações, como servidores e clientes de telefonia por Internet, e também nos intrometemos no campo do hardware, construindo nosso próprio telefone por Internet.

Há pouco tempo começamos a investigar como multimídia e outros serviços de rede podem ser localizados pela Internet. Além disso, a tecnologia de LAN sem fio oferece novas oportunidades de distribuição de conteúdo de multimídia por duplicação local em vez de redes peer-to-peer ou redes de distribuição de conteúdo de grande porte.

Em sua opinião, qual é o futuro da rede multimídia?

Estamos agora em uma fase de transição; poucos anos nos separam do IP como plataforma universal para serviços de multimídia. Esperamos que o rádio, o telefone e a TV continuem funcionando mesmo durante tempestades de neve e terremotos; portanto, quando a Internet assumir os papéis dessas redes dedicadas, os usuários poderão esperar o mesmo nível de confiabilidade.

Até certo ponto, o trabalho com redes enfrenta o mesmo problema que a pesquisa de sistemas operacionais tem enfrentado há muitos anos, e até mais. Conquanto ainda seja possível executar sistemas operacionais experimentais de pequeno porte em pequenas comunidades, executar uma rede separada ainda frustra, e muito, seu propósito. Teremos de aprender a projetar tecnologias de rede para interconectar provedoras concorrentes com grandes quantidades de usuários finais sem instrução ou com más intenções.

Avanços notáveis nas redes multimídia serão amplamente orientados por fatores que estão fora da área em si, por exemplo, os avanços nas velocidades de acesso e de backbone, bem como na capacidade computacional a custo mais baixo.

Por que o SIP tem um futuro promissor?

Enquanto prossegue a evolução das redes sem fio existentes hoje para 3G, há a esperança de um único mecanismo de sinalização para multimídia que abranja todos os tipos de rede, desde modem a cabo até redes telefônicas de empresas e redes públicas sem fio. Juntamente com softwares de rádio, isso possibilitará que, no futuro, um dispositivo único, tal como um telefone Bluetooth sem fio, possa ser utilizado em uma rede residencial, em uma rede empresarial via 802.11 e em redes de longa distância por meio de redes 3G. Mesmo antes de existir tal dispositivo único, universal, sem fio, os mecanismos de mobilidade pessoal permitem ocultar as diferenças entre redes. Um identificador torna-se o meio universal de alcançar uma pessoa, em vez de ter de lembrar ou ter de passar por meia dúzia de números de telefones específicos para tecnologias ou localizações.

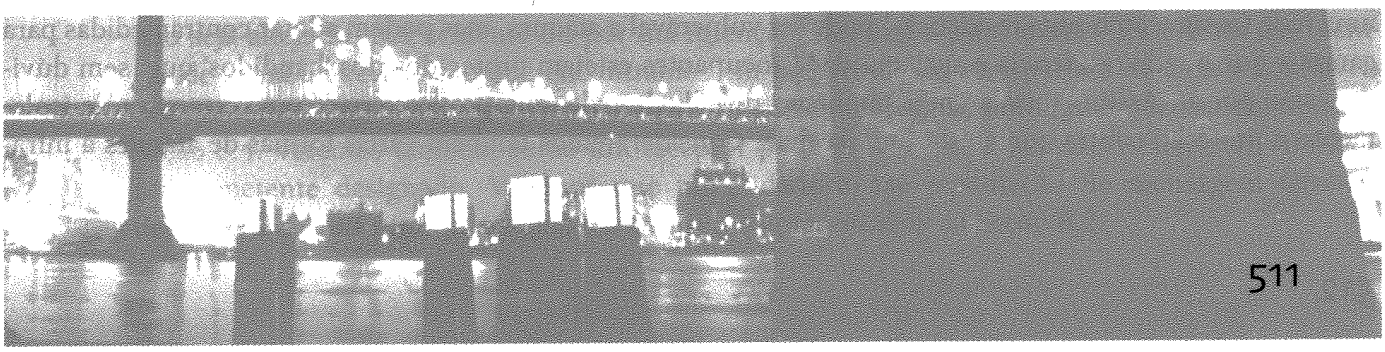
O SIP também separa o fornecimento de transporte de voz (bit) dos serviços de voz. Agora ficou tecnicamente possível acabar com o monopólio local de telefonia, no qual uma única empresa fornece transporte neutro de bits enquanto outras fornecem 'o tom de discar' IP e os serviços clássicos de telefonia, como gateways, transferência de chamadas e identificador de chamadas.

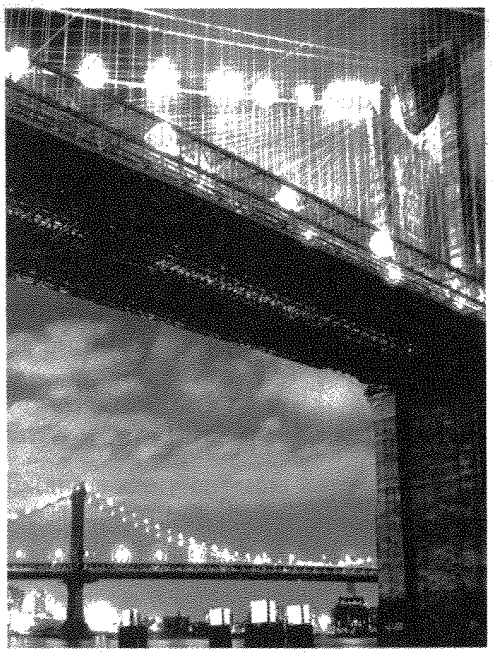
Além da sinalização de multimídia, o SIP oferece um novo serviço que estava faltando na Internet: notificação de eventos. Algo parecido com esse serviço já vem sendo oferecido por soluções improvisadas HTTP e por e-mail, mas nunca foi muito satisfatório. Uma vez que eventos são uma abstração comum para sistemas distribuídos, isso pode simplificar a construção de novos serviços.

Você pode dar algum conselho aos estudantes que estão ingressando no campo das redes?

O trabalho com redes é quase interdisciplinar. Ele tira subsídios da engenharia elétrica, da ciência da computação, da pesquisa operacional e de outras disciplinas. Assim, os pesquisadores de redes têm de estar familiarizados com assuntos que estão fora de suas áreas centrais de interesse.

O trabalho com redes pode trazer imensa satisfação, já que se trata de permitir que as pessoas se comuniquem e troquem idéias, um dos valores essenciais do ser humano. Muitas áreas da engenharia atingiram um patamar de desempenho: carros, trens e aviões ainda funcionam quase da mesma maneira que funcionavam há 20 anos, e provavelmente não vão avançar muito nos próximos 20 anos. A maior mudança será na capacidade de fazer com que essas entidades se comuniquem e, assim, aprimorem seu desempenho, tornando-se mais seguras e evitando travamentos.





Capítulo 8

Segurança em Redes de Computadores

Queremos lhe apresentar Alice e Bob, duas pessoas que desejam se comunicar, porém 'com segurança'. Como esse texto refere-se a redes, gostaríamos de observar que Alice e Bob podem ser dois roteadores que querem trocar tabelas de roteamento com segurança, um cliente e um servidor que querem estabelecer uma conexão de transporte segura ou duas aplicações de e-mail que querem trocar e-mails com segurança — todos esses tópicos serão examinados mais adiante neste capítulo. Alice e Bob são componentes conhecidos da comunidade de segurança, talvez porque o nome deles seja mais interessante do que uma entidade genérica denominada 'A' que quer se comunicar com segurança com uma entidade genérica denominada 'B'. Amores proibidos, comunicações em tempo de guerra e transações financeiras são as necessidades dos seres humanos comumente citadas quando o assunto é segurança nas comunicações; preferimos a primeira necessidade às duas últimas, e vamos usar, com muito prazer, Alice e Bob como nosso remetente e nosso destinatário e imaginá-los nesse primeiro cenário.

Dissemos que Alice e Bob querem se comunicar, porém 'com segurança', mas o que isso significa exatamente? Como veremos, a segurança (assim como o amor) é repleta de maravilhas; isto é, a segurança tem muitas facetas. É certo que Alice e Bob gostariam que o conteúdo de sua comunicação permanecesse secreto, a salvo de um bisbilhoteiro (digamos, um marido, ou esposa, ciumento). Provavelmente gostariam também de ter certeza de que estão realmente se comunicando um com o outro e de que, caso algum bisbilhoteiro interferira na comunicação, essa interferência seja detectada. Na primeira parte deste capítulo, estudaremos as técnicas que permitem criptografar/decriptar comunicações, autenticar a parte com quem estamos nos comunicando e assegurar a integridade da mensagem.

Na segunda parte deste capítulo, consideraremos o fato de que sistemas finais comunicantes executam em nós que estão dentro de uma rede empresarial ou residencial, de modo muito semelhante ao que Bob e Alice fariam se morassem em casas separadas e se comunicassem por meio do correio normal. Veremos que uma rede (assim como uma residência) pode ser vulnerável a ataques, mas podem adotar contramedidas para garantir que a rede/residência e suas entidades/ocupantes estejam protegidos dos 'bandidos' que, sem dúvida, estão por aí. Assim, abordaremos também firewalls de rede, que proporcionam um certo grau de isolamento e proteção contra quem está fora da rede. Consideraremos também várias formas de ataques à infra-

estrutura de rede e as contramedidas adotadas para frustrar esses ataques (ou ao menos atenuar seus efeitos). O capítulo termina com estudos de casos de segurança nas camadas de aplicação, transporte, rede e enlace.

8.1 O que é segurança na rede?

Vamos iniciar nosso estudo de segurança de redes voltando aos namorados citados no início, Alice e Bob, que querem se comunicar 'com segurança'. O que isso significa exatamente? Com certeza, Alice quer que somente Bob entenda a mensagem que ela enviou, mesmo que eles estejam se comunicando por um meio inseguro, em que um intruso (Trudy, a intrusa) pode interceptar, ler e executar processos computacionais com qualquer dado que seja transmitido de Alice a Bob. Bob também quer ter certeza de que a mensagem que recebe de Alice foi de fato enviada por ela, e Alice quer ter certeza de que a pessoa com quem está se comunicando é de fato Bob. Alice e Bob também querem ter certeza de que o conteúdo de suas mensagens não foi alterado em trânsito. Também querem, antes de mais nada, ter certeza de que podem se comunicar, isto é, de que ninguém lhes negue acesso aos recursos necessários para a comunicação. Dadas essas considerações, podemos identificar as seguintes propriedades desejáveis da **comunicação segura**:

- *Confidencialidade.* Somente o remetente e o destinatário pretendido devem poder entender o conteúdo da mensagem transmitida. O fato de abelhudos poderem interceptar a mensagem exige, necessariamente, que esta seja **cifrada** de alguma maneira (que seus dados sejam disfarçados) para impedir que uma mensagem interceptada seja **decifrada** (entendida) por um interceptador. Esse aspecto de confidencialidade é, provavelmente, o significado mais comumente percebido na expressão *comunicação segura*. Note, contudo, que essa não é apenas uma definição limitada de comunicação segura (relacionamos a seguir aspectos adicionais da segurança), mas também uma definição bastante restrita de *confidencialidade*. Por exemplo, Alice poderia também querer que o mero fato de ela estar se comunicando com Bob (ou os horários ou a freqüência de suas comunicações) fosse também um segredo! Estudaremos técnicas de criptografia para cifrar e decifrar dados na Seção 8.2. Veremos que comunicação confidencial muitas vezes depende de uma ou mais *chaves* utilizadas para criptografar/decriptar comunicações. Abordaremos o tópico de distribuição de chaves na Seção 8.5.
- *Autenticação.* O remetente e o destinatário precisam confirmar a identidade da outra parte envolvida na comunicação — confirmar que a outra parte realmente é quem alega ser. A comunicação pessoal entre seres humanos resolve facilmente esse problema por reconhecimento visual. Quando entidades comunicantes trocam mensagens por um meio pelo qual não podem ver a outra parte, a autenticação não é assim tão simples. Por que, por exemplo, você deveria acreditar que o e-mail que recebeu e que contém uma sentença afirmando que aquele e-mail veio de um amigo seu realmente veio daquele amigo? Se alguém o chama ao telefone dizendo ser de seu banco e perguntando qual é o número de sua conta, sua senha e seu saldo bancário, alegando finalidades de verificação, você daria essas informações? Esperamos que não. Examinaremos as técnicas de autenticação na Seção 8.3, incluindo algumas que, talvez surpreendentemente, também dependem de técnicas criptográficas, que estudaremos na Seção 8.2.
- *Integridade e não-repudiação de mensagem.* Mesmo que o remetente e o destinatário consigam se autenticar reciprocamente, eles também querem assegurar que o conteúdo de sua comunicação não seja alterado, por acidente ou por má intenção, durante a transmissão. Extensões das técnicas de soma de verificação que encontramos em protocolos de transporte e de enlace confiáveis podem ser utilizadas para proporcionar integridade à mensagem, um tópico que estudaremos na Seção 8.4. Veremos também como um receptor pode provar que a mensagem deve ter vindo de um remetente designado. Veremos que integridade e não-repudiação de mensagem também dependem de conceitos criptográficos que serão apresentados na Seção 8.2.

Disponibilidade e controle de acesso. A necessidade imperiosa de segurança na rede ficou dolorosamente óbvia nos últimos anos devido a numerosos ataques de recusa de serviço (*denial of service* — DoS) que inutilizaram uma rede, um hospedeiro, ou qualquer outro componente da infra-estrutura de rede, para seus usuários legítimos; o mais notório desses ataques DoS talvez tenha sido o cometido contra os sites Web de inúmeras empresas de alta visibilidade. Portanto, um requisito fundamental para comunicação segura deve ser, antes de mais nada, que ela possa ocorrer — que os 'bandidos' não possam impedir que a infra-estrutura seja utilizada por usuários legítimos. O fato de que alguns desses usuários possam ser legítimos, enquanto outros não, leva, naturalmente, à noção de controle de acesso, para garantir que entidades que procuram obter acesso a recursos possam fazê-lo somente se tiverem os direitos de acesso apropriados e realizarem seus acessos de uma maneira bem definida.

Confidencialidade, autenticação, integridade e não-repudiação de mensagem vêm sendo considerados componentes fundamentais da comunicação segura há bastante tempo [McCumber, 1991]. Disponibilidade e controle de acesso são extensões mais recentes da noção de comunicações seguras [Maconachy, 2001; Bishop, 2003], sem dúvida motivadas pelas preocupações muito reais com a segurança da infra-estrutura de rede contra um potencial ataque violento dos 'bandidos'. Um dos modos mais seguros para garantir que 'bandidos' não possam causar danos é assegurar, antes de mais nada, que seus pacotes não entrem na rede. Um firewall é um dispositivo situado entre a rede a ser protegida e o resto do mundo (os 'bandidos', 'eles'). Ele controla o acesso de e para a rede regulando quais pacotes podem passar para dentro e para fora da rede. Os firewalls tornaram-se rapidamente componentes comuns em redes, desde pequenas redes residenciais até as pertencentes às maiores empresas do mundo. Estudaremos como firewalls provêem controle de acesso (por pacote e por serviço) na Seção 8.6.

Nossa definição de comunicação segura concentrou-se primordialmente em *proteger* a comunicação e os recursos da rede. Na prática, a segurança na rede envolve não apenas proteção, mas também *detecção* de falhas em comunicações seguras e ataques à infra-estrutura e *reação* a esses ataques. Em muitos casos, um administrador pode implementar mecanismos especiais de proteção para reagir a ataques. Nesse sentido, a segurança da rede é conseguida por meio de um ciclo contínuo de proteção, detecção e reação. (Um livro, ligeiramente defasado, mas muito bom de ler, que ilustra aspectos deste ciclo, é [Stoll, 1995].)

Na continuação deste capítulo, abordaremos (na Seção 8.7) tipos comuns de ataques e as contramedidas que podem ser tomadas contra eles. Encerraremos o capítulo na Seção 8.8, com estudos de casos que aplicam técnicas das seções 8.2 a 8.5 nas camadas de aplicação, transporte, rede e enlace.

Agora que já determinamos o que significa segurança na rede, vamos considerar em seguida quais são, exatamente, as informações às quais um intruso pode ter acesso e que ações podem ser executadas por ele. A Figura 8.1 ilustra o cenário.

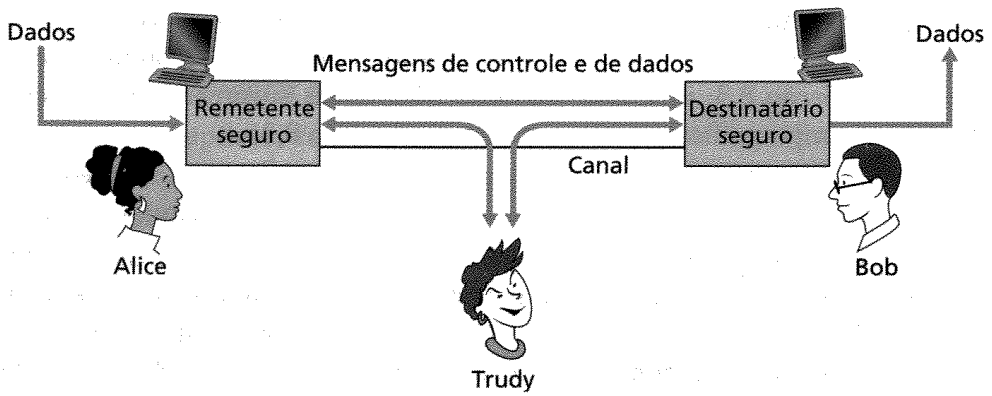


Figura 8.1 Remetente, destinatário e intruso (Alice, Bob e Trudy)

Alice, a remetente, quer enviar dados a Bob, o destinatário. Para trocar dados com segurança, além de atender aos requisitos de confidencialidade, autenticação e integridade de mensagens, Alice e Bob trocarão mensagens de controle e de dados (algo muito semelhante ao modo como remetentes e destinatários TCP trocam segmentos de controle e segmentos de dados). Todas ou algumas dessas mensagens normalmente serão criptografadas. Um intruso passivo pode, potencialmente, fazer o seguinte:

- *monitorar* — ouvir e gravar as mensagens de controle e de dados no canal;
- *modificar, inserir ou eliminar* mensagens ou conteúdo de mensagens.

Como veremos, a menos que sejam tomadas contramedidas adequadas, essas capacidades permitem que um intruso monte uma grande variedade de ataques à segurança: monitorar comunicações (e, possivelmente, roubar senhas e dados), fazer-se passar por outra entidade, seqüestrar uma sessão em curso, recusar serviço a usuários legítimos da rede sobrecarregando os recursos do sistema e assim por diante. Abordaremos esses tópicos na Seção 8.7. Essas e outras ameaças à segurança também são discutidas na coletânea de ensaios de [Denning, 1997] e no livro muito bom de ler de Rubin [Rubin, 2001]. O CERT Coordination Center [CERT, 2004] mantém um resumo de ataques comunicados. Veja também [Cisco Security, 2004; Voydock, 1983; Bhimani, 1996].

Agora que já temos certeza de que há ameaças reais (Trudy) à solta na Internet, quais são os equivalentes de Alice e Bob na Internet, esses nossos amigos que precisam se comunicar com segurança? Certamente, Bob e Alice podem ser dois usuários humanos em dois sistemas finais, por exemplo, uma Alice real e um Bob real que de fato querem trocar e-mails seguros. Eles podem também ser os participantes de uma transação de comércio eletrônico. Por exemplo, um Bob real pode querer transmitir com segurança o número de seu cartão de crédito a um servidor Web para comprar um produto pela rede. Entretanto, como descrito em [RFC 1636], os participantes que necessitam de comunicação segura podem também ser componentes da infra-estrutura da rede. Lembre-se de que o sistema de nomes de domínio (DNS, veja a Seção 2.5) ou daemons roteadores que trocam informações de roteamento (veja a Seção 4.6) requerem comunicação segura entre dois participantes. O mesmo é válido para aplicações de gerenciamento de rede, um tópico que examinaremos no Capítulo 9. Um intruso que conseguisse interferir, controlar ou corromper ativamente consultas e atualizações do DNS [RFC 2535, IETF dnsext 2004], processamentos de roteamento [Murphy, 2003] ou funções de gerenciamento de rede [RFC 2574] poderia causar uma devastação na Internet.

Estabelecida a estrutura, apresentadas algumas das definições mais importantes e justificada a necessidade de segurança na rede, vamos examinar a criptografia. Embora a utilização da criptografia para prover confidencialidade seja evidente por si só, veremos em breve que ela também é essencial para prover autenticação, integridade de mensagem, não-repudiação e controle de acesso — o que faz da criptografia uma pedra fundamental da segurança na rede.

8.2 Princípios de criptografia

Embora a criptografia tenha uma longa história que remonta, no mínimo, a Júlio César, técnicas modernas de criptografia, incluindo muitas das usadas na Internet, são baseadas em progressos feitos nos últimos 30 anos. O livro de Kahn, *The codebreakers* [Kahn, 1967], e o livro de Singh, *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography* [Singh, 1999], nos oferecem um panorama fascinante dessa longa história. [Kaufman, 1995] apresenta uma discussão técnica detalhada (mas interessante e de fácil leitura) da criptografia, em particular do ponto de vista da rede. [Diffie, 1998] fornece uma análise atraente e atualizada das questões políticas e sociais (por exemplo, privacidade) que hoje estão inextricavelmente entrelaçadas com a criptografia. Uma discussão completa sobre a criptografia exige um livro inteiro [Kaufman, 1995; Schneier, 1995]; portanto, trataremos apenas de seus aspectos essenciais, em particular do modo como as técnicas criptográficas são postas em prática na Internet. Um excelente site on-line é a página de perguntas e respostas mais freqüentes (FAQ) do RSA Labs [RSA FAQ, 2004]. Observamos também que, conquanto nessa seção focalizaremos a utilização da criptografia aplicada à confidencialidade, em

breve veremos que as técnicas criptográficas estão inextricavelmente entrelaçadas com a autenticação, a integridade de mensagens, a não-repudiação e outras.

Técnicas criptográficas permitem que um remetente disfarce os dados de modo que um intruso não consiga obter nenhuma informação dos dados interceptados. O destinatário, é claro, deve estar habilitado a recuperar os dados originais a partir dos dados disfarçados. A Figura 8.2 apresenta alguns dos componentes mais importantes da terminologia usada em criptografia.

Suponha agora que Alice queira enviar uma mensagem a Bob. A mensagem de Alice em sua forma original (por exemplo, "Bob, I love you. Alice")* é conhecida como **texto aberto** ou **texto claro**. Alice criptografa sua mensagem em texto aberto usando um **algoritmo de criptografia**, de modo que a mensagem criptografada, conhecida como **texto cifrado**, pareça ininteligível para qualquer intruso. O interessante é que em muitos sistemas criptográficos modernos, incluindo os usados na Internet, a técnica de codificação é *conhecida* — publicada, padronizada e disponível para qualquer um (por exemplo, [RFC 1321, RFC 2437, RFC 2420]), mesmo para um intruso em potencial! Evidentemente, se todos conhecem o método para codificar dados, então deve haver alguma informação secreta que impede que um intruso decifre os dados transmitidos. É aqui que entra a chave.

Na Figura 8.2, Alice fornece uma **chave**, K_A, uma cadeia de números ou de caracteres, como entrada para o algoritmo de criptografia. O algoritmo de criptografia pega essa chave e o texto aberto da mensagem, m, como entrada e produz texto cifrado como saída. A notação $K_A(m)$ refere-se à forma do texto cifrado (criptografado usando a chave K_A) da mensagem em texto aberto, m. O algoritmo criptográfico propriamente dito, que usa a chave K_A, ficará evidente do próprio contexto. De maneira semelhante, Bob fornecerá uma chave, K_B, ao **algoritmo de decriptação**, que pega o texto cifrado e a chave de Bob como entrada e produz o texto aberto original como saída. Isto é, se Bob receber uma mensagem criptografada $K_A(m)$, ele a decriptará calculando $K_B(K_A(m)) = m$. Em **sistemas de chaves simétricas**, as chaves de Bob e de Alice são idênticas e secretas. Em **sistemas de chaves públicas**, é usado um par de chaves. Uma das chaves é conhecida por Bob e por Alice (na verdade, é conhecida pelo mundo inteiro). A outra chave é conhecida apenas por Bob ou por Alice (mas não por ambos). Nas duas subseções seguintes, examinaremos com mais detalhes sistemas de chaves simétricas e de chaves públicas.

Concursos de quebra de códigos

Inicialmente adotado pelo governo dos Estados Unidos em 1977, o Data Encryption Standard (DES), um algoritmo de 56 bits, ainda é muito usado em todo o mundo por setores industriais, financeiros e outros, para proteger informações vulneráveis. A RSA Security vem patrocinando uma série de desafios para a quebra do DES, para enfatizar a necessidade de haver um código criptográfico mais resistente do que o atual padrão de 56 bits. O desafio consiste em decifrar a mensagem codificada com o DES de 56 bits em um período de tempo especificado. Os participantes enfrentam o desafio pesquisando exaustivamente todas as chaves secretas possíveis. A RSA oferece um prêmio de 10 mil dólares aos vencedores.

O primeiro desafio DES, em 1977, foi vencido por uma equipe do Colorado que descobriu a chave secreta em menos de quatro meses. Desde então, o aprimoramento da tecnologia tem possibilitado esforços muito mais rápidos e exaustivos. Em fevereiro de 1998, a Distributed.Net venceu o DES Challenge II-1 da RSA após 41 dias de pesquisa e, em julho, a Electronic Frontier Foundation (EFF) venceu o DES Challenge II-2 da RSA ao decifrar a mensagem DES em 56 horas.

Em janeiro de 1999, a Distributed.Net, uma coalizão mundial de entusiastas do computador, trabalhou com o Deep Crack da EFF, um supercomputador projetado especialmente, em parceria com uma rede mundial de aproximadamente 100 mil PCs na Internet, e venceu o DES Challenge III da RSA Data Security em apenas 22 horas e 15 minutos. O Deep Crack da EFF e os computadores da Distributed.Net estavam testando 245 bilhões de chaves por segundo quando descobriram a chave correta!

* A mensagem foi mantida no idioma original porque será apresentado o texto cifrado para ela. A tradução é "Bob, eu amo você. Alice". (N. da T.)

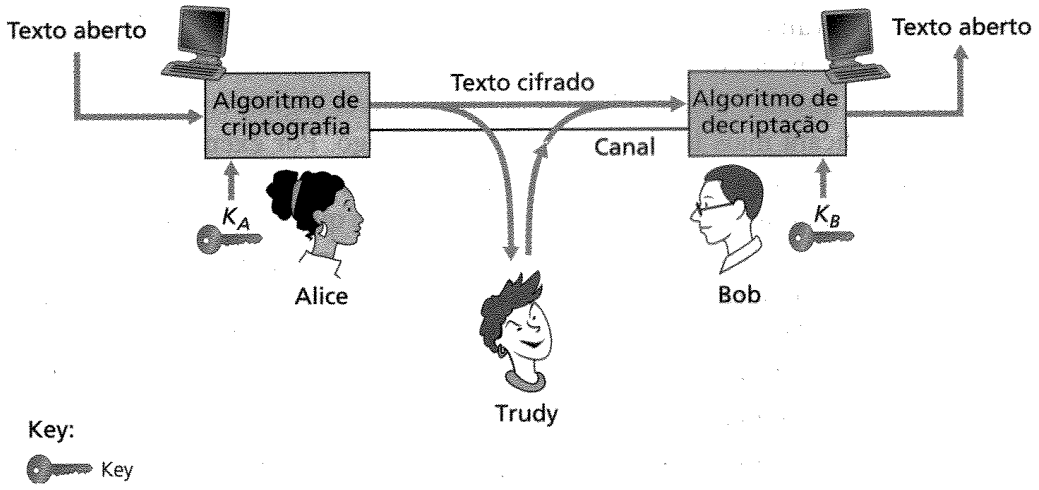


Figura 8.2 Componentes criptográficos

8.2.1 Criptografia de chaves simétricas

Todos os algoritmos criptográficos envolvem a substituição de um dado por outro, como tomar um trecho de um texto aberto e então, calculando e substituindo esse texto por outro cifrado apropriado, criar uma mensagem cifrada. Antes de estudar um sistema criptográfico moderno baseado em chaves, vamos abordar um algoritmo de chaves simétricas muito antigo, muito simples, atribuído a Júlio César e conhecido como **cifra de César** (uma cifra é um método para criptografar dados).

A cifra de César funciona tomando cada letra da mensagem do texto aberto e substituindo-a pela k-ésima letra sucessiva do alfabeto (permitindo a rotatividade do alfabeto, isto é, a letra 'z' seria seguida novamente da letra 'a'). Por exemplo, se $k = 3$, então a letra 'a' do texto aberto fica sendo 'd' no texto cifrado; 'b' no texto aberto se transforma em 'e' no texto cifrado, e assim por diante. Nesse caso, o valor de k serve de chave. Por exemplo, a mensagem "`bob, i love you. alice.`" se torna "`ere, l oryh brx. dolfh.`" em texto cifrado. Embora o texto cifrado na verdade pareça não ter nexo, você não levaria muito tempo para quebrar o código se soubesse que foi usada a cifra de César, pois há somente 25 valores possíveis para as chaves.

Um aprimoramento da cifra de César é a **cifra monoalfabética**, que também substitui uma letra do alfabeto por outra. Contudo, em vez de substituir as letras seguindo um padrão regular (por exemplo, substituição por um deslocamento de k para todas as letras), qualquer letra pode ser substituída por qualquer outra, contanto que cada letra tenha uma única letra substituta e vice-versa. A regra de substituição apresentada na Figura 8.3 mostra uma regra possível para codificar textos abertos.

A mensagem do texto aberto "`bob, I love you. alice.`" se torna "`nkn, s gktc wky. mgsbc.`" Assim, como aconteceu no caso da cifra de César, o texto parece sem nexo. A cifra monoalfabética também parece ser melhor que a cifra de César, pois há 26! (da ordem de 10^{26}) possíveis pares de letras, em vez de 25 pares possíveis!

Uma abordagem de força bruta que experimentasse todos os 10^{26} pares possíveis demandaria um esforço demasiadamente grande e impediria que esse fosse um método viável para quebrar o algoritmo criptográfico e decodificar a mensagem. Contudo, pela análise estatística da linguagem do texto aberto, por exemplo,

```
Letra no texto aberto: a b c d e f g h i j k l m n o p q r s t u v w x y z
Letra no texto cifrado: m n b v c x z a s d f g h j k l p o i u y t r e w q
```

Figura 8.3 Uma cifra monoalfabética

e sabendo que as letras 'e' e 't' são as mais freqüentes em textos em inglês (13 por cento e 9 por cento das ocorrências de letras, respectivamente) e sabendo também que determinados grupos de duas e de três letras aparecem com bastante freqüência em inglês (por exemplo, 'in', 'it', 'the', 'ion', 'ing', e assim por diante), torna-se relativamente fácil quebrar esse código. Se o intruso tiver algum conhecimento sobre o possível texto da mensagem, então ficará mais fácil ainda quebrar o código. Por exemplo, se a intrusa Trudy for a esposa de Bob e suspeitar que ele está tendo um caso com Alice, ela poderá facilmente imaginar que os nomes 'bob' e 'alice' apareçam no texto. Se Trudy tivesse certeza de que esses dois nomes aparecem no texto cifrado e tivesse uma cópia do texto cifrado da mensagem do exemplo, então ela poderia determinar imediatamente sete dos 26 pares de letras, o que resultaria em 10^9 possibilidades a menos para verificar pelo método da força bruta. Na verdade, se Trudy suspeitasse que Bob estava tendo um caso, ela poderia muito bem esperar encontrar algumas outras palavras preferenciais no texto também.

Considerando como seria fácil para Trudy quebrar o código criptográfico de Bob e Alice, podemos distinguir três cenários diferentes, dependendo do tipo de informação que o intruso tem:

- *Ataque exclusivo a texto cifrado.* Em alguns casos, o intruso pode ter acesso somente ao texto cifrado interceptado, sem ter nenhuma informação exata sobre o conteúdo do texto aberto. Já vimos como a análise estatística pode ajudar o **ataque exclusivo ao texto cifrado** em um esquema criptográfico.
- *Ataque com texto aberto conhecido.* Vimos anteriormente que, se Trudy, por alguma razão, tivesse certeza de que 'bob' e 'alice' apareciam no texto cifrado, ela poderia determinar os pares (texto cifrado, texto aberto) para as letras *a*, *l*, *i*, *c*, *e*, *b* e *o*. Trudy poderia também ser muito sortuda e ter gravado todas as transmissões de texto cifrado e descoberto uma versão decifrada pelo próprio Bob escrita em um pedaço de papel. Quando um intruso conhece alguns dos pares (texto aberto, texto cifrado), referimo-nos a isso como ataque ao esquema criptográfico a partir de **texto aberto conhecido**.
- *Ataque com texto aberto escolhido.* Nesse tipo de ataque, o intruso pode escolher a mensagem em texto aberto e obter seu texto cifrado correspondente. Para os algoritmos criptográficos simples que vimos até aqui, se Trudy conseguisse que Alice enviasse a mensagem "The quick fox jumps over the lazy brown dog",* ela poderia decifrar completamente o esquema criptográfico. Veremos em breve que, para técnicas de criptografia mais sofisticadas, um ataque com um texto aberto escolhido não significa necessariamente que a técnica criptográfica possa ser decifrada.

Quinhentos anos atrás foram inventadas técnicas que aprimoravam a cifra monoalfabética, conhecidas como cifras polialfabéticas. A idéia subjacente à **criptografia polialfabética** é usar várias cifras monoalfabéticas com uma cifra monoalfabética específica para codificar uma letra em uma posição específica no texto aberto da mensagem. Assim, a mesma letra, quando aparece em posições diferentes no texto aberto da mensagem, pode ser codificada de maneira diferente. Um exemplo de esquema criptográfico polialfabético é mostrado na Figura 8.4, na qual há duas cifras de César (com $k = 5$ e $k = 19$), que aparecem nas linhas da figura. Podemos optar pelo uso dessas duas cifras de César, C_1 e C_2, seguindo o modelo de repetição C_1, C_2, C_2, C_1, C_2. Isto é, a primeira letra do texto deve ser cifrada usando-se C_1, a segunda e a terceira, C_2, a quarta, C_1 e a quinta, C_2. O modelo, então, se repete, com a sexta letra sendo cifrada usando-se C_1; a sétima, com C_2, e assim por diante. Dessa maneira, a mensagem em texto aberto "bob, I love you." é cifrada como "ghu, n etox dhz.". Note que o primeiro 'b' da mensagem em texto aberto é cifrado usando-se C_1, ao passo que o segundo 'b' é cifrado usando-se C_2. Nesse exemplo, a 'chave' da codificação e da decodificação é o conhecimento das duas cifras de César ($k = 5$, $k = 19$) e do modelo C_1, C_2, C_2, C_1, C_2.

* A frase citada é célebre em textos de criptografia por ser uma frase com sentido que contém todas as letras do alfabeto inglês e possui um número mínimo de caracteres (35). (N. do R.T.)

Letra do texto aberto:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
$C_1(k = 5)$:	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e
$C_2(k = 19)$:	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s

Figura 8.4 Uma cifra polialfabética que utiliza duas cifras de César

Padrão para criptografia de dados (DES) e Padrão avançado de criptografia (AES)

Vamos agora avançar rapidamente até os tempos modernos e examinar o DES (*data encryption standard* — padrão para criptografia de dados) [NIST, 1993], um padrão de criptografia de chaves simétricas desenvolvido em 1977 e atualizado mais recentemente, em 1993, pelo U.S. National Bureau of Standards para uso comercial e não classificado do governo norte-americano. O DES codifica texto aberto em porções de 64 bits usando uma chave de 64 bits. Na verdade, oito desses 64 bits da chave são bits de paridade ímpar (há um bit de paridade para cada um dos 8 bits), de modo que a chave DES tem efetivamente 56 bits de comprimento. O National Institute of Standards and Technology (sucessor do National Bureau of Standards) assim estabelece o objetivo do DES: "O objetivo é embaralhar completamente os dados e a chave, de modo que todos os bits do texto cifrado dependam de todos os bits de dados e de todos os bits da chave (...) com um bom algoritmo, não deverá haver nenhuma correlação entre o texto cifrado e os dados originais e a chave" [NIST, 1999a].

A operação básica do DES é ilustrada na Figura 8.5. Em nossa discussão, vamos descrever de maneira geral a operação do DES, deixando os detalhes fundamentais de nível de bits (há muitos) para outras fontes [Kaufman, 1995; Schneier, 1995]. [Schneier, 1995] fornece também uma implementação em linguagem C. O DES consiste em dois estágios de permutação (o primeiro e o último passos do algoritmo) nos quais todos os 64 bits são permutados, havendo 16 rodadas idênticas de operação entre eles. A operação de cada rodada é idêntica e toma a saída de dados da rodada anterior como entrada. Durante cada rodada, os 32 bits da extrema direita da entrada são deslocados para os 32 bits da esquerda da saída. Toda a entrada de 64 bits até a i-ésima rodada e a chave de 48 bits para a i-ésima rodada (derivada da chave DES maior de 56 bits) são tomadas como entrada para uma função que envolve a expansão das porções de 4 bits da entrada para porções de 6 bits, fazendo-se o OU exclusivo com as porções expandidas de 6 bits da chave K_i de 48 bits, uma operação de substituição, e o ulterior OU exclusivo com os 32 bits da extrema esquerda da entrada; consulte [Kaufman, 1995; Schneier, 1995] para mais detalhes. A saída de 32 bits resultante da função é então usada como os 32 bits da extrema direita da saída de 64 bits da rodada, como mostra a Figura 8.5. A decodificação funciona pela reversão das operações dos algoritmos.

Até que ponto o DES funciona? Até que ponto ele é seguro? Ninguém pode ter certeza [Kaufman, 1995]. Em 1997, uma empresa de segurança de redes, a RSA Data Security Inc., lançou um desafio (DES Challenge) para quebrar (decodificar) uma frase curta que tinha sido criptografada usando o DES de 56 bits. A frase "`Strong cryptography makes the world a safer place.`" (a boa criptografia faz do mundo um lugar mais seguro) foi decodificada em menos de quatro meses por uma equipe que usou voluntários por toda a Internet para explorar sistematicamente o espaço de chaves. A equipe reivindicou o prêmio de 10 mil dólares após ter testado apenas um quarto do espaço de chaves — cerca de 18 quatrilhões de chaves [RSA Challenge, 2002]. O desafio DES Challenge III, de 1999, foi vencido em pouco mais de 22 horas por uma rede de voluntários e um computador especialmente construído para a ocasião, que custou menos de 250 mil dólares (apelidado de 'Deep Crack'), e está documentado na Internet [EFF, 1999].

Devemos observar também que, até aqui, consideramos apenas a criptografia de uma quantidade de 64 bits. Quando são criptografadas mensagens mais longas, o que é tipicamente o caso, o DES é em geral usado junto com a técnica conhecida como **encadeamento de blocos de cifras**, na qual a versão criptografada da j-ésima quantidade de 64 bits e a unidade de dados de ordem $(j + 1)$ são combinadas através de uma operação "OU exclusiva" antes de a unidade de ordem $(j + 1)$ ser criptografada.

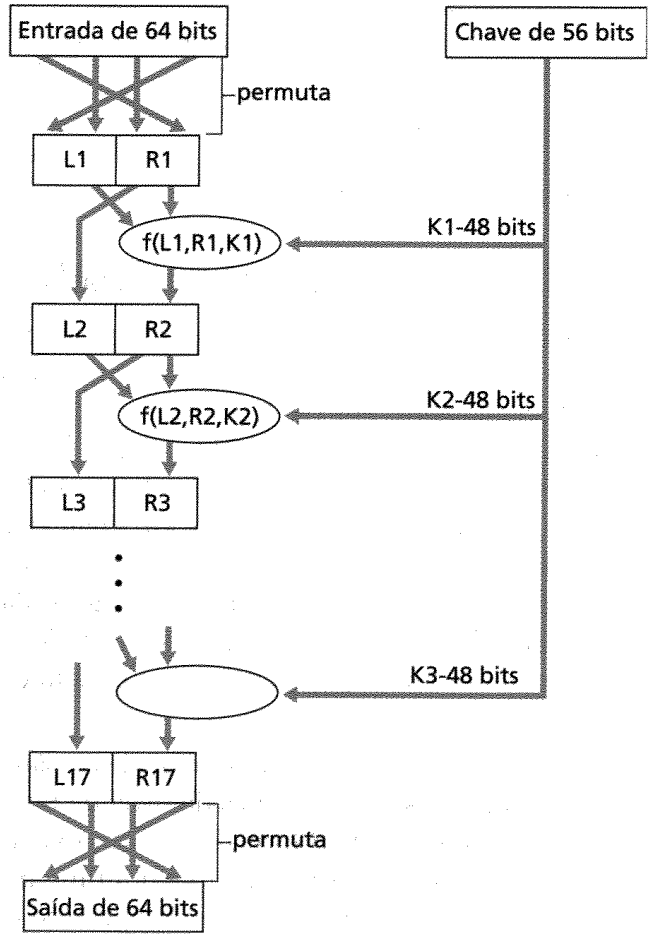


Figura 8.5 Operação básica do DES

Se o DES de 56 bits for considerado muito inseguro, pode-se simplesmente executar o algoritmo de 56 bits várias vezes, tomando a saída de 64 bits de uma iteração do DES como entrada para a iteração DES seguinte e usando uma chave criptográfica diferente para cada rodada. Por exemplo, o **DES triplo (3DES)** é um padrão proposto pelo governo norte-americano [NIST, 1999b] para substituir o DES, o qual está sendo desativado e é permitido apenas em sistemas legados. Em uma configuração, o 3DES executa primeiro o algoritmo de criptografia DES sobre os dados utilizando uma primeira chave de 56 bits; em seguida, executa o algoritmo de decriptação do DES sobre a saída da primeira rodada de criptografia usando uma segunda chave e, finalmente, executa o algoritmo de criptografia DES sobre a saída da segunda rodada usando uma terceira chave. O 3DES foi proposto como o padrão criptográfico para o PPP (protocolo ponto-a-ponto) [RFC 2420] para a camada de enlace (veja a Seção 5.7). Uma discussão detalhada sobre os comprimentos das chaves e sobre o tempo e o orçamento estimados necessários para quebrar o 3DES pode ser encontrada em [Blaze, 1996].

Em novembro de 2001, o NIST anunciou o sucessor do DES: o Padrão Avançado de Criptografia (Advanced Encryption Standard — AES) [NIST 2001; Danielyan, 2001], também conhecido como algoritmo de Rijndael [Daemen, 2000]. O AES é um algoritmo de chave simétrica que processa dados em blocos de 128 bits e pode funcionar com chaves de 128, 192 e 256 bits de comprimento. O NIST estima que uma máquina que conseguisse quebrar o DES de 56 bits em 1 segundo (isto é, experimentar 2^{55} chaves por segundo) levaria aproximadamente 149 trilhões de anos para quebrar uma chave AES de 128 bits.

8.2.2 Criptografia de chave pública

Por mais de dois mil anos (desde a época da cifra de César até a década de 1970), a comunicação cifrada exigia que as duas partes comunicantes compartilhassem um segredo em comum — a chave simétrica usada para cifrar e decifrar. Uma dificuldade dessa abordagem é que as duas partes têm de concordar, de alguma maneira, com a chave compartilhada, mas, para fazê-lo, é preciso comunicação (presumivelmente segura)! Talvez as partes pudessem se encontrar antes, escolher a chave pessoalmente (por exemplo, dois dos centuriões de César poderiam se encontrar nos banhos romanos) e, mais tarde, se comunicar de modo cifrado. Em um mundo em rede, contudo, o mais provável é que as partes comunicantes nunca possam se encontrar e jamais possam conversar a não ser pela rede. É possível que elas se comuniquem por criptografia sem compartilhar uma chave comum secreta conhecida com antecedência? Em 1976, Diffie e Hellman [Diffie, 1976] apresentaram um algoritmo (conhecido como Troca de Chaves Diffie Hellman — Diffie Hellman Key Exchange) que faz exatamente isso — uma abordagem da comunicação segura radicalmente diferente e de uma elegância maravilhosa que levou ao desenvolvimento dos atuais sistemas de criptografia de chaves públicas. Veremos em breve que os sistemas de criptografia de chaves públicas também têm diversas propriedades maravilhosas que os tornam úteis não somente para criptografia, mas também para autenticação e assinaturas digitais. É interessante que, recentemente, veio à luz que idéias semelhantes às de [Diffie, 1976] e às da [RSA, 1978] foram desenvolvidas independentemente no início da década de 1970 em uma série de relatórios secretos escritos por pesquisadores do Grupo de Segurança para Comunicação e Eletrônica (Communications-Electronics Security Group) do Reino Unido [Ellis, 1987]. Como acontece com freqüência, grandes idéias podem surgir de modo independente em diversos lugares; felizmente, os progressos da criptografia de chaves públicas ocorreram não apenas no âmbito privado, mas também no público.

A utilização de criptografia de chaves públicas, como conceito, é bastante simples. Suponha que Alice queira se comunicar com Bob. Como mostra a Figura 8.6, em vez de Bob e Alice compartilharem uma única chave secreta (como no caso dos sistemas de chaves simétricas), Bob (o destinatário das mensagens de Alice) tem duas chaves — uma **chave pública**, que está à disposição do *mundo todo* (inclusive à disposição de Trudy, a intrusa), e uma **chave privada**, que apenas ele (Bob) conhece. Usaremos a notação K_B^+ e K_B^- para nos referirmos às chaves pública e privada de Bob, respectivamente. Para se comunicar com Bob, Alice busca primeiramente a chave pública de Bob. Em seguida, ela criptografa sua mensagem, m, usando a chave pública de Bob e um algoritmo criptográfico conhecido (por exemplo, padronizado), isto é, Alice calcula $K_B^+(m)$. Bob recebe a mensagem criptografada de Alice e usa sua chave privada e um algoritmo de decriptação conhecido (por exemplo, padronizado) para decifrar a mensagem de Alice, isto é, Bob calcula $K_B^-(K_B^+(m))$. Veremos, mais adiante, que há algoritmos e técnicas de criptografia/decriptação para escolher chaves públicas e privadas de modo que $K_B^-(K_B^+(m)) = m$; isto é, aplicando a chave pública de Bob, K_B^+, à mensagem m, para obter $K_B^+(m)$, e então aplicando a chave privada de Bob, K_B^-, à versão criptografada de m, isto é, calculando $K_B^-(K_B^+(m))$, obtemos m novamente. Esse resultado é notável! Dessa maneira Alice pode utilizar a chave de Bob disponível publicamente para enviar uma mensagem secreta a Bob sem que nenhum deles tenha de permutar nenhuma chave secreta! Veremos em breve que nós podemos permutar a chave pública e a chave privada de criptografia e obter o mesmo resultado notável, isto é, $K_B^-(K^+(m)) = K_B^+(K_B^-(m)) = m$.

A utilização de criptografia de chave pública é, portanto, conceitualmente simples. Mas há duas preocupações que podem nos ocorrer imediatamente. A primeira preocupação é que, embora um intruso que intercepte a mensagem cifrada de Alice veja apenas dados ininteligíveis, ele conhece tanto a chave (a chave pública de Bob, que está disponível a todos) quanto o algoritmo que Alice usou para a criptografia. Assim, Trudy pode montar um ataque com texto aberto escolhido utilizando o algoritmo criptográfico padronizado conhecido e a chave criptográfica de acesso público de Bob para codificar a mensagem que quiser! Ela pode até tentar, por exemplo, codificar mensagens, ou partes delas, que suspeita que Alice poderia enviar. Fica claro que, para a criptografia de chave pública funcionar, a escolha de chaves e de códigos de criptografia/decriptação deve ser feita de tal maneira que seja impossível (ou, ao menos, tão difícil que se torne

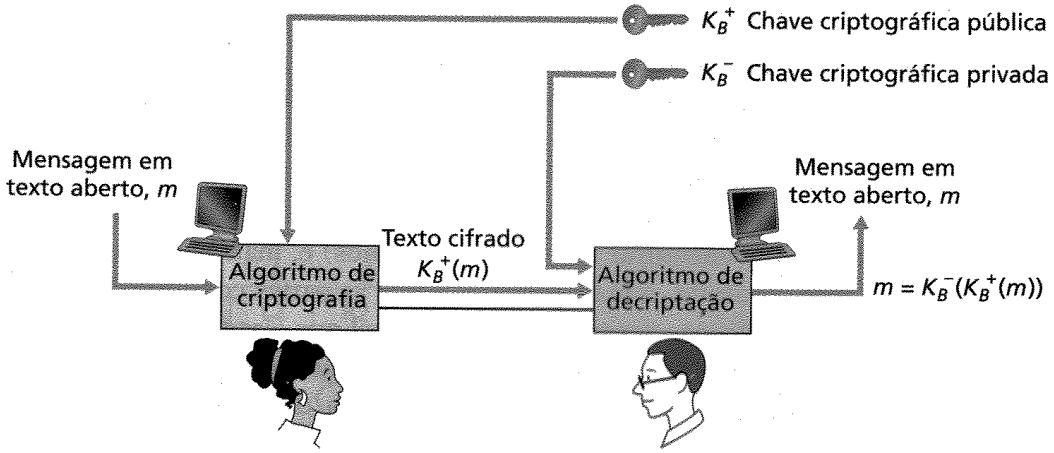


Figura 8.6 Criptografia de chaves públicas

quase impossível) para um intruso determinar a chave privada de Bob ou conseguir decifrar ou adivinhar a mensagem de Alice a Bob. A segunda preocupação é que, como a chave criptográfica de Bob é pública, qualquer um pode enviar uma mensagem cifrada a Bob, incluindo Alice ou alguém *que se passa* por Alice. No caso de uma única chave secreta compartilhada, o fato de o remetente conhecer a chave secreta identifica implicitamente o remetente para o destinatário. No caso da criptografia de chave pública, contudo, isso não acontece, já que qualquer um pode enviar uma mensagem cifrada a Bob usando a chave dele, que está publicamente disponível a todos. É preciso uma assinatura digital, um tópico que estudaremos na Seção 8.4, para vincular um remetente a uma mensagem.

Embora existam muitos algoritmos e chaves que tratam dessas preocupações, o **algoritmo RSA** (cujo nome se deve a seus inventores, Ron Rivest, Adi Shamir e Leonard Adleman) tornou-se quase um sinônimo de criptografia de chave pública. Em primeiro lugar, vamos ver como o RSA funciona e, depois, examinar por que ele funciona. Suponha que Bob queira receber mensagens cifradas, como mostra a Figura 8.6. Há dois componentes inter-relacionados no RSA:

- A escolha da chave pública e da chave privada.
- O algoritmo de criptografia/decriptação.

Para escolher as chaves pública e privada, Bob deve executar as seguintes etapas:

1. Escolher dois números primos grandes, p e q. Que ordem de grandeza devem ter p e q? Quanto maiores os valores, mais difícil será quebrar o RSA, mas mais tempo se levará para realizar a codificação e a decodificação. O RSA Laboratories recomenda que o produto de p e q seja da ordem de 1024 bits para uso empresarial e de 768 bits para utilização com 'informações menos valiosas' [RSA Key, 2004] (o que nos leva a pensar por que o uso empresarial é considerado muito mais importante do que outros usos!). Para uma discussão sobre como achar números primos grandes, consulte [Caldwell, 2004].
2. Computar $n = pq$ e $z = (p - 1)(q - 1)$.
3. Escolher um número e menor do que n que não tenha fatores comuns (exceto o 1) com z. (Nesse caso, dizemos que e e z são números primos entre si.) A letra 'e' é usada já que esse valor será utilizado na criptografia ('encryption', em inglês).
4. Achar um número d, tal que $ed - 1$ seja divisível exatamente (isto é, não haja resto na divisão) por z. A letra 'd' é usada porque seu valor será utilizado na decriptação. Em outras palavras, dado e, escolhemos d tal que o resto da divisão de ed por z seja o número inteiro 1. (O número inteiro, que é o resto da divisão de x por um inteiro n, é denominado x mod n).

5. A chave pública que Bob põe à disposição de todos, K_B^+, é o par de números (n,e); sua chave privada, K_B^-, é o par de números (n,d).

A criptografia feita por Alice e a decriptação feita por Bob acontecem como segue:

Suponha que Alice queira enviar a Bob um padrão de bits, ou número m, tal que $m < n$. Para codificar, Alice calcula a potência m^e e, então, determina o resto inteiro da divisão de m^e por n. Assim, o valor cifrado, c, da mensagem em texto aberto, m, que Alice envia é:

$$c = m^e \bmod n.$$

Para decifrar a mensagem em texto cifrado recebida, c, Bob processa

$$m = c^d \bmod n,$$

que exige o uso de sua chave secreta (n,d).

Como exemplo simples de RSA, suponha que Bob escolha $p = 5$ e $q = 7$. (Admitimos que esses valores são muito pequenos para ser seguros.) Então, $n = 35$ e $z = 24$. Bob escolhe $e = 5$, já que 5 e 24 não têm fatores comuns. Por fim, ele escolhe $d = 29$, já que $5 \cdot 29 - 1$ (isto é, $ed - 1$) é divisível exatamente por 24. Ele divulga os dois valores, $n = 35$ e $e = 5$, e mantém em segredo o valor $d = 29$. Observando esses dois valores públicos, suponha que Alice queira agora enviar as letras 'l', 'o', 'v' e 'e' a Bob. Interpretando cada letra como um número entre 1 e 26 (com 'a' sendo 1 e 'z' sendo 26), Alice e Bob realizam a criptografia e a decriptação mostradas nas tabelas 8.1 e 8.2, respectivamente.

Dado que o exemplo fictício das tabelas 8.1 e 8.2 já produziu alguns números extremamente grandes e visto que sabemos, porque vimos anteriormente, que p e q devem ter, cada um, algumas centenas de bits de comprimento, várias questões práticas nos vêm à mente no caso do RSA. Como escolher números primos grandes? Como escolher e e d? Como calcular exponenciais de números grandes? A discussão desses assuntos está além do escopo deste livro; consulte [Kaufman, 1995] e as referências ali citadas para mais detalhes.

Observamos aqui que a exponenciação exigida pelo RSA é um processo que consome tempo considerável. O DES, ao contrário, é, no mínimo, cem vezes mais veloz em software e entre mil e dez mil vezes mais veloz em hardware [RSA Fast, 2004]. Como resultado, o RSA é freqüentemente usado na prática em combinação com o DES ou com o AES. Por exemplo, se Alice quer enviar a Bob uma grande quantidade de dados cifrados à alta velocidade, ela pode fazer o seguinte. Primeiramente ela escolhe uma chave DES que será utilizada para codificar os dados em si; essa chave às vezes é denominada **chave de sessão**, K_S. Alice deve informar a Bob essa chave de sessão, já que essa é a chave simétrica compartilhada que eles usarão para o DES. Ela então criptografa o valor da chave de sessão usando a chave pública RSA de Bob, isto é, ela processa $c = (K_S)^e \bmod n$. Bob recebe a chave de sessão codificada RSA, c, e a decifra para obter a chave de sessão K_S. Ele agora conhece a chave que Alice usará para transferir dados cifrados em DES.

Letra do texto aberto	m: representação numérica	m^e	Texto cifrado $c = m^e \bmod n$
l	12	248832	17
o	15	759375	15
v	22	5153632	22
e	5	3125	10

Tabela 8.1 Criptografia RSA para Alice: $e = 5$, $n = 35$

Texto cifrado c	c^d	$m = c^d \bmod n$	Letra do texto aberto
17	481968572106750915091411825223071697	12	l
15	127834039488589391112327575683593 75	15	o
22	851643319086537701956194499721110603 0592	22	v
10	10000000000000000000000000000000	5	e

Tabela 8.2 Decriptação RSA para Bob: $d = 29$, $n = 35$

Por que o RSA funciona?

A criptografia/decriptação do RSA parece mágica. Por que será que, aplicando o algoritmo de criptografia e, em seguida, o algoritmo de decriptação, podemos recuperar a mensagem original? Para entender por que o RSA funciona, precisamos realizar operações usando a aritmética de módulo n. Em aritmética modular, são realizadas as operações comuns de adição, multiplicação e exponenciação. Contudo, o resultado de cada operação é substituído pelo resto inteiro da divisão de cada resultado por n. Tomemos $n = pq$, onde p e q são os números primos grandes usados no algoritmo RSA.

Lembre-se de que na criptografia RSA uma mensagem (representada por um número inteiro) m é primeiramente elevada à potência e usando-se aritmética de módulo n para criptografar. A decriptação é feita elevando-se esse valor à potência d, novamente usando a aritmética de módulo n. O resultado de uma etapa de criptografia, seguida de uma etapa de decriptação, é então $(m^e)^d$. Vamos ver agora o que podemos dizer sobre essa quantidade. Temos:

$$(m^e)^d \bmod n = m^{ed} \bmod n.$$

Embora estejamos tentando eliminar um pouco da mágica do modo de funcionamento do RSA, precisaremos usar, aqui, outro resultado bastante mágico da teoria dos números. Especificamente, precisamos do resultado que diga que, se p e q forem primos e $n = pq$, então $x^y \bmod n$ será o mesmo que $x^{(y \bmod (p-1)(q-1))} \bmod n$ [Kaufman, 1995]. Aplicando esse resultado, temos

$$(m^e)^d \bmod n = m^{(ed \bmod (p-1)(q-1))} \bmod n.$$

Mas lembre-se de que escolhemos e e d tais que $ed - 1$ seja exatamente divisível (isto é, não há resto) por $(p-1)(q-1)$, o que equivale a dizer que ed é divisível por $(p-1)(q-1)$ com resto 1 e então $ed \bmod (p-1)(q-1) = 1$. Isso nos dá

$$(m^e)^d \bmod n = m^1 \bmod n = m,$$

ou seja,

$$(m^e)^d \bmod n = m$$

Esse é o resultado que esperávamos! Efetuando primeiramente a exponenciação da potência e (isto é, criptografando) e depois a exponenciação da potência d (isto é, decriptando), obtemos o valor original m. E mais notável ainda é o fato de que, se primeiramente elevarmos à potência d e, em seguida, à potência e, isto é, se invertermos a ordem da criptografia e da decriptação, realizando inicialmente a operação de decriptação e, em seguida, aplicando a operação de criptografia, também obteremos o valor original m! (A prova desse resultado segue exatamente o mesmo raciocínio anterior, visto que $m^{ed} = m^{de}$.) Veremos em breve que essa propriedade maravilhosa do algoritmo RSA

$$(m^e)^d \bmod n = m = (m^d)^e \bmod n$$

será de grande utilidade.

A segurança do RSA reside no fato de que não se conhecem algoritmos para fatorar rapidamente um número; nesse caso, o valor público n, em números primos p e q. Se alguém conhecesse os números p e q, então, dado o valor público e, poderia facilmente processar a chave secreta d. Por outro lado, não se sabe se *existem* ou não algoritmos rápidos para fatorar um número e, nesse sentido, a segurança do RSA não é garantida.

8.3 Autenticação

Autenticação é o processo de provar a própria identidade a alguém. Como seres humanos, autenticamo-nos mutuamente de muitas maneiras: reconhecemos mutuamente nosso rosto quando nos encontramos, reconhecemos mutuamente nossa voz ao telefone, somos autenticados pela autoridade alfandegária que nos compara à foto em nosso passaporte.

Nesta seção, veremos como uma parte pode autenticar uma outra parte quando as duas estão se comunicando por uma rede. Examinaremos aqui a autenticação de uma parte 'ao vivo', no instante em que a comunicação está realmente ocorrendo. Veremos que há uma diferença sutil entre esse problema e o problema de provar que uma mensagem recebida em algum momento no passado (e que pode ter sido arquivada, por exemplo) é, na verdade, de quem se declarou como o remetente. Esse último problema é denominado problema da **assinatura digital**, um assunto que examinaremos na Seção 8.4.

Ao fazerem autenticação pela rede, as partes comunicantes não podem confiar em informações biométricas, como a aparência visual ou a voz característica. Na verdade, veremos em nossos estudos de caso posteriores que muitas vezes são componentes da rede, como roteadores e processos cliente-servidor, que devem se autenticar mutuamente. Aqui, a autenticação deve ser feita exclusivamente na base de mensagens e de dados trocados como parte de um **protocolo de autenticação**. Normalmente, um protocolo de autenticação seria executado antes que as duas partes comunicantes executassem qualquer outro protocolo (por exemplo, um protocolo de transferência confiável de dados, um protocolo de troca de informações de roteamento ou um protocolo de e-mail). O protocolo de autenticação estabelece primeiramente as identidades das partes de maneira satisfatória para ambas; somente após a autenticação, as partes se lançam à tarefa que têm em mãos.

Como no caso do desenvolvimento de nosso protocolo de transferência confiável de dados (rdt), no Capítulo 3, achamos instrutivo desenvolver várias versões de um protocolo de autenticação, que denominaremos **pa** (protocolo de autenticação) e que apresentarão alguns furos em cada versão à medida que avançarmos. (Se você gostar dessa evolução por etapas de um projeto, gostará também de [Bryant, 1988], que conta uma história fictícia entre projetistas de um sistema aberto de autenticação em rede e de como eles descobrem muitas questões sutis envolvidas.)

Vamos admitir que Alice precisa se autenticar para Bob.

8.3.1 Protocolo de autenticação ap1.0

O protocolo de autenticação mais simples que podemos imaginar talvez seja um protocolo em que Alice apenas envia uma mensagem a Bob dizendo que ela é Alice. Esse protocolo é mostrado na Figura 8.7. A falha aqui é óbvia — não há meios de Bob saber se a pessoa que está enviando a mensagem "Eu sou Alice" é realmente Alice. Por exemplo, Trudy (a intrusa) poderia muito bem enviar essa mensagem.

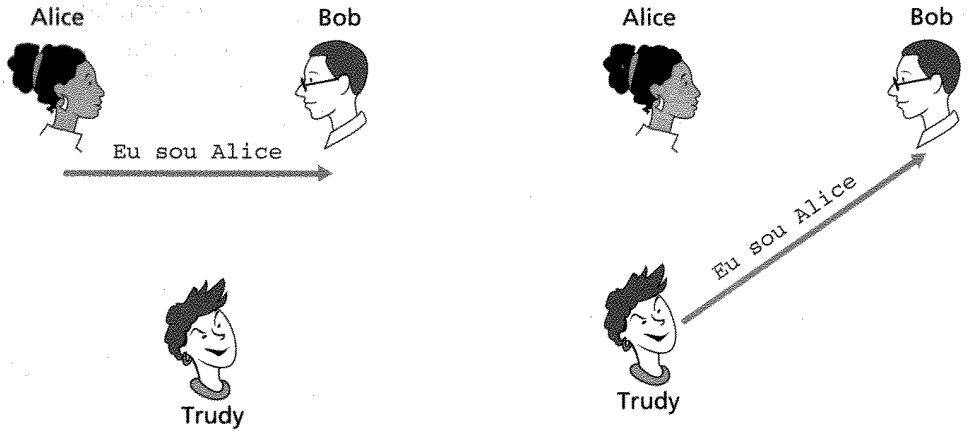


Figura 8.7 Protocolo ap1.0 e um cenário de falha

8.3.2 Protocolo de autenticação ap2.0

Se Alice tiver um endereço de rede conhecido (por exemplo, um endereço IP), que usa sempre que quer se comunicar, Bob poderia tentar autenticar Alice verificando se o endereço de fonte no datagrama IP que carrega a mensagem de autenticação é aquele mesmo endereço conhecido de Alice. Se for, Alice estará autenticada. Isso poderia impedir que um intruso de rede muito ingênuo se fizesse passar por Alice, mas não deteria o estudante determinado que está lendo este livro, ou muitos outros!

Dado que já estudamos as camadas de rede e de enlace, sabemos que não é tão difícil (por exemplo, se alguém tivesse acesso ao código do sistema operacional e pudesse construir seu próprio núcleo de sistema operacional, como é o caso do Linux e de vários outros sistemas operacionais de acesso grátis) criar um datagrama IP, colocar qualquer endereço IP de fonte que quiséssemos (por exemplo, o endereço IP conhecido de Alice) dentro do datagrama IP e enviá-lo pelo protocolo de camada de enlace ao roteador de primeiro salto. Dali em diante, o datagrama incorretamente endereçado seria transmitido até Bob. Essa abordagem, mostrada na Figura 8.8, é um tipo de falsificação do IP, uma técnica muito conhecida de ataque à segurança que estudaremos com mais detalhes na Seção 8.7. A falsificação de IP poderá ser evitada se o roteador de primeiro salto de Trudy estiver configurado para repassar apenas datagramas que contivessem o endereço IP de fonte da própria Trudy [RFC 2827]. Contudo, essa capacidade não é universalmente disseminada nem obrigatória. Assim, seria tolice de Bob achar que o administrador de rede de Trudy (que poderia ser ela mesma) configuraria o roteador de primeiro salto dela para repassar apenas os datagramas adequadamente endereçados.

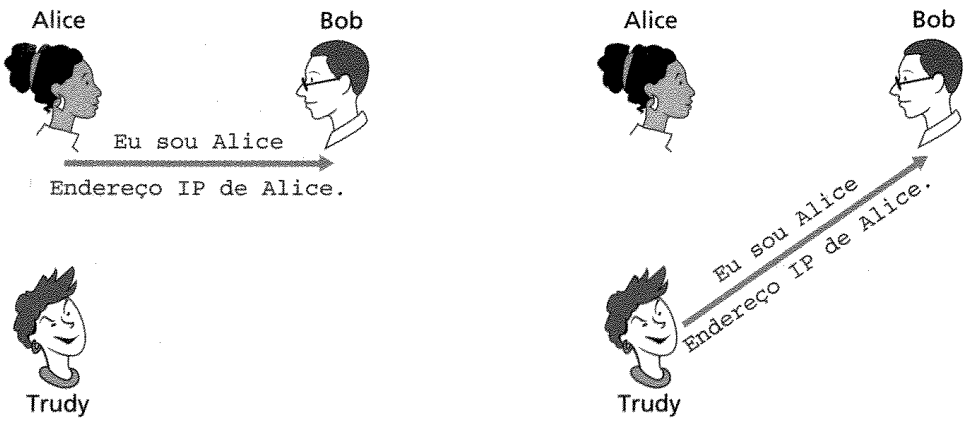


Figura 8.8 Protocolo ap2.0 e um cenário de falha

8.3.3 Protocolo de autenticação ap3.0

Uma abordagem clássica da autenticação é usar uma senha secreta. Temos PINs (Personal Identification Numbers) que nos identificam em caixas automáticos e senhas de login em sistemas operacionais. A senha é um segredo compartilhado pelo autenticador e pela pessoa que está sendo autenticada. Vimos na Seção 2.2 que o HTTP usa um esquema de autenticação baseado em senha. Telnet e FTP também utilizam senha de autenticação. Assim, no protocolo *ap3.0* Alice envia sua senha secreta a Bob, como mostra a Figura 8.9.

Em vista da utilização tão amplamente disseminada de senhas, seria normal que achássemos que o protocolo *ap3.0* é razoavelmente seguro. E estaríamos errados! A falha de segurança, nesse caso, é visível. Se Trudy monitorar uma comunicação de Alice, ela poderá descobrir a senha de Alice. Para que você não pense que isso é improvável, considere o fato de que, quando se inicia uma sessão Telnet com outro computador e se faz o login, a senha de login não é criptografada para ser enviada ao servidor Telnet. Alguém conectado ao Telnet cliente ou à LAN do servidor poderia procurar (ler e armazenar) todos os pacotes transmitidos pela LAN e, dessa maneira, se apossar da senha de login. De fato, essa é uma abordagem muito conhecida para roubar senhas — consulte, por exemplo, [Jimenez, 1997]. Essa ameaça, obviamente, é muito real; portanto, o *ap3.0* com certeza não é apropriado.

8.3.4 Protocolo de autenticação ap3.1

Nossa próxima idéia para ajustar o *ap3.0* é, naturalmente, criptografar a senha. Criptografando a senha de Alice, podemos impedir que Trudy descubra aquela senha. Admitindo que Alice e Bob compartilhem uma chave simétrica secreta, K_{A-B}, então Alice pode criptografar a senha e enviar sua mensagem de identificação, "`Eu sou Alice`", e sua senha criptografada a Bob. Bob então decripta a senha e, se a senha estiver correta, autentica Alice. Ele fica tranqüilo ao autenticar Alice, já que ela conhece não somente a senha, mas também o número da chave secreta compartilhada necessário para criptografar a senha. Vamos denominar esse protocolo *ap3.1*.

Embora seja verdade que o *ap3.1* impede que Trudy descubra a senha de Alice, o uso de criptografia nesse caso não resolve o problema da autenticação. Bob está sujeito a um **ataque de reprodução**: basta Trudy monitorar uma comunicação de Alice, gravar a versão criptografada da senha e, mais tarde, reproduzir essa versão criptografada para Bob, fingindo que é Alice. O uso de uma senha criptografada no *ap.3.1* não faz com que a situação fique muito diferente daquela da Figura 8.9.

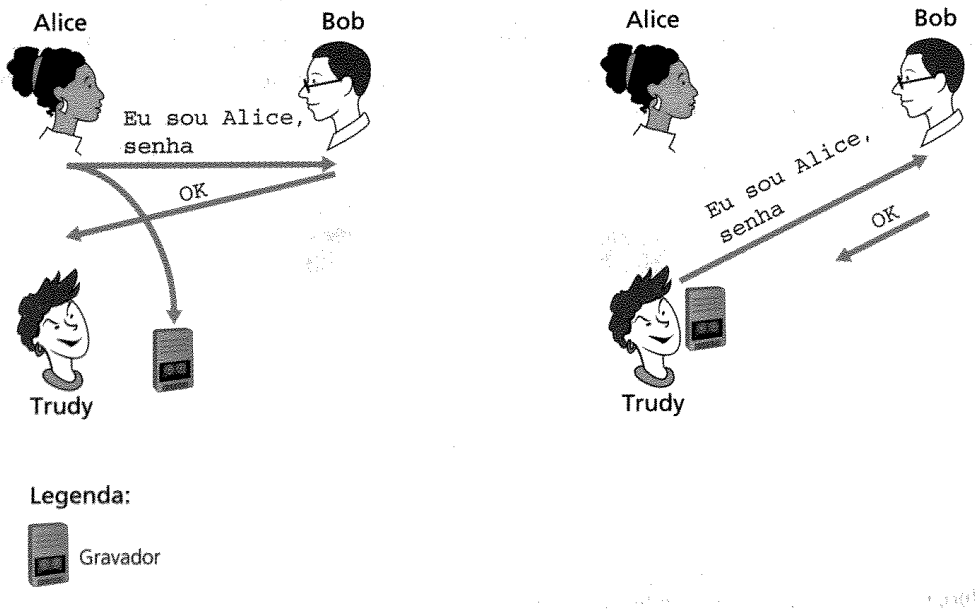


Figura 8.9 Protocolo ap3.0 e um cenário de falha

8.3.5 Protocolo de autenticação ap4.0

O problema com o *ap3.1* é que a mesma senha é usada repetidamente. Um modo de solucionar esse problema seria utilizar cada vez uma senha diferente. Alice e Bob poderiam combinar uma seqüência de senhas (ou um algoritmo para geração de senhas) e usar cada uma delas apenas uma vez, em seqüência. Essa idéia é usada no sistema S/KEY [RFC 1760], que adota um mecanismo lançado por Lamport [Lamport, 1981] para gerar uma seqüência de senhas.

No entanto, em vez de pararmos nesse ponto da solução, vamos considerar uma abordagem mais geral para combater o ataque de reprodução. O cenário de falha da Figura 8.9 resultou do fato de Bob não conseguir distinguir a autenticação original de Alice da reprodução da autenticação original de Alice. Isto é, Bob não conseguiu saber se Alice estava ao vivo na conexão (isto é, se naquela ocasião Alice estava realmente na outra extremidade da conexão) ou se as mensagens que ele estava recebendo eram uma reprodução gravada de uma autenticação anterior de Alice. O leitor muito (muito) observador vai lembrar que o protocolo de apresentação de três vias do TCP tinha de enfrentar o mesmo problema — o lado servidor de uma conexão TCP não era obrigado a aceitar uma conexão se o segmento SYN recebido fosse uma cópia antiga (retransmissão) de um segmento SYN de uma conexão anterior. Como o lado servidor do TCP resolvia o problema de determinar se o cliente estava realmente ao vivo? Ele escolhia um número de seqüência inicial que não tinha sido usado por um longo período de tempo, enviava esse número ao cliente e ficava esperando que o cliente respondesse com um segmento ACK contendo esse número. Podemos adotar a mesma idéia aqui, para a finalidade de autenticação.

Um **nonce** é um número que um protocolo usará apenas uma vez. Isto é, assim que um protocolo usar um nonce, nunca mais o utilizará. Nosso protocolo *ap4.0* usa um nonce como segue:

1. Alice envia a mensagem "`Eu sou Alice`" para Bob.
2. Bob escolhe um nonce, R, e o envia a Alice.
3. Alice criptografa o nonce usando a chave simétrica secreta K_{A-B} que combinou com Bob e envia o nonce cifrado $K_{A-B}(R)$ de volta a Bob. Como acontece no protocolo *ap3.1*, é o fato de Alice conhecer a chave K_{A-B} e usá-la para criptografar um valor que permite que Bob saiba que a mensagem que recebeu foi gerada por Alice. O nonce é utilizado para assegurar que Alice está ao vivo.
4. Bob decripta a mensagem recebida. Se o nonce decriptado for igual ao nonce que enviou a Alice, então ela estará autenticada.

O protocolo *ap4.0* está ilustrado na Figura 8.10. Usando um valor R apenas uma única vez e, em seguida, verificando o valor devolvido, $K_{A-B}(R)$, Bob pode ter certeza de que Alice é quem diz ser (já que conhece o valor da chave secreta necessário para criptografar R) e também de que ela está ao vivo (já que ela tem o nonce criptografado R que Bob acabou de criar).

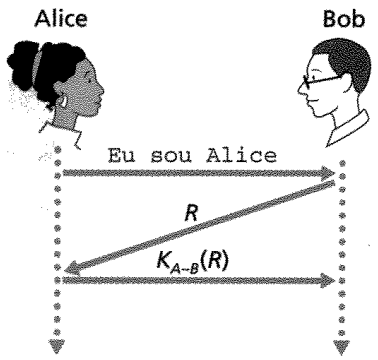


Figura 8.10 Protocolo ap4.0: não há cenário de falha

8.3.6 Protocolo de autenticação ap5.0

A utilização de um nonce e da criptografia de chaves simétricas criou a base do nosso bem-sucedido protocolo de autenticação *ap4.0*. Uma pergunta natural é se podemos usar um nonce e criptografia de chaves públicas (em vez de criptografia de chaves simétricas) para resolver o problema da autenticação. A utilização de uma abordagem de chaves públicas evitaria a dificuldade existente em qualquer sistema compartilhado — a preocupação com o modo pelo qual as duas partes ficarão conhecendo o valor da chave secreta compartilhada, em primeiro lugar. O *ap.5.0* é um protocolo que utiliza criptografia de chaves públicas de um modo análogo à utilização de criptografia de chaves simétricas pelo protocolo *ap4.0*.

1. Alice envia a mensagem "`Eu sou Alice`" para Bob.
2. Bob escolhe um nonce, R, e o envia a Alice. Mais uma vez, o nonce será usado para ele se certificar de que Alice está ao vivo.
3. Alice usa sua chave privada, K_A^-, para criptografar o nonce e envia o valor resultante $K_A^-(R)$ a Bob. Visto que somente Alice conhece sua chave privada, ninguém, exceto ela, pode gerar $K_A^-(R)$.
4. Bob aplica a chave pública de Alice, K_A^+, à mensagem recebida; isto é, processa $K_A^+(K_A^-(R))$. Lembre-se da nossa discussão sobre a criptografia de chave pública RSA, na Seção 8.2, que $K_A^+(K_A^-(R)) = R$. Assim, Bob calcula R e autentica Alice.

O funcionamento do protocolo *ap5.0* é ilustrado na Figura 8.11 O protocolo *ap5.0* é tão seguro quanto o protocolo *ap4.0*? Ambos usam nonces. Como o *ap5.0* usa técnicas de chave pública, ele requer que Bob recupere a chave pública de Alice. Isso leva a um cenário interessante, mostrado na Figura 8.12, em que Trudy pode se passar por Alice perante Bob:

1. Trudy envia a Bob a mensagem "`Eu sou Alice`".
2. Bob escolhe um nonce, R, e o envia a Alice, mas a mensagem é interceptada por Trudy.
3. Trudy usa sua chave privada K_T^-, para criptografar o nonce e envia o valor resultante, $K_T^-(R)$, a Bob. Para Bob, $K_T^-(R)$ é apenas um punhado de bits e ele não sabe se os bits representam $K_T^-(R)$ ou $K_A^-(R)$.
4. Bob deve agora pegar a chave pública de Alice para aplicar K_A^+ ao valor que acabou de receber. Ele envia a mensagem a Alice, solicitando seu K_A^+ (Bob também pode recuperar a chave pública de Alice do site Web dela). Trudy intercepta também essa mensagem e responde a Bob devolvendo K_T^+, isto é, a chave pública dela, Trudy. Bob então calcula $K_T^+(K_T^-(R)) = R$ e, assim, autentica Trudy como se ela fosse Alice!

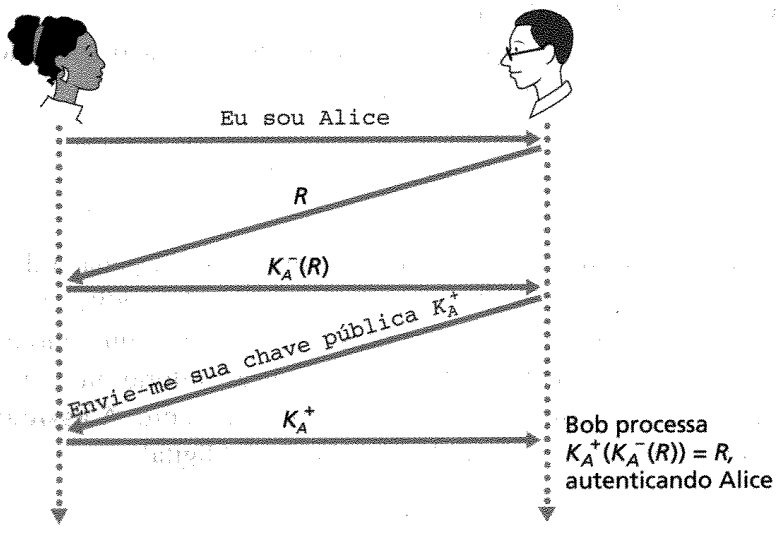


Figura 8.11 Protocolo ap5.0 trabalhando corretamente

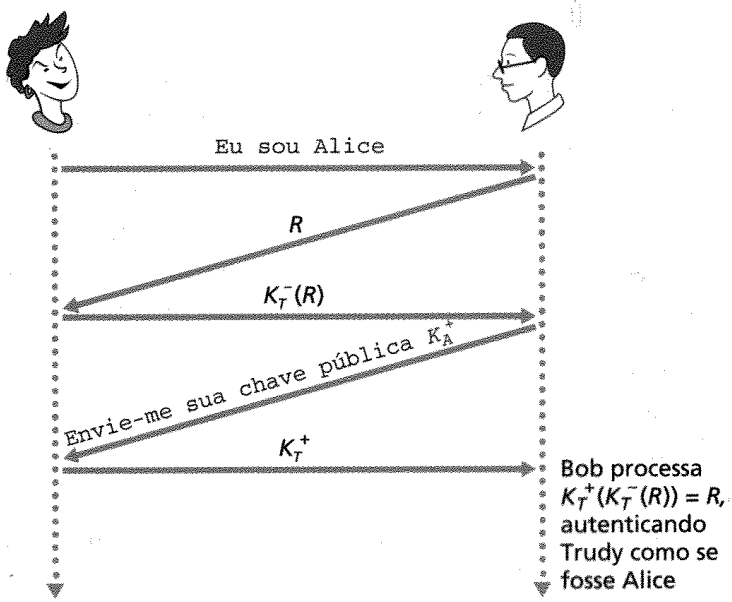


Figura 8.12 Uma falha de segurança no protocolo ap5.0

Fica claro nesse cenário que o protocolo *ap5.0* é apenas tão seguro quanto a distribuição de chaves públicas. Felizmente, existem maneiras seguras de distribuir chaves públicas, como veremos na Seção 8.5.

No cenário da Figura 8.12, Bob e Alice podem acabar descobrindo, juntos, que alguma coisa está errada, pois Bob dirá que interagiu com Alice, mas Alice sabe que isso não aconteceu. Há um ataque ainda mais insidioso que poderia impedir essa descoberta. No cenário da Figura 8.13, Alice e Bob estão conversando, mas Trudy, explorando a mesma falha no protocolo de autenticação, consegue se interpor de maneira transparente entre Alice e Bob. Em particular, se Bob começar a enviar dados cifrados a Alice usando a chave criptográfica que recebeu de Trudy, Trudy poderá recuperar o texto aberto da comunicação entre Bob e Alice. Ao mesmo tempo, Trudy pode repassar os dados de Bob a Alice (após ter decriptado os dados usando a chave pública real de Alice).

Bob está lá, feliz da vida, enviando dados criptografados. Alice também está lá, feliz da vida, recebendo dados criptografados e usando para isso sua própria chave pública; ambos não sabem da presença de Trudy. Se acontecer de Bob e Alice se encontrarem mais tarde e discutirem sua interação, Alice terá recebido exatamente o que Bob enviou, de modo que ninguém perceberá que alguma coisa está errada. Esse é um exemplo do denominado **ataque do homem do meio** (no caso, da 'mulher do meio'). Às vezes ele também é denominado **ataque da brigada do balde**, já que o modo como Trudy está passando os dados entre Alice e Bob parece a passagem dos baldes de água pela longa fila de pessoas (a 'brigada do balde') que estão apagando um incêndio utilizando uma fonte de água longínqua.

8.4 Integridade

Pense no número de vezes em que você assinou seu nome em um pedaço de papel durante a última semana. Você assina cheques, comprovantes de operação de cartões de crédito, documentos legais e cartas. Sua assinatura atesta o fato de que você (e não outra pessoa) conhece o conteúdo do documento e/ou concorda com ele. No mundo digital, freqüentemente se quer indicar o dono ou o criador de um documento ou deixar claro que alguém concorda com o conteúdo de um documento. A **assinatura digital** é uma técnica criptográfica usada para cumprir essas finalidades no mundo digital.

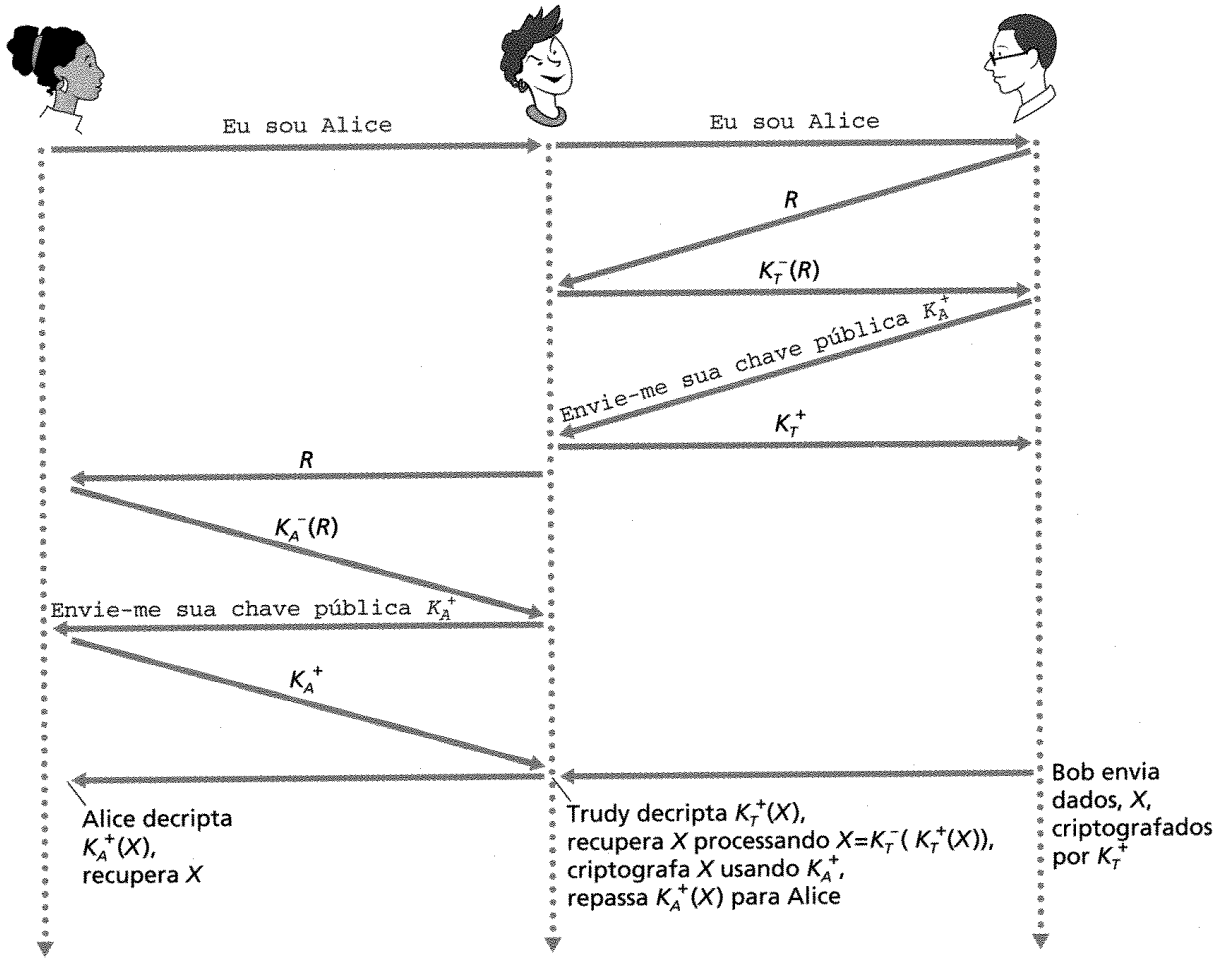


Figura 8.13 Um ataque do homem do meio

Exatamente como acontece com as assinaturas por escrito, a assinatura digital deve ser verificável, não falsificável e incontestável. Isto é, deve ser possível provar que um documento assinado por um indivíduo foi na verdade assinado por ele (a assinatura tem de ser verificável) e que somente aquele indivíduo poderia ter assinado o documento (a assinatura não pode ser falsificada, e o signatário não pode mais tarde repudiar o documento, nem negar que o assinou). Isso se consegue facilmente usando técnicas de criptografia de chaves públicas.

8.4.1 Geração de assinaturas digitais

Suponha que Bob queira assinar digitalmente um documento, m. Imagine que o documento seja um arquivo ou uma mensagem que Bob vai assinar e enviar. Como mostra a Figura 8.14, para assinar esse documento Bob simplesmente usa sua chave criptográfica privada K_B^- para processar $K_B^-(m)$. A princípio, pode parecer estranho que Bob esteja usando sua chave privada (que, como vimos na Seção 8.2, foi usada para decriptar uma mensagem que tinha sido criptografada com sua chave pública) para assinar um documento. Mas lembre-se de que criptografia e decriptação nada mais são do que uma operação matemática (exponenciação à potência e ou d no RSA; veja a Seção 8.2) e que a intenção de Bob não é embaralhar ou disfarçar o conteúdo do documento, mas assinar o documento de maneira que este seja verificável, não falsificável e incontestável. Bob tem o documento, m, e sua assinatura digital do documento é $K_B^-(m)$.

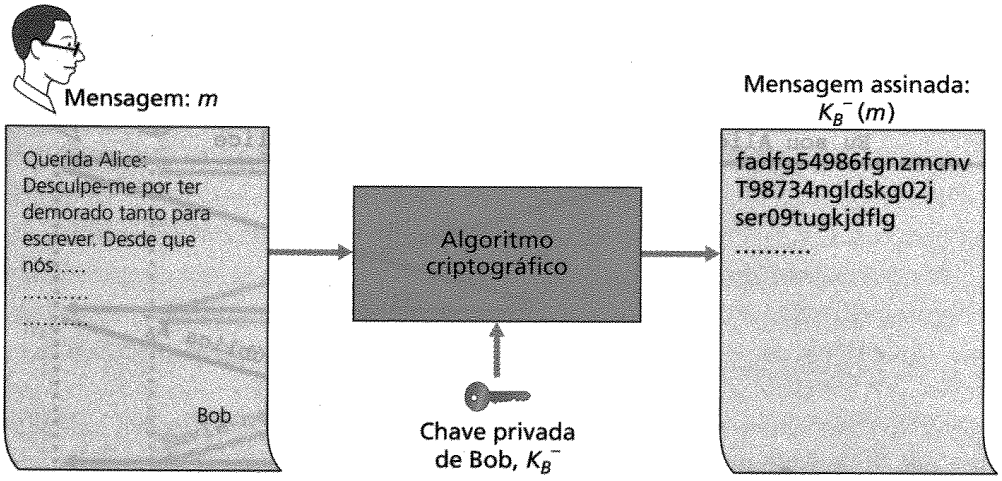


Figura 8.14 Criação de uma assinatura digital para um documento

A assinatura digital $K_B^-(m)$ atende às nossas exigências de ser verificável, não falsificável e não repudiável? Suponha que Alice tenha m e $K_B^-(m)$. Ela quer provar na Justiça (em ação litigiosa) que Bob de fato assinou o documento e que ele era a única pessoa que poderia tê-lo assinado. Alice pega a chave pública de Bob, $K_B^+(m)$, e a aplica à assinatura digital $K_B^-(m)$ associada ao documento m. Isto é, ela processa $K_B^+(K_B^-(m))$, e, voilà, com dramática encenação, produz m, que é uma reprodução exata do documento original! Ela então argumenta que somente Bob poderia ter assinado o documento pelas seguintes razões:

- Quem quer que tenha assinado o documento deve ter usado a chave criptográfica privada, K_B^-, para processar a assinatura $K_B^-(m)$, de modo que $K_B^+(K_B^-(m)) = m$.
- A única pessoa que poderia conhecer a chave privada K_B^-, é Bob. Lembre-se de que dissemos em nossa discussão do RSA, na Seção 8.2, que conhecer a chave pública, K_B^+, não serve para descobrir a chave privada, K_B^-. Portanto, a única pessoa que poderia conhecer K_B^- é aquela que gerou o par de chaves, (K_B^+, K_B^-), em primeiro lugar, Bob. (Note que, para isso, admitimos que Bob não passou K_B^- a ninguém e que ninguém roubou K_B^- de Bob.)

Também é importante notar que, se o documento original, m, for modificado para algum modelo alternativo m', a assinatura que Bob criou para m não será válida para m', já que $K_B^+(K_B^-(m))$ não é igual a m'.

Assim, vemos que as técnicas da criptografia de chaves públicas fornecem uma maneira simples e elegante de assinar documentos digitalmente, uma maneira que é verificável, não falsificável e não repudiável e que protege contra modificações posteriores do documento.

8.4.2 Resumos de mensagem

Vimos na subseção anterior que a tecnologia de chaves públicas pode ser usada para criar uma assinatura digital. Uma preocupação quanto à assinatura de dados por criptografia é que criptografar e decriptar são processamentos dispendiosos em termos computacionais. No caso da assinatura digital de um documento realmente importante, digamos, uma fusão entre duas grandes empresas multinacionais ou um acordo com uma criança para que ela limpe seu quarto uma vez por semana, o custo de processamento pode não ser importante. No entanto, muitos dispositivos e processos de rede (por exemplo, roteadores trocando informações de roteamento e agentes de usuários trocando correspondência por e-mail) trocam dados rotineiramente que talvez não precisassem ser criptografados. No entanto, eles querem garantir que:

- O remetente dos dados é quem diz ser, ou seja, que o remetente assinou os dados e que essa assinatura possa ser verificada.

- Os dados transmitidos não foram modificados desde o momento em que o remetente os criou e assinou.

Dada a sobrecarga de criptografia e decriptação, a assinatura de dados por criptografia/decriptação da mensagem completa pode ser exagerada. Uma abordagem mais eficiente, usando os denominados resumos de mensagem, pode atingir esses dois objetivos sem codificar completamente a mensagem.

Um **resumo de mensagem** é muito parecido com uma soma de verificação. Algoritmos de resumo de mensagem pegam uma mensagem m, de comprimento arbitrário, e calculam uma 'impressão digital' dos dados, com comprimento fixo, conhecida como resumo de mensagem $H(m)$. O resumo de mensagem protege os dados, uma vez que, se m for modificado para m' (seja por má intenção ou por acidente), então a mensagem $H(m)$ processada para os dados originais (e transmitida com os dados) não combinará com $H(m')$ processada sobre os dados modificados, m'. Conquanto o resumo de mensagem proporcione integridade aos dados, que auxílio presta na assinatura da mensagem m? O objetivo aqui é que, em vez de Bob assinar digitalmente a mensagem inteira processando $K_B^-(m)$, ele possa assinar apenas o resumo de mensagem processando $K_B^-(H(m))$. Isto é, ter m e $K_B^-(H(m'))$ juntos (note que m não é criptografada) deve ser tão bom quanto ter uma mensagem completa assinada $K_B^-(m)$. Isso significa que m e $K_B^-(H(m))$, juntos, devem ser não falsificáveis, verificáveis e não repudiáveis. A não-falsificabilidade exigirá que o algoritmo de resumo de mensagem que calcula o resumo de mensagem tenha algumas propriedades especiais, como veremos a seguir.

Nossa definição de resumo de mensagem pode parecer muito semelhante à definição de uma soma de verificação (por exemplo, a soma de verificação da Internet; veja a Seção 3.3.2) ou de um código de detecção de erros mais poderoso, como a verificação de redundância cíclica (veja a Seção 5.2). Ele é realmente diferente? Somas de verificação, verificações de redundância cíclica e resumos de mensagem são exemplos das denominadas **funções de hash**. Como mostra a Figura 8.15, uma função de hash pega uma entrada de dados, m, e processa uma cadeia de tamanho fixo conhecida como hash. A soma de verificação da Internet, as CRCs e os resumos de mensagem estão de acordo com essa definição. Se assinar um resumo de mensagem é tão bom quanto assinar a mensagem inteira, em particular se isso satisfará à exigência de não haver possibilidade de falsificação, então o algoritmo de resumo de mensagem deve ter a seguinte propriedade adicional:

- É inviável, em termos computacionais, achar quaisquer duas mensagens diferentes x e y tais que $H(x) = H(y)$.

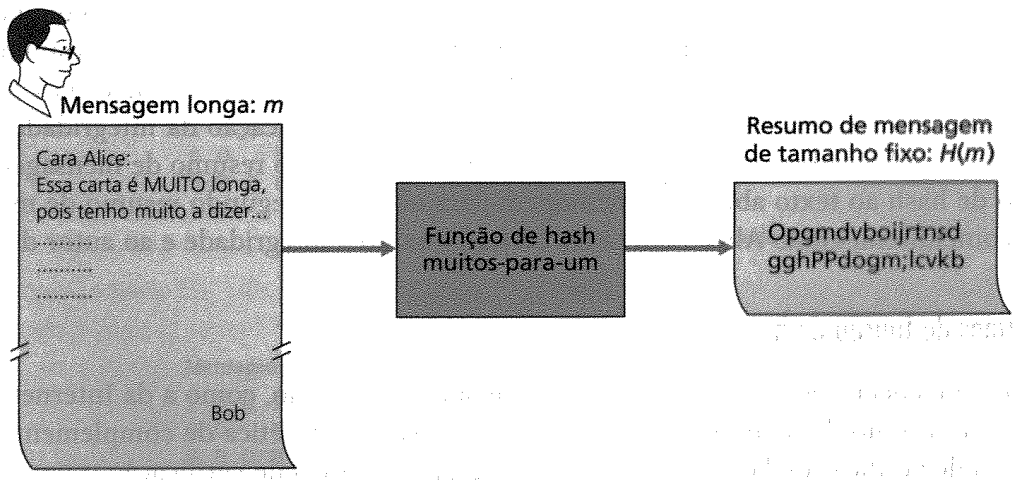


Figura 8.15 Funções de hash são usadas para criar resumos de mensagem

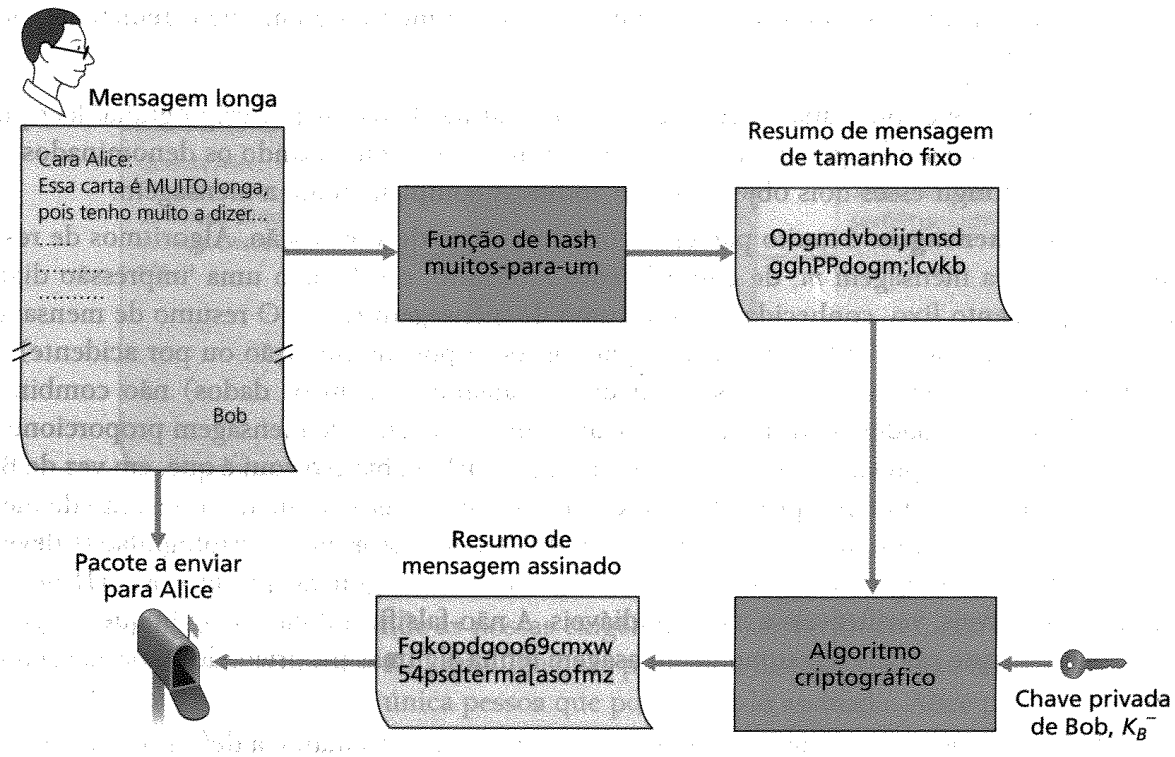


Figura 8.16 Envio de uma mensagem assinada digitalmente

Informalmente, essa propriedade significa que, em termos de processamento, é impraticável que um intruso substitua uma mensagem por outra se a primeira estiver protegida por um resumo de mensagem. Isto é, se $(m,H(m))$ for o par mensagem e resumo de mensagem criado pelo remetente, então um intruso não poderá falsificar o conteúdo de uma outra mensagem, y, que tenha o mesmo valor de resumo de mensagem da mensagem original. Quando Bob assina m, processando $K_B^-(H(m))$, sabemos que nenhuma outra mensagem pode substituir m. Além disso, a assinatura digital de Bob, $H(m)$, identifica exclusivamente Bob como o signatário verificável e incontestável de $H(m)$ (e, em conseqüência, também de m), como discutimos na Seção 8.4.1.

No contexto em que Bob envia uma mensagem a Alice, a Figura 8.16 apresenta um resumo do procedimento operacional para a criação de uma assinatura digital. Bob processa sua mensagem longa original por meio de uma função de hash para criar um resumo de mensagem. Então, ele assina digitalmente o resumo de mensagem com sua chave privada. A mensagem original (em texto aberto) e o resumo de mensagem assinado digitalmente (daqui em diante denominado assinatura digital) são, então, enviados a Alice. A Figura 8.17 mostra um resumo do procedimento operacional de verificação da integridade da mensagem. Alice aplica a chave pública do remetente à mensagem para recuperar o resumo de mensagem. Ela também aplica a função de hash ao texto aberto da mensagem para obter um segundo resumo de mensagem. Se os dois resumos coincidirem, então Alice poderá ter certeza quanto à integridade e ao autor da mensagem.

8.4.3 Algoritmos de função de hash

É bom nos convencermos de que uma simples soma de verificação, como a da Internet, daria um péssimo algoritmo de resumo de mensagem. Em vez de processar a aritmética de complemento de 1 (como é feito para a soma de verificação da Internet), vamos efetuar uma soma de verificação tratando cada caractere como um byte e somando os bytes usando porções de 4 bytes por vez. Suponha que Bob deva a Alice 100,99 dólares e lhe envie um vale contendo a sentença encadeada 'IOU100.99BOB.' (IOU — I Owe You — Eu lhe devo.) A representação ASCII (em notação hexadecimal) para essas letras é 49, 4F, 55, 31, 30, 30, 2E, 39, 39, 42, 4F, 42.

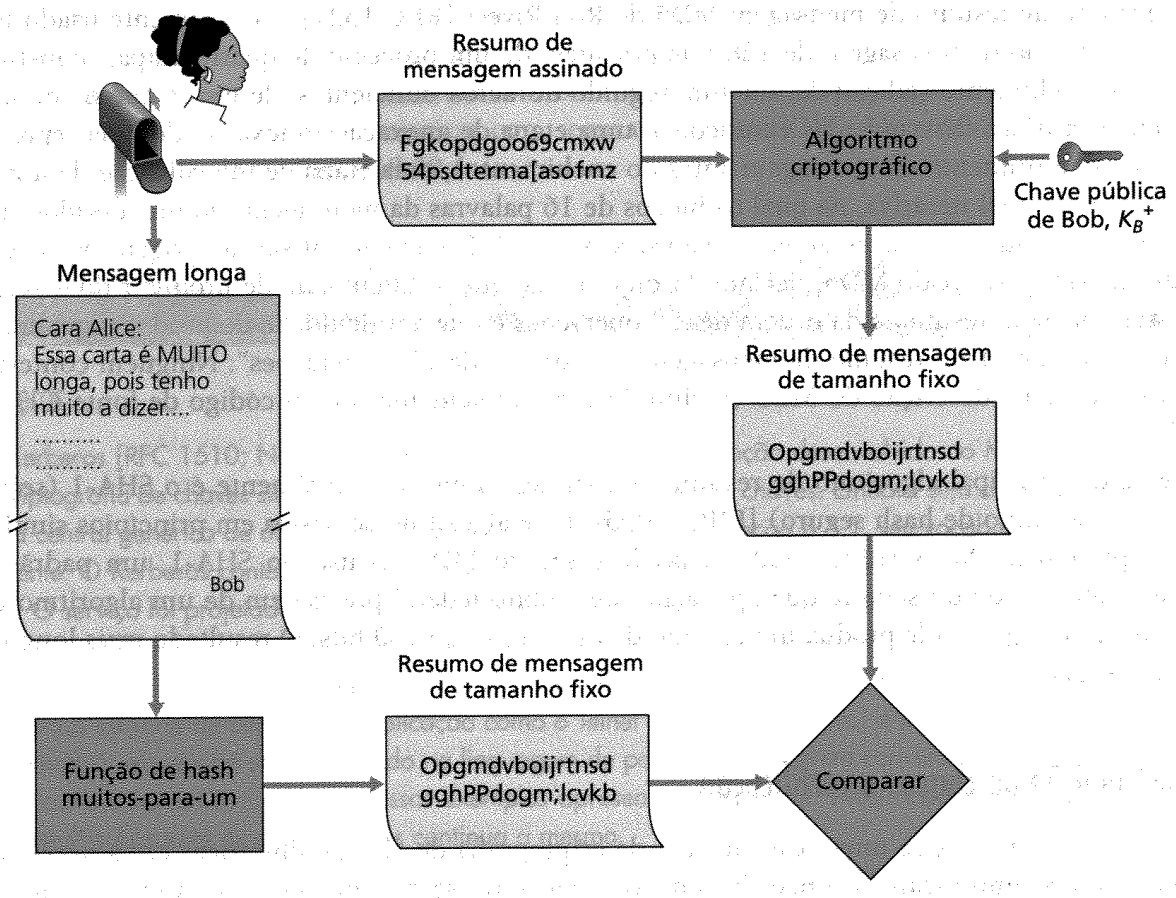


Figura 8.17 Verificação da integridade de uma mensagem assinada

A Figura 8.18 (parte de cima) mostra que a soma de verificação de 4 bytes para essa mensagem é B2 C1 D2 AC. Uma mensagem ligeiramente diferente (e que sairia muito mais cara para Bob) é mostrada na parte de baixo da Figura 8.18. As mensagens 'IOU100.99BOB' e 'IOU900.19BOB' têm a mesma soma de verificação. Assim, esse algoritmo simples viola as duas exigências citadas anteriormente. Fornecidos os dados originais, é simples descobrir outro conjunto de dados com a mesma soma de verificação. É claro que, para efeito de segurança, precisaremos de uma função de hash muito mais poderosa do que uma soma de verificação.

Mensagem	Representação ASCII				
I O U 1	49	4F	55	31	
0 0 . 9	30	30	2E	39	
9 B O B	39	42	4F	42	
	B2	C1	D2	AC	Soma de verificação

Mensagem	Representação ASCII				
I O U 9	49	4F	55	39	
0 0 . 1	30	30	2E	31	
9 B O B	39	42	4F	42	
	B2	C1	D2	AC	Soma de verificação

Figura 8.18 Mensagem inicial e mensagem fraudulenta têm a mesma soma de verificação!

O algoritmo de resumo de mensagem MD5 de Ron Rivest [RFC 1321] é amplamente usado hoje. Ele processa um resumo de mensagem de 128 bits por meio de um processo de quatro etapas, constituído de uma etapa de enchimento (adição de um 'um' seguido de 'zeros' suficientes, de modo que o comprimento da mensagem satisfaça determinadas condições), uma etapa de anexação (anexação de uma representação de 64 bits do comprimento da mensagem antes do enchimento), uma etapa de inicialização de um acumulador e uma etapa final iterativa, na qual os blocos de 16 palavras da mensagem são processados (misturados) em quatro rodadas de processamento. Não se sabe se o MD5 de fato satisfaz às exigências citadas anteriormente. Rivest, o autor do MD5, declara: "Conjetura-se que a dificuldade de produzir duas mensagens que tenham o mesmo resumo é da ordem de 2^{64} operações e que a dificuldade de produzir qualquer mensagem que tenha um dado resumo de mensagem é da ordem de 2^{128} operações". Ninguém contestou essa declaração. Para uma descrição do MD5 (incluindo uma implementação em código de fonte C), consulte [RFC 1321].

O segundo principal algoritmo de resumo de mensagem em uso atualmente é o SHA-1 (*secure hash algorithm* — algoritmo de hash seguro) [FIPS, 1995]. Esse algoritmo se baseia em princípios similares aos usados no projeto do MD4 [RFC 1320], o predecessor do MD5. O uso do SHA-1, um padrão federal norte-americano, é exigido sempre que aplicações de âmbito federal precisarem de um algoritmo de resumo de mensagem seguro. Ele produz um resumo de mensagem de 160 bits. O resultado mais longo torna o SHA-1 mais seguro.

8.5 Distribuição de chaves e certificação

Na Seção 8.2, vimos que uma desvantagem da criptografia de chaves simétricas era a necessidade de que as duas partes comunicantes concordassem com sua chave secreta previamente. Com a criptografia de chaves públicas, esse acordo *a priori* sobre um valor secreto não é necessário. Contudo, como discutimos na Seção 8.2, a criptografia de chaves públicas também tem suas dificuldades, em particular o problema de obter a chave pública verdadeira de alguém. Ambos os problemas — determinação de uma chave compartilhada para a criptografia de chaves simétricas e obtenção de uma chave pública segura, no caso da criptografia de chaves públicas — podem ser solucionados usando-se um **intermediário de confiança**. Para criptografia de chaves simétricas, o intermediário de confiança é denominado **central de distribuição de chaves** (*key distribution center* — KDC), uma entidade de rede única e de confiança com a qual o usuário estabelece uma chave secreta compartilhada. Veremos que a KDC pode ser usada para obter as chaves compartilhadas necessárias para comunicação segura com todas as outras entidades de rede, evitando algumas das armadilhas que identificamos na Seção 8.3. No caso da criptografia de chaves públicas, o intermediário de confiança é denominado **autoridade certificadora** (*certification authority* — CA). Uma CA certifica que uma chave pública pertence a uma determinada entidade (uma pessoa ou uma rede). No caso de uma chave pública certificada, se a confiança depositada na CA que certificou a chave for absoluta, poderemos ter certeza quanto a quem pertence a chave pública. Uma vez certificada, uma chave pública então pode ser distribuída de qualquer lugar, incluindo um servidor de chave pública, uma página Web pessoal ou um disquete.

8.5.1 A central de distribuição de chaves

Suponha, mais uma vez, que Bob e Alice queiram se comunicar usando criptografia de chaves simétricas. Eles nunca se encontraram e, assim, não combinaram previamente uma chave secreta compartilhada. Como é possível, então, que eles agora combinem uma chave secreta, dado que só podem se comunicar um com o outro pela rede? Uma solução freqüentemente adotada na prática é usar uma KDC de confiança.

A KDC é um servidor que compartilha uma chave simétrica secreta única com cada um de seus usuários registrados. Essa chave poderá ser instalada manualmente no servidor quando o usuário se registrar pela primeira vez. A KDC conhece a chave secreta de cada usuário, e cada um deles pode se comunicar com

segurança com a KDC usando essa chave. Vejamos como conhecer essa chave única permite que um usuário obtenha com segurança uma chave para se comunicar com qualquer outro usuário registrado. Suponha que Alice e Bob sejam usuários da KDC; eles conhecem apenas suas próprias chaves individuais, K_{A-KDC} e K_{B-KDC}, respectivamente, para se comunicar com segurança com a KDC. Alice dá o primeiro passo, e eles continuam como ilustrado na Figura 8.19.

Princípios na prática

Kerberos

Kerberos [RFC 1510; Neuman, 1994] é um serviço de autenticação desenvolvido no MIT que usa técnicas de criptografia de chaves simétricas e uma central de distribuição de chaves. Embora seja conceitualmente idêntico à central genérica de distribuição de chaves [KDC] descrita na Seção 8.5.1, seu vocabulário é ligeiramente diferente. O Kerberos contém também diversas variações e extensões interessantes dos mecanismos básicos da KDC. O serviço foi projetado para autenticar usuários que acessam servidores de rede e era inicialmente dirigido para a utilização dentro de um único domínio administrativo, como um campus ou uma empresa. Assim, o Kerberos é estruturado na linguagem de usuários que querem acessar serviços da rede (servidores) utilizando programas de rede de camada de aplicação como o Telnet (para login remoto) e o NFS (para acesso a arquivos remotos), em vez de ser estruturado na linguagem de pessoas que querem conversar entre si e precisam se autenticar mutuamente, como nos nossos exemplos apresentados até aqui. Não obstante, a chave (trocadilho proposital) subjacente às duas técnicas continua a mesma.

O servidor de autenticação (authentication server — SA) do Kerberos desempenha o papel da KDC. Ele é o repositório não apenas das chaves secretas de todos os usuários (de modo que cada usuário pode se comunicar com o SA com segurança), mas também das informações sobre quais usuários têm privilégios de acesso a quais serviços em quais servidores de rede. Quando Alice quer acessar um serviço em Bob (que agora consideramos um servidor), o protocolo segue fielmente o exemplo dado na Figura 8.19.

1. Alice contata o SA Kerberos, indica que quer usar Bob. Toda a comunicação entre Alice e o SA é criptografada usando uma chave secreta compartilhada entre Alice e o SA. No Kerberos, Alice primeiramente fornece seu nome e senha a seu hospedeiro local. Este e o SA então determinam a chave de sessão secreta única para criptografar a comunicação entre Alice e o SA.
2. O SA autentica Alice, verifica se ela tem privilégios de acesso a Bob, e gera uma chave simétrica de sessão única, *R1*, para a comunicação entre Bob e Alice. O servidor de autenticação (na terminologia do Kerberos servidor bilheteiro) envia a Alice o valor de *R1* e também um bilhete de entrada para os serviços de Bob. O bilhete contém o nome de Alice, a chave da sessão Alice-Bob, *R1*, e o prazo de validade, tudo criptografado usando a chave secreta de Bob (conhecida apenas por Bob e por SA), como ilustrado na Figura 8.19. O bilhete de Alice é válido só até o fim do prazo de validade, e será rejeitado por Bob se apresentado após esse horário. Para o Kerberos V4, o tempo máximo de vida útil de um bilhete é de cerca de 21 horas. No Kerberos V5, o tempo de vida útil deve expirar antes do final do ano 9999, um sério problema para o ano 10000!
3. Alice então envia seu bilhete a Bob. Ela também envia uma marca de tempo criptografada por *R1* que é usada como um nonce. Bob decripta o bilhete usando sua chave secreta, obtém a chave da sessão e decripta a marca de tempo utilizando a chave de sessão que acabou de descobrir. Ele devolve o nonce a Alice, criptografado por *R1*, mostrando, assim, que ele conhece *R1* e que está ao vivo.

A versão mais recente do Kerberos (V5) fornece suporte para vários servidores de autenticação, delegação de direitos de acesso e bilhetes renováveis. [Kaufman, 1995] e [RFC 1510] dão mais detalhes sobre o assunto.

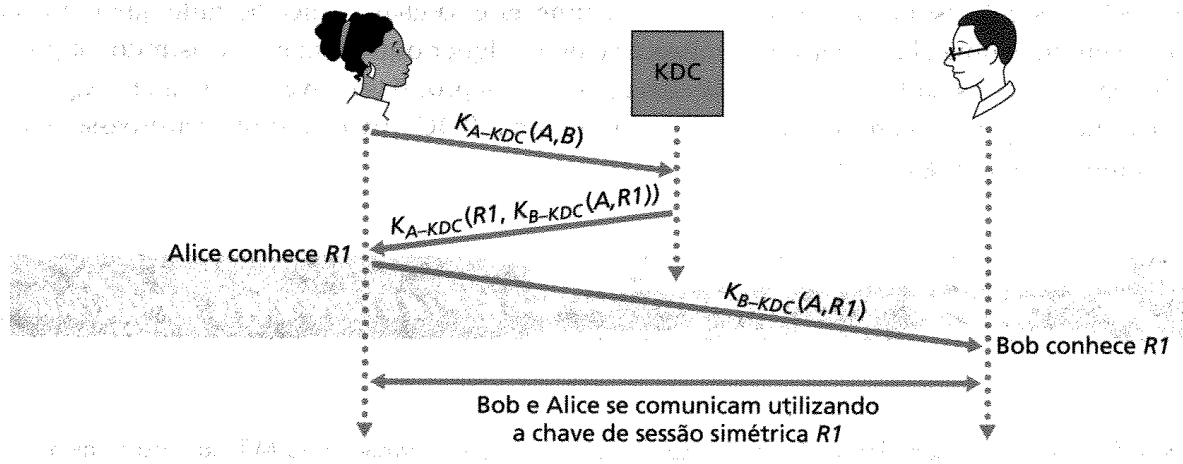


Figura 8.19 Estabelecimento de uma chave de sessão única usando uma central de distribuição de chaves

1. Usando K_{A-KDC} para criptografar sua comunicação com a KDC, Alice envia uma mensagem à KDC dizendo que ela (A) quer se comunicar com Bob (B). Denominamos essa mensagem $K_{A-KDC}(A, B)$.
2. A KDC, como conhece K_{A-KDC}, decripta $K_{A-KDC}(A, B)$. Em seguida, ela gera um número aleatório $R1$. Esse é o valor da chave compartilhada que Alice e Bob usarão para realizar criptografia simétrica quando se comunicarem mutuamente. Essa chave é denominada **chave de sessão única**, pois Alice e Bob a utilizarão apenas durante essa única sessão. A KDC agora precisa informar o valor de $R1$ a Alice e Bob. Assim, ela devolve uma mensagem a Alice, criptografada usando K_{A-KDC}, contendo o seguinte:

- $R1$, a chave de sessão única que Alice e Bob usarão para comunicação.
- Um par de valores: A e $R1$, criptografados pela KDC usando a chave de Bob, K_{B-KDC}. Denominamos esse valor $K_{B-KDC}(A, R1)$. É importante notar que a KDC está enviando à Alice não somente o valor de $R1$ para seu uso, mas também uma versão criptografada de $R1$ e o nome de Alice, criptografado usando a chave de Bob. Alice não pode decriptar esse par de valores na mensagem (ela não conhece a chave criptográfica de Bob), mas, na verdade, ela não precisa fazê-lo. Veremos em breve que Alice simplesmente repassa o par de valores criptografados para Bob, que pode decifrá-los.

A KDC coloca esses itens em uma mensagem, criptografa-os usando a chave compartilhada de Alice e os envia a ela. A mensagem da KDC para Alice é, portanto, $K_{A-KDC}(R1, K_{B-KDC}(R1))$.

3. Alice recebe a mensagem da KDC, decripta a mensagem e dela extrai $R1$. Agora Alice conhece a chave de sessão única, $R1$. Ela também extrai $K_{B-KDC}(A, R1)$ e a repassa a Bob.
4. Bob decripta a mensagem recebida, $K_{B-KDC}(A, R1)$, usando K_{B-KDC} e extrai A e $R1$. Ele agora conhece a chave de sessão única $R1$ e sabe quem é a pessoa com a qual está compartilhando essa chave, A. É claro que ele toma o cuidado de autenticar Alice usando $R1$ antes de prosseguir.

8.5.2 Certificação de chaves públicas

Uma das características principais da criptografia de chaves públicas é que é possível que duas entidades troquem mensagens secretas sem ter de trocar chaves secretas. Por exemplo, quando Alice quer enviar uma mensagem secreta a Bob, ela simplesmente criptografa a mensagem com a chave pública de Bob e lhe envia a mensagem criptografada; Alice não precisa conhecer a chave privada de Bob nem Bob precisa conhecer a chave privada de Alice. Assim, a criptografia de chaves públicas evita a necessidade de infra-estrutura de KDC.

Evidentemente, mesmo com criptografia de chaves públicas, as entidades comunicantes ainda têm de trocar chaves públicas. Um usuário pode disponibilizar sua chave pública de muitas maneiras, por exemplo, apresentando a chave em sua página Web pessoal, colocando-a em um servidor de chaves públicas ou enviando-a a um correspondente por e-mail. Um site Web de comércio pode colocar sua chave pública em seu servidor de modo que os browsers descarreguem automaticamente a chave pública quando se conectam com o site. Roteadores podem colocar suas chaves públicas em servidores de chaves públicas, permitindo, desse modo, que outras entidades da rede as recuperem.

Há, contudo, um problema sutil, mas crítico, com a criptografia de chaves públicas. Para perceber melhor esse problema, vamos considerar uma transação comercial pela Internet, por exemplo. Suponha que Alice trabalhe no ramo de pizzas para viagem e que aceite pedidos pela Internet. Bob, que adora pizza, envia a Alice uma mensagem em texto aberto que contém o endereço de sua casa e o tipo de pizza que quer. Nessa mensagem, ele inclui também uma assinatura digital (isto é, um resumo de mensagem assinado extraído da mensagem original em texto aberto). Como discutimos na Seção 8.4, Alice pode obter a chave pública de Bob (de sua página Web pessoal, de um servidor de chaves públicas ou de uma mensagem de e-mail) e verificar a assinatura digital. Dessa maneira, ela se certifica de que foi Bob, e não algum adolescente brincalhão, quem fez o pedido.

Tudo parece caminhar bem até que entra em cena a esperta Trudy. Como mostrado na Figura 8.20, Trudy decide fazer uma travessura. Ela envia uma mensagem a Alice na qual diz que é Bob, fornece o endereço de Bob e pede uma pizza. Ela também anexa uma assinatura digital, mas faz isso assinando o resumo de mensagem com sua chave privada. Ela também se faz passar por Bob enviando a Alice sua chave pública, mas dizendo que a chave pertence a Bob. Nesse exemplo, Alice aplicará a chave pública de Trudy (pensando que é a de Bob) à assinatura digital e concluirá que a mensagem em texto aberto foi, na verdade, criada por Bob. Este ficará muito surpreso quando o entregador aparecer em sua casa com uma pizza bem variada!

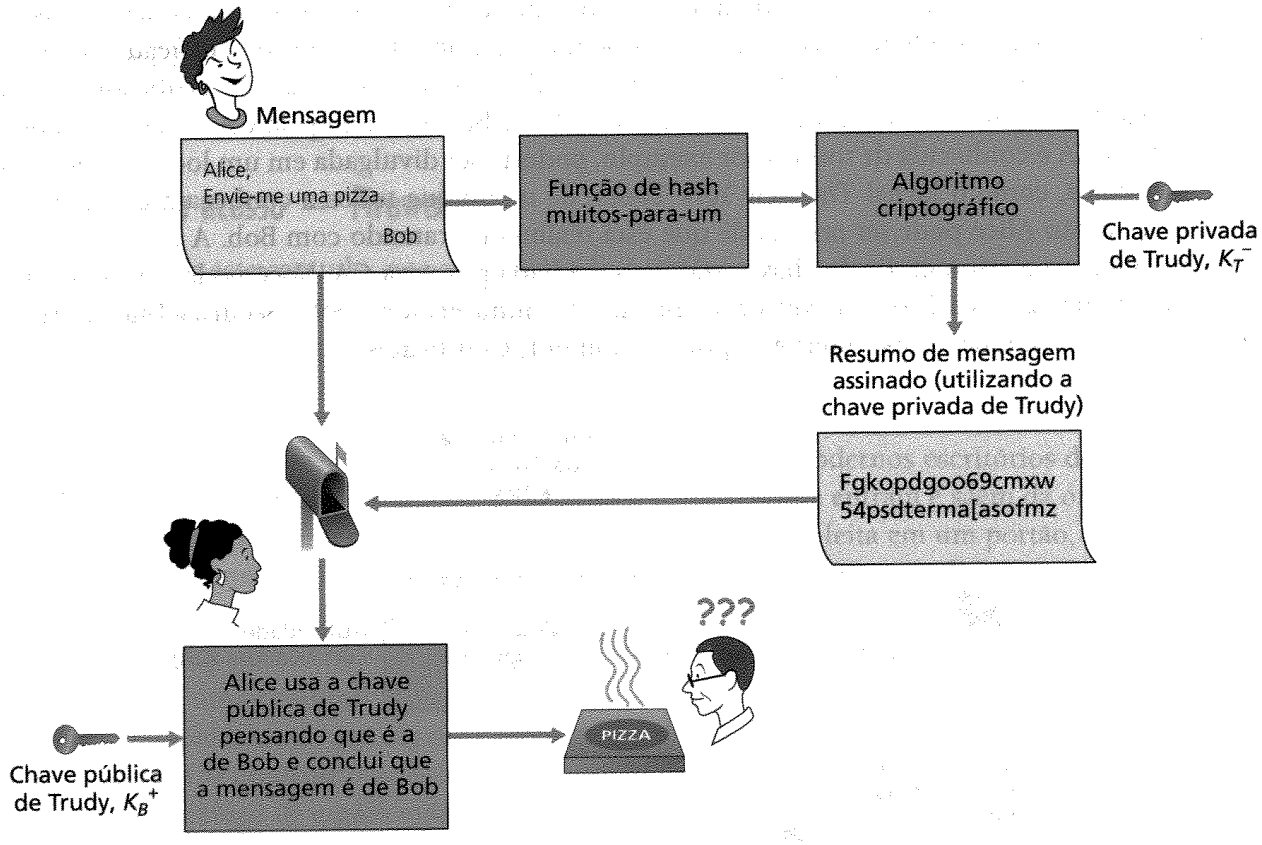


Figura 8.20 Trudy se passa por Bob usando criptografia de chaves públicas

Por esse exemplo, vemos que, para que a criptografia de chaves públicas seja útil, as entidades (usuários, browsers, roteadores e outras) precisam ter certeza de que possuem a chave pública da entidade com a qual estão se comunicando. Por exemplo, quando Alice estiver se comunicando com Bob usando criptografia de chaves públicas, ela precisará saber, com certeza, que a chave pública que supostamente é de Bob, é de fato dele. Tínhamos preocupações semelhantes em nossos protocolos de autenticação apresentados nas figuras 8.12 e 8.13.

A vinculação de uma chave pública a uma entidade particular é feita, tipicamente, por uma autoridade certificadora (*certification authority* — CA), cuja tarefa é validar identidades e emitir certificados. Uma CA tem as seguintes incumbências:

1. Uma CA verifica se uma entidade (pessoa, roteador e assim por diante) é quem diz ser. Não há procedimentos obrigatórios para o modo como deve ser feita a certificação. Quando tratamos com uma CA, devemos confiar que ela tenha realizado uma verificação de identidade adequadamente rigorosa. Por exemplo, se Trudy conseguisse entrar na autoridade certificadora Fly-by-Night, e simplesmente declarasse "Eu sou Alice" e recebesse certificados associados à identidade de Alice, então não se deveria dar muita credibilidade a chaves públicas certificadas pela autoridade certificadora Fly-by-Night. Por outro lado, seria mais sensato (ou não!) estar mais inclinado a confiar em uma CA que faz parte de um programa federal — ou estadual — como o programa do Estado de Utah que licencia CAs nesse Estado [Utah, 1999]. O grau de confiança que se tem na identidade associada a uma chave pública equivale apenas ao grau de confiança depositada na CA e em suas técnicas de verificação de identidades. Que rede emaranhada de confiança estamos tecendo!
2. Tão logo verifique a identidade da entidade, a CA cria um **certificado** que vincula a chave pública da entidade à identidade verificada. O certificado contém a chave pública e a informação exclusiva que identifica mundialmente o proprietário da chave pública (por exemplo, o nome de alguém ou um endereço IP). Essas etapas são mostradas na Figura 8.21.

Vejamos, agora, como certificados podem ser usados para combater os espertinhos das pizzas, como Trudy e outros indesejáveis. Quando Alice recebe o pedido de Bob, ela obtém o certificado de Bob, que pode estar na página Web dele, em uma mensagem de e-mail ou em um servidor de certificados. Alice usa a chave pública da CA para verificar a validade do certificado de Bob assinado pela CA. Se admitirmos que a chave da própria CA é conhecida de todos (por exemplo, poderia ser divulgada em um local público, conhecido e de confiança, como o jornal *The New York Times*, de modo que todos a conheçam e que não possa ser falsificada), então Alice pode ter certeza de que está realmente tratando com Bob. A Figura 8.21 ilustra as etapas envolvidas na criptografia de chaves públicas mediada por uma CA. Você pode ver os certificados de CAs armazenados em seu browser Netscape clicando Communicator, Tools, Security Info, Certificates; no Internet Explorer, clique Tools, Internet Options, Content, Certificates.

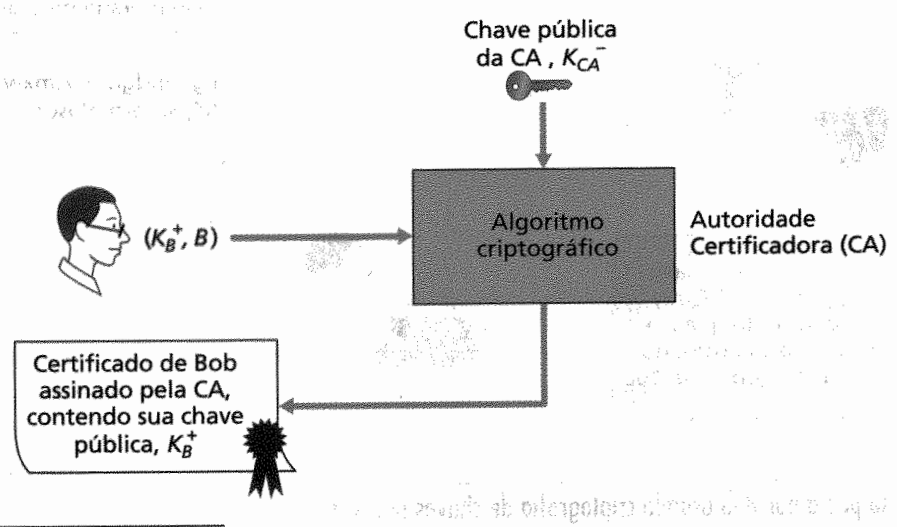


Figura 8.21 Bob obtém um certificado da CA

Nome do campo	Descrição
Versão	Número da versão da especificação X.509
Número de série	Identificador exclusivo emitido pela CA para um certificado
Assinatura	Especifica o algoritmo usado pela CA para assinar esse certificado
Nome do emissor	Identidade da CA que emitiu o certificado, em formato de nome distinto (DN) [RFC 2253]
Período de validade	Início e fim do período de validade do certificado
Nome do sujeito	Identidade da entidade cuja chave pública está associada a esse certificado, em formato DN
Chave pública do sujeito	A chave pública do sujeito, bem como uma indicação do algoritmo de chave pública (e parâmetros do algoritmo) a ser usado com essa chave

Tabela 8.3 Campos selecionados de uma chave pública X.509 e RFC 1422

Tanto a International Telecommunication Union (ITU) como a IETF desenvolveram padrões para autoridades certificadoras. Na recomendação ITU X.509 [ITU, 1993], encontramos a especificação de um serviço de autenticação, bem como uma sintaxe específica para certificados.

O RFC 1422 descreve um gerenciamento de chaves baseado em CA para utilização com o e-mail seguro pela Internet. Essa recomendação é compatível com X.509, mas vai além desta, pois estabelece procedimentos e convenções para uma arquitetura de gerenciamento de chaves. A Tabela 8.3 apresenta alguns campos importantes de um certificado.

Com o recente crescimento do comércio eletrônico e a conseqüente necessidade de garantir a segurança das transações, tem havido um interesse crescente em autoridades certificadoras. Entre as empresas que fornecem serviços de CA estão a Digital Signature Trust Company [Digital Signature, 2004] e a Verisign [Verisign, 2004].

A Figura 8.22 mostra um certificado emitido pela Thawte Consulting para a Netfarmers Enterprises, Inc., como visto por meio de um browser Netscape.

8.6 Controle de acesso — Firewalls

Vimos, em todo este capítulo, que a Internet não é um lugar muito seguro — nela, os delinqüentes estão por toda parte, criando todo tipo de destruição. Do ponto de vista de um administrador, o mundo está dividido claramente em dois campos — os bonzinhos (que pertencem à organização que administra a rede e que deveriam poder acessar recursos dentro da rede que ele administra de um modo relativamente livre de restrições) e os bandidos (todo o resto, cujo acesso aos recursos da rede deve ser cuidadosamente inspecionado). Em muitas organizações, que vão desde castelos medievais a modernos escritórios de empresas, há um único ponto de entrada/saída onde ambos, bonzinhos e bandidos que entram e saem da organização, passam por inspeção de segurança. Em castelos medievais, essa inspeção era feita em um portão, na extremidade de uma ponte levadiça; em escritórios empresariais, a inspeção é feita na central de recepção/segurança. Em redes de computadores, quando o tráfego que entra/sai de uma rede passa por inspeção de segurança, é registrado, descartado e/ou transmitido, a entidade que faz isso é um dispositivo conhecido como um firewall.

Um **firewall** é uma combinação de hardware e software que isola a rede interna de uma organização da Internet em geral, permitindo que alguns pacotes passem e bloqueando outros. Um firewall permite que um administrador de rede controle o acesso entre o mundo externo e os recursos da rede que administra gerenciando o fluxo de tráfego de e para esses recursos. A Figura 8.23 mostra um firewall situado exatamente na fronteira entre a rede administrada e o restante da Internet. Embora organizações de grande porte possam utilizar vários níveis de firewalls ou firewalls distribuídos [Ioannidis, 2000], localizar um firewall em um único ponto de acesso à rede, como mostra a Figura 8.23, facilita a administração e a imposição de uma política de segurança de acesso.

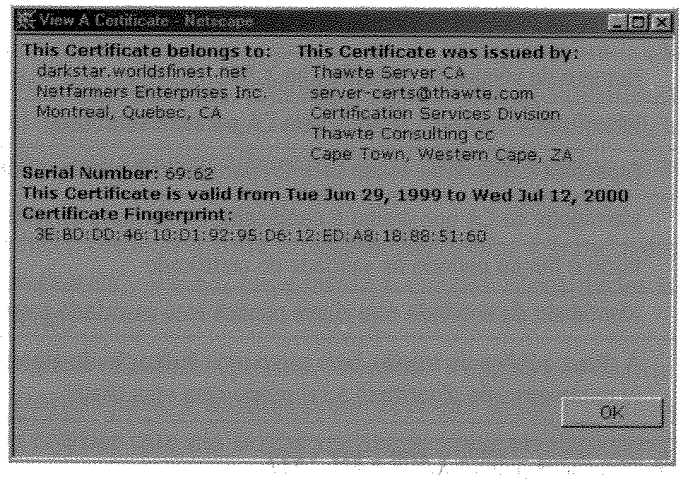


Figura 8.22 Um certificado emitido pela Thawte Consulting à Netfarmers Enterprises, Inc.

Há dois tipos de firewall: **firewalls de filtragem de pacotes** (que funcionam na camada de rede) e **gateways de camada de aplicação** (que funcionam na camada de aplicação). Estudaremos cada um deles nas duas próximas subseções.

8.6.1 Filtragem de pacotes

Como mostra a Figura 8.23, uma organização normalmente tem um roteador de borda que conecta sua rede interna com seu ISP (e dali com a Internet pública, mais ampla). Todo o tráfego que sai ou que entra na rede interna passa por esse roteador e é nesse roteador que ocorre a **filtragem de pacotes**. Os filtros de pacotes primeiramente analisam os cabeçalhos de datagramas e então aplicam regras de filtragem especificadas por um conjunto definido pelo administrador, as quais determinam se o datagrama será descartado ou passará. As decisões de filtragem são normalmente baseadas em:

- Endereço IP de origem e de destino.
- Porta TCP ou UDP de origem e de destino.
- Tipo de mensagem ICMP.
- Datagramas de inicialização de conexão usando bits TCP SYN ou ACK.

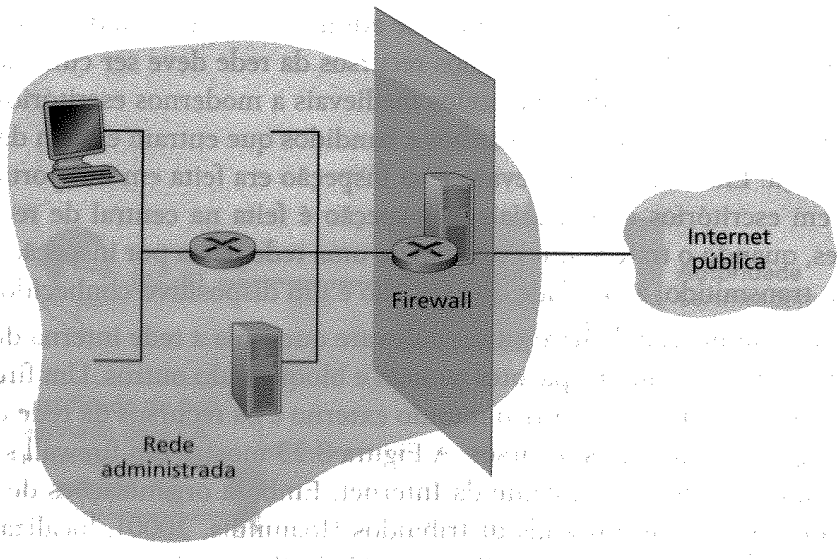


Figura 8.23 Localização de um firewall entre a rede administrada e o mundo exterior

Como exemplo simples, um filtro pode ser ajustado para bloquear todos os segmentos UDP e todas as conexões Telnet. Essa configuração evita que o pessoal externo se conecte com os hospedeiros internos usando Telnet e que o pessoal interno se conecte com hospedeiros externos usando Telnet, bloqueando todos os segmentos TCP (cada um encapsulado em um datagrama) cujo número de porta de origem ou de destino seja 23 (que corresponde ao Telnet). A filtragem do tráfego UDP é uma política popular nas empresas — para desgosto dos fornecedores de áudio e vídeo de fluxo contínuo, cujos produtos entram por UDP no modo default. A filtragem de conexões Telnet também é popular, pois impede que intrusos se conectem aos computadores internos [CERT Filtering, 2002] e provê uma lista de portas e protocolos de filtragem de pacotes recomendados para evitar inúmeros furos de segurança muito conhecidos em aplicações de rede existentes.

Uma política de filtragem também pode ser baseada na combinação de endereços e números de porta. Por exemplo, o roteador de filtragem pode bloquear todos os datagramas Telnet (os que têm número de porta 23), exceto os que estão vindo ou indo de ou para uma lista de endereços IP específicos. Essa política permite conexões Telnet de e para os hospedeiros que estão na lista. Infelizmente, basear a política em endereços externos não oferece nenhuma proteção contra datagramas cujo endereço de fonte pertence a um hospedeiro que está na lista de permitidos, mas que, na verdade, foi enviado por um outro hospedeiro (que muito provavelmente pertence a um bandido que falsificou o endereço de fonte dos datagramas). Examinaremos esse tipo de falsificação de IP, denominada *spoofing*, na próxima seção.

A filtragem pode também ser baseada no bit TCP ACK ter ou não valor. Esse estratagema é bastante útil quando uma organização quer permitir que seus clientes internos se conectem com servidores externos, mas quer impedir que clientes externos se conectem com servidores internos. Lembre-se de que o primeiro segmento de todas as conexões TCP (veja a Seção 3.5) tem o bit ACK com valor 0, ao passo que todos os outros segmentos da conexão têm o bit ACK com valor 1. Assim, se uma organização quiser impedir que clientes externos iniciem uma conexão com servidores internos, ela simplesmente filtrará todos os segmentos que entram que tenham o bit ACK ajustado em 0. Essa política elimina todas as conexões TCP originadas do exterior, mas permite conexões que se originam internamente.

Embora esses exemplos possam dar a impressão de que é razoavelmente fácil especificar regras de filtragem que implementem uma determinada política de segurança, na verdade há muitas sutilezas e armadilhas potenciais envolvidas. Para ilustrar algumas dessas questões, vamos considerar um exemplo simples, adaptado de [Chapman, 1992]. A filtragem de pacotes funciona verificando regras de filtragem seqüencialmente em relação ao datagrama que está sendo inspecionado; a primeira regra que foi verificada pelo datagrama determina a ação a ser executada. Nosso cenário é o seguinte: suponha que Alice administra uma rede empresarial 222.22.0.0/16 e, como regra geral, quer rejeitar acesso à sua rede a partir da Internet pública (regra R3 na Tabela 8.4). Todavia, Alice está colaborando com Bob e seus colegas, que estão em uma universidade, e, portanto, ela quer permitir que usuários da universidade de Bob (cujo endereço de rede é 111.11/16) tenham acesso a uma sub-rede específica, 222.22.22/24, dentro da rede de sua empresa (regra R1). Um fator complicador é que Trudy, uma hacker muito conhecida, está na mesma universidade de Bob e sua sub-rede, 111.11.11/24, é um porto inseguro repleto de hackers. Portanto, Alice não quer que nenhum tráfego que venha de 111.11.11/24 entre em nenhum lugar de sua rede (regra R2). As regras de filtragem de pacotes de Alice estão resumidas na Tabela 8.4.

Regra	Endereço de fonte	Endereço de destino	Ação	Comentários
R1	111.11/16	222.22.22/24	permitir	Permite a entrada de datagramas da rede da universidade de Bob em uma sub-rede restrita.
R2	111.11.11/24	222.22/16	negar	Não permite que tráfego da sub-rede de Trudy entre em nenhum lugar da rede de Alice.
R3	0.0.0.0/0	0.0.0.0/0	negar	Não permite a entrada de tráfego na rede de Alice.

Tabela 8.4 Regras de filtragem de pacotes

Número do Datagrama	Endereço IP de Fonte	Endereço IP de Destino	Ação desejada	Ação sob R2, R1, R3	Ação sob R1, R2, R3
P1	111.11.11.1 (sub-rede de hackers)	222.22.6.6 (rede da empresa)	negar	negar (R2)	negar (R2)
P2	111.11.11.1 (sub-rede de hackers)	222.22.22.2 (sub-rede especial)	negar	negar (R2)	permitir (R1)
P3	111.11.6.6 (rede da universidade, exceto a sub-rede de hackers)	222.22.22.2 (sub-rede especial)	permitir	permitir (R1)	permitir (R1)
P4	111.11.6.6 (rede da universidade, exceto a sub-rede de hackers)	222.22.6.6 (rede da empresa)	negar	negar (R3)	negar (R3)

Tabela 8.5 Resultado de filtragem de pacotes segundo a seqüência das regras

A Tabela 8.5 mostra a manipulação de datagramas selecionados pelo firewall da empresa de Alice quando as regras são aplicadas segundo uma de duas seqüências diferentes. Quando as regras são aplicadas na seqüência que começa pelo endereço de fonte mais específico: R2, R1, R3 (a penúltima coluna na Tabela 8.5), obtemos os resultados desejados: datagramas como P1, cuja fonte está na sub-rede de hackers e cujo destino está dentro da rede da empresa, mas fora da sub-rede especial, são impedidos de passar pelo firewall (pela Regra 2). De modo semelhante, datagramas como P2, que vêm da sub-rede de hackers e se destinam à sub-rede especial, são impedidos de passar pelo firewall (pela Regra 2). Sob a seqüência de regras R2, R1, R3, datagramas que vêm da rede geral da universidade têm permissão de passar para a sub-rede especial (datagrama P3), mas são impedidos de entrar em outras partes da rede da empresa (datagrama P4).

Suponha, entretanto, que as regras são aplicadas na seqüência R1, R2, R3 (a última coluna na Tabela 8.5). Nesse caso, o datagrama P2 obtém, por engano, acesso à sub-rede especial sob a a regra R1, visto que os endereços de fonte e de destino de P2 estão de acordo com a especificação de fonte e de destino segundo a regra R1, e essa regra está sendo aplicada em primeiro lugar. A lição que nos dá esse simples exemplo com três regras é clara — *a seqüência de avaliação de regras pelo firewall é importante*. Fica claro que é preciso pensar nisso com muito cuidado — imagine as dificuldades envolvidas na especificação de firewalls com milhares de regras! Poderíamos nos sentir tentados a pensar que aplicar regras mais específicas em primeiro lugar poderia sempre evitar comportamentos não previstos ou indesejados que surgissem em razão da seqüência de regras. Entretanto, veremos nos problemas ao final deste capítulo que esse não é necessariamente o caso.

8.6.2 Gateway de aplicação

Nos exemplos que acabamos de mostrar, vimos que a filtragem de pacotes permite que uma organização faça uma filtragem grosseira de cabeçalhos IP e TCP/UDP, incluindo endereços IP, números de porta e bits de reconhecimento. Vimos, por exemplo, que a filtragem baseada em uma combinação de endereços IP e números de porta pode permitir que clientes internos executem Telnet para o exterior, ao mesmo tempo que impede que clientes externos executem Telnet para o interior. Mas, e se uma organização quiser fornecer o serviço Telnet a um conjunto restrito de usuários internos (em vez de a endereços IP)? E se a organização quiser que esses usuários privilegiados se autentiquem antes de obter permissão para criar sessões Telnet com o mundo externo? Essas tarefas estão além das capacidades de um filtro. Na verdade, informações sobre a identidade de usuários internos não estão incluídas nos cabeçalhos IP/TCP/UDP; elas estão nos dados da camada de aplicação.

Para assegurar um nível mais refinado de segurança, os firewalls têm de combinar filtro de pacotes com gateways de aplicação. Gateways de aplicação fazem mais do que examinar cabeçalhos IP/TCP/UDP e tomam decisões com base em dados da aplicação. Um **gateway de aplicação** é um servidor específico de

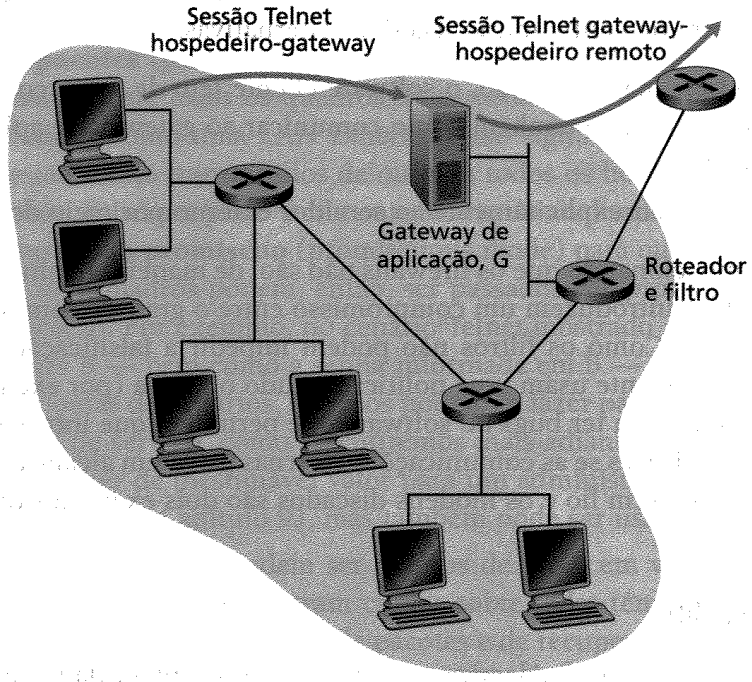


Figura 8.24 Firewall composto de um gateway de aplicação e um filtro

aplicação através do qual todos os dados da aplicação (que entram e que saem) devem passar. Vários gateways de aplicação podem executar no mesmo hospedeiro, mas cada gateway é um servidor separado, com seus próprios processos.

Para termos uma percepção dos gateways de aplicação, vamos projetar um firewall que permite que apenas um conjunto restrito de usuários execute Telnet para o exterior e impede que todos os clientes externos executem Telnet para o interior. Essa política pode ser aplicada pela implementação da combinação de um filtro de pacotes (em um roteador) com um gateway de aplicação de Telnet, como mostra a Figura 8.24. O filtro do roteador está configurado para bloquear todas as conexões Telnet, exceto aquelas que se originam do endereço IP do gateway de aplicação. Essa configuração de filtro força todas as conexões Telnet de saída a passarem pelo gateway de aplicação. Considere agora um usuário interno que quer executar Telnet com o mundo exterior. Em primeiro lugar, ele tem de estabelecer uma sessão Telnet com o gateway de aplicação. Uma aplicação que está executando no gateway — e que fica à escuta de sessões Telnet que entram — solicita ao usuário sua identificação e senha. Quando o usuário fornece essas informações, o gateway de aplicação verifica se ele tem permissão para executar Telnet com o mundo exterior. Se não tiver, a conexão Telnet do usuário interno ao gateway será encerrada pelo gateway. Se o usuário tiver permissão, o gateway (1) pedirá ao usuário o nome do computador externo com o qual ele quer se conectar, (2) estabelecerá uma sessão Telnet entre o gateway e o hospedeiro externo e (3) passará ao hospedeiro externo todos os dados que chegam do usuário e ao usuário todos os dados que chegam do hospedeiro externo. Assim, o gateway de aplicação Telnet não só autoriza o usuário, mas também atua como um servidor Telnet e um cliente Telnet, passando informações entre o usuário e o servidor Telnet remoto. Note que o filtro permitirá a etapa 2, porque é o gateway que inicia a conexão Telnet com o mundo exterior.

Redes internas freqüentemente têm vários gateways de aplicação, como gateways para Telnet, HTTP, FTP e e-mail. De fato, o servidor de correio (veja a Seção 2.4) e o cache Web de uma organização são gateways de aplicação.

Gateways de aplicação não estão isentos de desvantagens. Em primeiro lugar, é preciso um gateway de aplicação diferente para cada aplicação diferente. Em segundo lugar, há um preço a pagar em termos de desempenho, visto que todos os dados serão repassados por meio do gateway. Isso se torna uma preocupa-

ção particularmente quando vários usuários ou aplicações estão utilizando o mesmo gateway. Por fim, é preciso fazer um certo esforço extra de configuração. Há duas alternativas:

- Quando o usuário faz uma solicitação, o software cliente tem de saber como contatar o gateway em vez do servidor externo, e também como comunicar ao gateway de aplicação a qual servidor este deve se conectar, ou
- o usuário deve se conectar explicitamente ao servidor externo por meio do gateway de aplicação.

Concluímos esta seção mencionando que os firewalls não são, de maneira nenhuma, uma panacéia para os problemas de segurança. Eles introduzem um compromisso entre o grau de comunicação com o mundo exterior e o nível de segurança. Como os filtros não podem impedir a falsificação de endereços IP e de números de porta, eles freqüentemente usam uma política de tudo ou nada (por exemplo, banir todo o tráfego UDP). Gateways também podem ter bugs no software que permitem que intrusos penetrem neles. Por fim, firewalls são ainda menos eficazes se as comunicações internas puderem alcançar o mundo externo sem passar por eles. As comunicações sem fio e os modens discados são dois exemplos disso.

8.7 Ataques e contramedidas

Agora que já examinamos criptografia/decriptação, autenticação, integridade de mensagem, distribuição de chaves e firewalls, vamos ver como essas técnicas podem ser utilizadas para combater vários ataques que já foram lançados contra uma rede. Alguns dos ataques mais sensacionais à segurança, como o CodeRed [CERT 2001-19], o vírus Melissa [CERT 1999-04] e o verme Slammer [CERT 2003-04; Moore, 2003] usam a Internet para se propagar, mas atacam sistemas operacionais (via transbordamento de buffer em um servidor Microsoft IIS, no caso do Code Red) ou softwares de aplicação (Microsoft Word, no caso do vírus Melissa) em vez de atacar a rede em si. Como este livro trata de redes, concentraremos nosso foco em ataques que exploram, desabilitam ou envolvem a rede de algum modo especial.

8.7.1 Mapeamento

No mundo real (no sentido de oposto ao mundo cibernético), um ataque é quase sempre precedido de coleta de informações. Em filmes, gângsteres 'vigiam o ponto'; soldados fazem o reconhecimento da área. A finalidade é óbvia — quanto mais soubermos sobre o alvo, menor é a probabilidade de sermos pegos e maior a probabilidade de sucesso. Isso também vale para o mundo cibernético. Antes de atacar uma rede, os invasores gostariam de saber os endereços IP das máquinas pertencentes à rede, quais sistemas operacionais elas utilizam e os serviços que esses sistemas oferecem. Com essas informações, os ataques podem ter um foco mais concentrado e a probabilidade de causar alarme é menor. O processo de coleta de informações é conhecido como **mapeamento**.

Um programa como o *ping* pode ser utilizado para determinar os endereços IP das máquinas presentes na rede simplesmente observando quais endereços respondem à mensagem *ping*. **Varredura de portas** (port scanning) refere-se à técnica de contatar seqüencialmente (seja via uma requisição de conexão TCP, seja via um simples datagrama UDP) números de portas em uma máquina e ver o que acontece em resposta. Essas respostas, por sua vez, podem ser usadas para determinar os serviços (por exemplo, HTTP ou FTP) oferecidos pela máquina. O Nmap [Nmap, 2004] é uma aplicação de código-fonte aberto amplamente utilizada para exploração de redes e para auditoria de desempenho, que executa varredura de portas. Muitos firewalls, como os vendidos pela Checkpoint [Checkpoint, 2004], detectam mapeamento e varredura de portas, bem como outras atividades mal-intencionadas, e as informam ao administrador da rede.

8.7.2 Análise de pacotes

Um **analisador de pacotes** (*packet sniffer*) é um programa que funciona em um dispositivo acoplado à rede e que recebe passivamente *todos* os quadros de camada de enlace que passam por sua interface de rede. Em ambiente *broadcast* como uma LAN Ethernet, isso significa que o analisador de pacotes recebe todos os quadros que estão sendo transmitidos de ou para todos os hospedeiros na LAN. Qualquer hospedeiro que tenha uma placa Ethernet pode facilmente servir de analisador de pacotes, pois basta ajustar o adaptador da Ethernet no **modo promíscuo** (promiscuous mode) para que ele receba todos os quadros que passam pela Ethernet. Esses quadros podem, então, ser passados aos programas de aplicação que extraem dados de camada de aplicação. Por exemplo, no cenário Telnet mostrado na Figura 8.25, o pedido de senha de login enviado de A para B — bem como a senha informada em B — é analisado no hospedeiro C. Após obter senhas de acesso a contas de usuários, intrusos podem se fazer passar pelos donos das contas para lançar um ataque de negação de serviço, como discutiremos mais adiante. Assim, a análise de pacotes é uma faca de dois gumes — pode ser de inestimável valor para um administrador de rede realizar a monitoração e a administração da rede (veja o Capítulo 9), mas também pode ser usada por um hacker inescrupuloso.

Softwares para análise de pacotes podem ser obtidos de graça em vários sites Web ou adquiridos no mercado. Sabe-se que professores que ministram cursos sobre redes têm passado exercícios de laboratório que envolvem escrever um programa de análise de pacotes e de recuperação de dados no nível de aplicação. De fato, os Ethereal Labs [Ethereal, 2004] associados a este texto (veja a apresentação do Ethereal Lab ao final do Capítulo 1) usam exatamente esse tipo de analisador de pacotes!

A chave para detectar análise de pacotes é detectar interfaces de rede que estão configuradas para modo promíscuo. Quando se trata de uma empresa, os administradores de redes podem instalar um software em todos os computadores, que os alertará quando uma interface está configurada para modo promíscuo. Há vários estratagemas que também podem ser executados remotamente para detectar interfaces promíscuas. Por exemplo, um hospedeiro que responda a um datagrama ICMP de solicitação de eco (isto é, que envia uma resposta de eco ICMP; veja a Figura 4.21) que está corretamente endereçado ao datagrama IP, porém contém um endereço MAC incorreto (no nível do quadro), provavelmente tem sua interface configurada em modo promíscuo. A chave para conviver com a análise de pacotes é criptografar todos os dados (em particular senhas) que cruzam um enlace de rede.

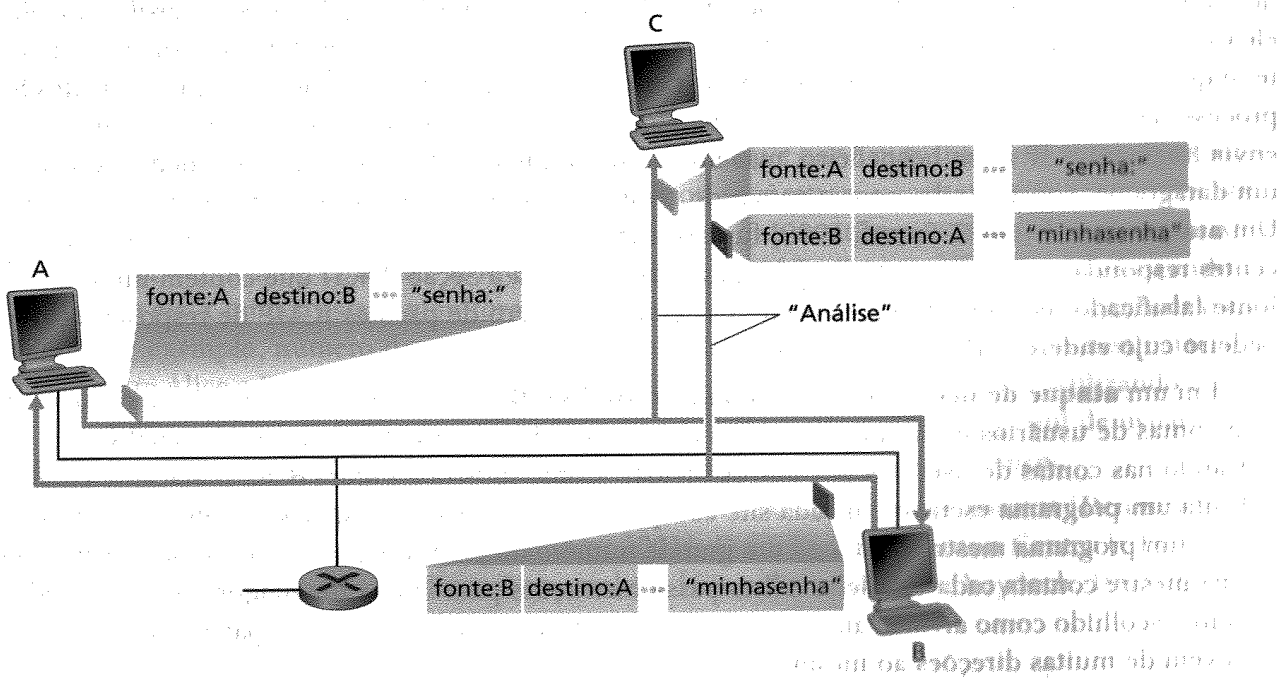


Figura 8.25 Análise de pacotes

8.7.3 Falsificação

Qualquer equipamento conectado à Internet necessariamente envia datagramas IP para a rede. Lembre-se de que esses datagramas portam o endereço IP do remetente, bem como dados da camada superior. Um usuário que tenha completo controle sobre o software do equipamento (em particular sobre o sistema operacional) pode facilmente modificar os protocolos daquele equipamento e colocar um endereço IP arbitrário no campo Endereço de Fonte (Source Address) do datagrama. Isso é conhecido como **falsificação de IP**. Dessa maneira, um usuário pode montar um pacote IP que contenha quaisquer dados de carga útil (camada superior) que quiser e fazer com que pareça que os dados foram enviados de um hospedeiro IP arbitrário. A falsificação de IP é muito usada em ataques de recusa de serviço para ocultar a identidade de quem originou o ataque, como discutiremos a seguir. Se o endereço de fonte IP de um datagrama for falsificado, é difícil descobrir o hospedeiro que o enviou.

Do ponto de vista técnico, é fácil evitar falsificação. Roteadores que executam **filtragem de entrada** verificam os endereços IP de datagramas que estão chegando e determinam se o endereço de fonte está na faixa de endereços de rede que se sabe que podem ser alcançados por meio daquela interface. Essa verificação pode ser realizada com facilidade na borda da rede, por exemplo, em um gateway ou firewall de uma empresa onde é conhecida uma determinada faixa de endereços dentro da qual estão todos os hospedeiros existentes na empresa. A filtragem de ingresso hoje é considerada uma boa prática na Internet [RFC 2827]. Embora seja simples do ponto de vista técnico, a filtragem de ingresso não pode ser imposta e, portanto, de um ponto de vista social, é difícil de implantar — em conseqüência, não é implementada universalmente.

8.7.4 Ataques de negação de serviço e ataques de negação de serviço distribuídos

Há uma categoria muito ampla de ameaças à segurança que pode ser classificada como **ataques de negação de serviço** (*denial-of-service* — DoS). Como o nome sugere, um ataque DoS torna impossível a utilização de uma rede, de um hospedeiro ou de qualquer outro componente da infra-estrutura da rede pelos usuários legítimos. Em geral, um ataque DoS funciona pela criação de uma quantidade tão grande de trabalho para a infra-estrutura sob ataque que o trabalho legítimo não pode ser realizado. Em um **ataque de inundação SYN** [Cert SYN, 1996], o atacante inunda um servidor com pacotes TCP SYN, cada um com um endereço IP de fonte falsificado. O servidor, incapaz de diferenciar um pacote SYN legítimo de um falsificado, conclui a segunda etapa da apresentação TCP (veja a Seção 3.5.6) para um SYN falsificado, alocando a ele estrutura de dados e estado. A terceira etapa da apresentação de três vias nunca é concluída pelo atacante, o que deixa um número cada vez maior de conexões parcialmente abertas. A carga de pacotes SYN a ser processada e a exaustão da memória livre podem derrubar o servidor. Uma forma relacionada de ataque envia fragmentos IP a um hospedeiro, mas nunca um número suficiente desses fragmentos que completem um datagrama, consumindo, com o passar do tempo, uma quantidade cada vez maior de armazenamento. Um **ataque smurf** [CERT Smurf, 1998] funciona fazendo com que um grande número de hospedeiros inocentes responda a pacotes ICMP de solicitação de eco (veja a Seção 4.4.3) que contêm um endereço IP de fonte falsificado. Isso resulta no envio de um grande número de pacotes ICMP de resposta de eco ao hospedeiro cujo endereço IP é objeto da falsificação.

Em um **ataque de negação de serviço distribuído (DDoS)**, em primeiro lugar o atacante obtém acesso a contas de usuários em inúmeros hospedeiros por toda a Internet (por exemplo, analisando senhas ou entrando nas contas de usuários por outros meios). Como mostra a Figura 8.26, o atacante então instala e executa um programa escravo em cada site comprometido que fica lá, esperando, quietinho, pelos comandos de um programa mestre. Tão logo um grande número de programas escravos esteja executando, o programa mestre contata cada um deles e lhes passa instruções para lançar um ataque DoS contra algum hospedeiro escolhido como alvo. O ataque coordenado resultante dessa manobra é particularmente devastador, pois vem de muitas direções ao mesmo tempo.

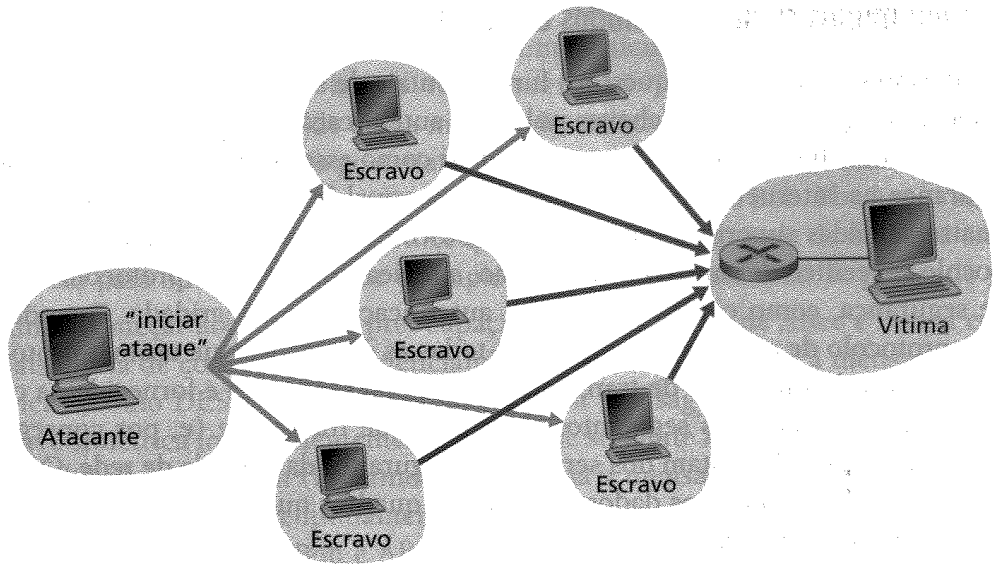


Figura 8.26 Um ataque DDoS

Ataques DoS chegaram às manchetes em fevereiro de 2000 quando os sites da eBay, do Yahoo, da CNN e outros sites de grande porte foram atacados [Cnet, 2000]. É difícil se proteger contra ataques DoS e, mais difícil ainda, contra ataques DDoS. A filtragem de pacotes é difícil porque é difícil distinguir datagramas bons de datagramas ruins. Por exemplo, como um firewall de filtragem de pacotes sabe se um SYN que está chegando é para uma conexão legítima (por exemplo, para um cliente que quer fazer uma compra no site da empresa) ou se é um SYN mal-intencionado que travará os recursos do servidor por não completar a apresentação de três vias subseqüente? A falsificação de IP dificulta a localização da(s) fonte(s) verdadeira(s) do(s) ataque(s). Há várias pesquisas recentes [Savage, 2000; Snoeren, 2001; Adler, 2002] que estudaram técnicas para marcar cabeçalhos IP quando eles passam por um roteador, de modo a poder traçar o caminho percorrido por um fluxo de datagramas DoS até sua fonte. Uma vez identificado um hospedeiro fonte comprometido, ele pode ser posto em quarentena, embora esse processo seja usualmente lento e exija intervenção humana. Resolver ataques DDoS é ainda mais difícil e demorado.

8.7.5 Seqüestro

Suponha que Alice e Bob participem de uma conexão em curso e que Trudy dispõe de meios para monitorar pacotes que fluem entre eles. Trudy pode se apossar, ou seqüestrar, a conexão em curso entre Bob e Alice. Em particular, ela pode enganar Bob e fazê-lo acreditar que continua se comunicando com Alice, embora esteja se comunicando com ela, Trudy. Em primeiro lugar, Trudy tira Alice de cena lançando um ataque DoS contra ela. Como já estava monitorando a comunicação entre Bob e Alice, Trudy conhece o estado completo (por exemplo, número de seqüência, número de ACK, janela anunciada pelo destinatário) da conexão TCP entre Alice e Bob. Assim, ela pode falsificar datagramas IP enviados a Bob (utilizando o endereço de Alice como endereço de fonte) contendo segmentos TCP válidos e uma carga útil de usuário arbitrária. Imagine a devastação que Trudy pode causar no relacionamento entre Bob e Alice!

Vários ataques a redes e ameaças à segurança são discutidos na coletânea de ensaios [Denning, 1997] e no livro muito bom de ler escrito por Rubin [Rubin, 2001]. O CERT Coordination Center [CERT, 2004] mantém um resumo de ataques comunicados. Veja também [Cisco Security, 2004; Voydock, 1983; Bhimani, 1996].

8.8 Segurança em muitas camadas: estudos de casos

Nas seções anteriores, examinamos questões fundamentais de segurança na rede, incluindo criptografia de chaves simétricas e criptografia de chaves públicas, autenticação, distribuição de chaves, integridade de mensagens e assinaturas digitais. Vamos agora examinar como essas ferramentas estão sendo usadas para proporcionar segurança na Internet. É interessante que é possível fornecer serviços de segurança em qualquer uma das quatro camadas superiores da pilha de protocolos da Internet. Quando é fornecida segurança para um protocolo específico de camada de aplicação, a aplicação que usa o protocolo desfruta de um ou mais serviços de segurança, como confidencialidade, autenticação ou integridade. Quando a segurança é fornecida para um protocolo de camada de transporte, todas as aplicações que usam o protocolo desfrutam dos serviços de segurança do protocolo de transporte. Quando a segurança é fornecida na camada de rede, de hospedeiro a hospedeiro, todos os segmentos de camada de transporte (e, por conseguinte, todos os dados de camada de aplicação) desfrutam dos serviços de segurança da camada de rede. Quando a segurança é fornecida por enlace, então todos os dados de todos os quadros que estão trafegando pelo enlace recebem os serviços de segurança do enlace.

Nesta seção, veremos como as ferramentas de segurança estão sendo usadas nas camadas de aplicação, transporte e rede. Para mantermos a consistência da estrutura geral deste livro, começaremos pelo topo da pilha de protocolos, discutindo a segurança na camada de aplicação. Nossa abordagem é usar uma aplicação específica, o e-mail, como um estudo de caso para a segurança na camada de aplicação. Em seguida, desceremos pela pilha de protocolos. Examinaremos o protocolo SSL (que fornece segurança ao TCP na camada de transporte), o Ipsec (que fornece segurança à camada de rede) e a segurança do protocolo IEEE 802.11 para LANs sem fio.

É bem possível que você esteja pensando por que a funcionalidade de segurança da Internet está sendo fornecida em mais de uma camada. Não seria suficiente simplesmente fornecer funcionalidade de segurança na camada de rede e esquecer o assunto? Há duas respostas para essa pergunta. Em primeiro lugar, embora a segurança na camada de rede possa oferecer um 'cobertor de segurança' com a criptografia de todos os dados dos datagramas (isto é, de todos os segmentos de camada de transporte) e com a autenticação de todos os endereços IP de origem, ela não pode garantir segurança no nível do usuário. Por exemplo, um site comercial não pode confiar na segurança da camada IP para autenticar um cliente que está comprando mercadorias nesse site. Assim, há necessidade de uma funcionalidade de segurança nas camadas mais altas, bem como um cobertor de segurança nas camadas mais baixas. Em segundo lugar, em geral é mais fácil disponibilizar novos serviços de Internet, incluindo serviços de segurança, nas camadas mais altas da pilha de protocolos. Enquanto aguardamos que a segurança seja disseminada de maneira ampla na camada de rede, o que provavelmente ainda levará alguns anos para acontecer, muitos desenvolvedores de aplicação tomam a iniciativa de fazê-lo mesmo assim e introduzem a funcionalidade de segurança em suas aplicações favoritas. Um exemplo clássico é o PGP (Pretty Good Privacy), que fornece e-mail seguro (e será discutido mais adiante nesta seção). Como exige apenas programas de aplicação cliente e servidor, o PGP foi uma das primeiras tecnologias de segurança a ser usada amplamente na Internet.

8.8.1 E-mail seguro

Nesta seção, usaremos muitas das técnicas apresentadas na seção anterior para criar um projeto de alto nível para um sistema de e-mail seguro. Criamos esse projeto de alto nível de maneira incremental, introduzindo, a cada etapa, novos serviços de segurança. Em nosso projeto de um sistema de e-mail seguro, vamos manter em mente o exemplo picante apresentado na Seção 8.1 — o caso de amor entre Alice e Bob. Imagine que Alice quer enviar uma mensagem de e-mail para Bob e Trudy quer bisbilhotar.

Antes de avançar e projetar um sistema de e-mail seguro para Alice e Bob, devemos considerar quais características de segurança seriam as mais desejáveis para eles. A primeira característica, e a mais importante, é a confidencialidade. Como foi discutido na Seção 8.1, nem Alice nem Bob querem que Trudy leia a

mensagem de e-mail de Alice. A segunda característica que Alice e Bob provavelmente gostariam de ver no sistema de e-mail seguro é a autenticação do remetente. Em particular, quando Bob receber a seguinte mensagem de Alice: "Eu não o amo mais. Nunca mais quero vê-lo. Da anteriormente sua, Alice", ele naturalmente gostaria de ter certeza de que a mensagem veio de Alice, e não de Trudy. Outra característica de segurança de que os dois amantes gostariam de dispor é a integridade de mensagem, isto é, a certeza de que a mensagem que Alice enviar não será modificada no trajeto até Bob. Por fim, o sistema de e-mail deve fornecer autenticação do receptor, isto é, Alice quer ter certeza de que ela de fato está enviando a mensagem para Bob, e não para outra pessoa (por exemplo, Trudy) que possa estar se passando por Bob.

Portanto, vamos começar abordando a preocupação mais premente de Alice e Bob, a confidencialidade. A maneira mais direta de conseguir confidencialidade é Alice criptografar a mensagem com tecnologia de chaves simétricas (como DES ou AES) e Bob decriptar a mensagem ao recebê-la. Como discutido na Seção 8.2, se a chave simétrica for suficientemente longa e se somente Alice e Bob possuírem a chave, então será extremamente difícil que alguém (incluindo Trudy) leia a mensagem. Embora essa seja uma abordagem direta, ela apresenta a dificuldade fundamental que discutimos na Seção 8.2 — é difícil distribuir uma chave simétrica de modo que apenas Bob e Alice tenham cópias dela. Portanto, é natural que consideremos uma abordagem alternativa — a criptografia de chaves públicas (usando, por exemplo, RSA). Na abordagem de chaves públicas, Bob disponibiliza publicamente sua chave pública (por exemplo, em um servidor de chaves públicas ou em sua página Web pessoal) e Alice criptografa sua mensagem com a chave pública de Bob, e envia a mensagem criptografada para o endereço de e-mail de Bob. (A mensagem criptografada é encapsulada com cabeçalhos MIME e enviada por SMTP comum, como discutimos na Seção 2.4.) Quando Bob recebe a mensagem, ele simplesmente a decripta com sua chave privada. Admitindo que Alice tenha certeza de que aquela chave pública é a de Bob (e que é suficientemente longa), essa abordagem é um meio excelente de fornecer a confidencialidade desejada. Um problema, contudo, é que a criptografia de chaves públicas é relativamente ineficiente, sobretudo para mensagens longas. (Mensagens de e-mail longas agora são muito comuns na Internet, devido ao crescimento da utilização de anexos, imagens, áudio e vídeo.)

Para superar o problema da eficiência, vamos fazer uso de uma chave de sessão (discutida na Seção 8.5). Em particular, Alice (1) escolhe uma chave simétrica, K_S, aleatoriamente, (2) criptografa sua mensagem m com a chave simétrica K_S, (3) criptografa a chave simétrica com a chave pública de Bob, K_B^+, (4) concatena a mensagem criptografada e a chave simétrica criptografada de modo que formem um 'pacote' e (5) envia o pacote ao endereço de e-mail de Bob. Os passos estão ilustrados na Figura 8.27. (Nessa figura e nas subseqüentes, o sinal '+' dentro de um círculo representa formar a concatenação e o sinal '−' dentro de um círculo, desfazer a concatenação.) Quando Bob receber o pacote, ele (1) usará sua chave privada K_B^- para obter a chave simétrica K_S, e (2) utilizará a chave simétrica K_S para decriptar a mensagem m,

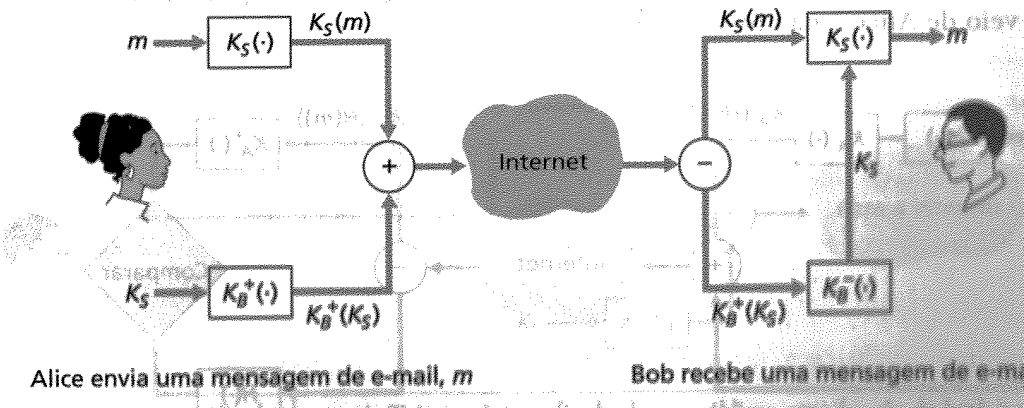


Figura 8.27 Alice usa uma chave de sessão simétrica, KS, para enviar um e-mail secreto para Bob

História

Phil Zimmermann e o PGP

Philip R. Zimmermann é o criador do Pretty Good Privacy (PGP). Por causa disso, ele foi alvo de uma investigação criminal que durou três anos, porque o governo norte-americano sustentava que as restrições sobre a exportação de software de criptografia tinham sido violadas quando o PGP se espalhou por todo o mundo após sua publicação como software de uso livre em 1991. Após ter sido liberado como software compartilhado, alguém o colocou na Internet e estrangeiros o descarregaram. Nos Estados Unidos, os programas de criptografia são classificados como artefatos militares por lei federal e não podem ser exportados.

A despeito da falta de financiamento, da inexistência de representantes legais, da falta de uma empresa para lhe dar apoio e das intervenções governamentais, mesmo assim o PGP se tornou o software de criptografia para e-mail mais usado no mundo. O extraordinário é que o governo dos Estados Unidos pode ter contribuído inadvertidamente para a disseminação do PGP devido ao caso Zimmermann.

O governo norte-americano arquivou o caso no início de 1996. O anúncio foi festejado pelos ativistas da Internet. O caso Zimmermann tinha se transformado na história de um inocente que lutava por seus direitos contra os desmandos de um governo poderoso. A desistência do governo foi uma notícia bem-vinda, em parte por causa da campanha em favor da censura na Internet desencadeada pelo Congresso e em parte por causa dos esforços feitos pelo FBI para conseguir maior amplitude para a espionagem governamental.

Após o governo ter arquivado o caso, Zimmermann fundou a PGP Inc., que foi adquirida pela Network Associates em dezembro de 1997. Zimmermann é agora membro do conselho da Network Associates, bem como consultor independente para assuntos de criptografia.

Agora que projetamos um sistema de e-mail seguro que fornece confidencialidade, vamos desenvolver um outro sistema que forneça autenticação do remetente e também integridade de mensagem. Vamos supor, no momento, que Alice e Bob não estejam mais preocupados com confidencialidade (querem compartilhar seus sentimentos com todos!) e que estejam somente preocupados com a autenticação do remetente e com a integridade da mensagem. Para realizar essa tarefa, usaremos assinaturas digitais e resumos de mensagem, como descrito na Seção 8.4. Especificamente, Alice (1) aplica uma função de hash H (por exemplo, MD5) à sua mensagem m, para obter um resumo de mensagem, (2) assina o resultado da função de hash com sua chave privada K_A^- para criar uma assinatura digital, (3) concatena a mensagem original (não criptografada) com a assinatura para criar um pacote e (4) envia o pacote ao endereço de e-mail de Bob. Quando Bob recebe o pacote, ele (1) aplica a chave pública de Alice, K_A^+, ao resumo de mensagem assinado e (2) compara o resultado dessa operação com o próprio hash H da mensagem. As etapas são ilustradas na Figura 8.28. Como discutimos na Seção 8.5, se os dois resultados forem iguais, Bob poderá ter razoável certeza de que a mensagem veio de Alice e não foi alterada.

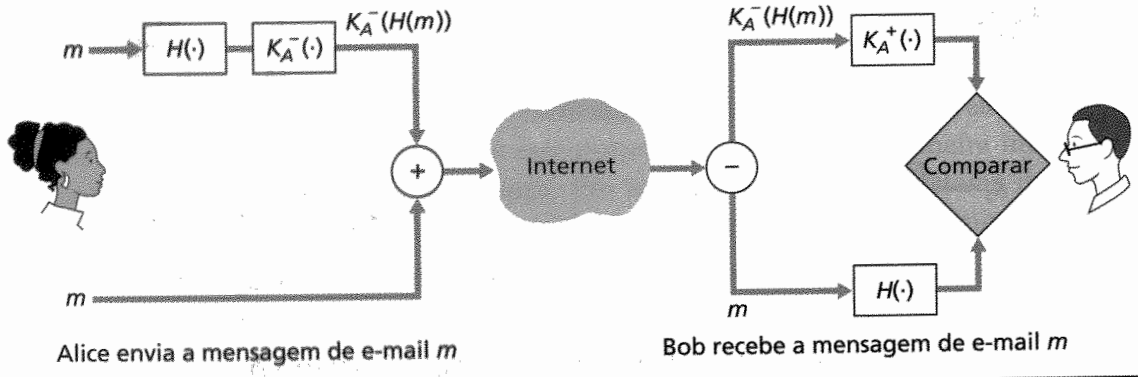


Figura 8.28 Utilização de funções de hash e assinaturas digitais para fornecer autenticação de remetente e integridade de mensagem

Vamos considerar agora o projeto de um sistema de e-mail que forneça confidencialidade, autenticação de remetente e integridade de mensagem. Isso pode ser feito pela combinação dos procedimentos das figuras 8.27 e 8.28. Em primeiro lugar, Alice cria um pacote preliminar, exatamente como mostra a Figura 8.28, constituído de sua mensagem original junto com um hash da mensagem assinado digitalmente. Em seguida, ela trata esse pacote preliminar como uma mensagem em si e envia essa nova mensagem seguindo as etapas do remetente mostradas na Figura 8.27, criando um novo pacote, que é enviado a Bob. As etapas seguidas são mostradas na Figura 8.29. Quando Bob recebe o pacote, ele aplica primeiramente seu lado da Figura 8.27 e depois seu lado da Figura 8.28. É preciso ficar claro que esse projeto atinge o objetivo de fornecer confidencialidade, autenticação de remetente e integridade de mensagem. Note que nesse esquema Alice utiliza criptografia de chaves públicas duas vezes: uma vez com sua chave privada e uma vez com a chave pública de Bob. De maneira semelhante, Bob também aplica a criptografia de chaves públicas duas vezes — uma vez com sua chave privada e uma vez com a chave pública de Alice.

O projeto de e-mail seguro ilustrado na Figura 8.29 provavelmente fornece segurança satisfatória para os usuários de e-mail na maioria das ocasiões. Mas ainda resta uma questão importante a ser abordada. O projeto da Figura 8.29 requer que Alice obtenha a chave pública de Bob e que Bob obtenha a chave pública de Alice. A distribuição dessas chaves é um problema nada trivial. Por exemplo, Trudy poderia se disfarçar de Bob e dar a Alice sua própria chave pública, dizendo que é a chave pública de Bob. Como aprendemos na Seção 8.5, uma abordagem popular para distribuir chaves públicas com segurança é utilizar uma autoridade certificadora.

PGP

Projetado originalmente por Phil Zimmermann em 1991, o PGP (**pretty good privacy** — privacidade razoável) é um esquema de criptografia para e-mail que se tornou um padrão de fato. O site do PGP é acessado mais de um milhão de vezes por mês por usuários de 166 países [PGPI, 2004]. Versões do PGP estão disponíveis em domínio público; por exemplo, você pode encontrar o software PGP para sua plataforma favorita, bem como grande quantidade de material de leitura interessante, na home page internacional do PGP [PGPI, 2004]. (Um ensaio particularmente interessante, escrito pelo autor do PGP, é encontrado em [Zimmermann, 2004].) O PGP também pode ser adquirido no comércio como acessório plug-in para muitos agentes de usuário de e-mail, incluindo o Exchange e o Outlook, da Microsoft, e o Eudora, da Qualcomm. O projeto do PGP é, em essência, idêntico ao projeto apresentado na Figura 8.29. Dependendo da versão, o software do PGP usa MD5 ou SHA para processar o resumo de mensagem; CAST, DES triplo ou IDEA para criptografar chaves simétricas, e RSA para criptografar chaves públicas. Além disso, o PGP fornece compressão de dados.

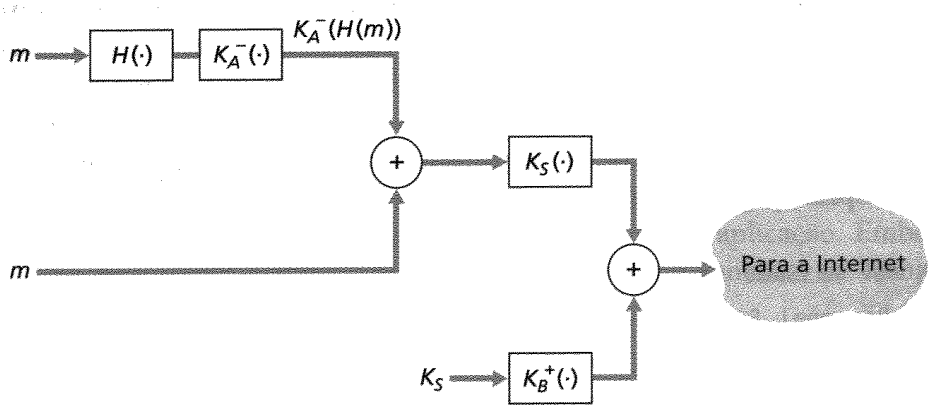


Figura 8.29 Alice usa criptografia de chaves simétricas, criptografia de chaves públicas, uma função de hash e uma assinatura digital para fornecer confidencialidade, autenticação de remetente e integridade de mensagem

```
- - - - - BEGIN PGP SIGNED MESSAGE - - - - -
Hash: SHA1
Bob:
Can I see you tonight?
Passionately yours, Alice
- - - - - BEGIN PGP SIGNATURE - - - - -
Version: PGP for Personal Privacy 5.0
Charset: noconv
yhHJRHhGJGhgg/12EpJ+lo8gE4vB3mqJhFEvZP9t6n7G6m5Gw2
- - - - - END PGP SIGNATURE - - - - -
```

Figura 8.30 Uma mensagem PGP assinada

Quando o PGP é instalado, o software cria um par de chaves públicas para o usuário. A chave pública pode então ser colocada no site do usuário ou em um servidor de chaves públicas. A chave privada é protegida pelo uso de uma senha. A senha tem de ser informada todas as vezes que o usuário acessar a chave privada. O PGP oferece ao usuário a opção de assinar digitalmente a mensagem, criptografar a mensagem ou, ainda, ambas as opções: assinar digitalmente e criptografar a mensagem. A Figura 8.30 mostra uma mensagem PGP assinada. Essa mensagem aparece após o cabeçalho MIME. Os dados codificados da mensagem correspondem a $K_A^-(H(m))$, isto é, ao resumo de mensagem assinado digitalmente. Como discutimos antes, para que Bob verifique a integridade da mensagem, ele precisa ter acesso à chave pública de Alice.

A Figura 8.31 mostra uma mensagem PGP secreta. Essa mensagem também aparece após o cabeçalho MIME. É claro que a mensagem em texto aberto não está incluída na mensagem de e-mail secreta. Quando um remetente (como Alice) quer não apenas a confidencialidade, mas também a integridade, o PGP contém uma mensagem como a da Figura 8.31 dentro da mensagem da Figura 8.30.

O PGP também fornece um mecanismo para certificação de chaves públicas, mas esse mecanismo é bem diferente daquele da autoridade certificadora mais convencional. As chaves públicas do PGP são certificadas por uma rede de confiança. A própria Alice pode certificar qualquer par chave/usuário quando ela achar que esse par realmente está correto. Além disso, o PGP permite que Alice declare que ela confia em outro usuário para atestar a autenticidade de mais chaves. Alguns usuários do PGP assinam reciprocamente suas chaves montando grupos de assinatura de chaves. Os usuários se reúnem fisicamente, trocam disquetes contendo as chaves públicas e certificam suas chaves reciprocamente, assinando-as com suas chaves privadas. As chaves públicas PGP também são distribuídas por **servidores de chaves públicas PGP**, pela Internet. Quando um usuário apresenta uma chave pública a esse servidor, este armazena uma cópia da chave, envia uma cópia a todos os outros servidores de chaves públicas e apresenta a chave a quem quer que a solicite. Embora os grupos de assinantes de chaves e os servidores de chaves públicas PGP realmente existam, o modo mais comum utilizado pelos usuários para distribuir suas chaves públicas é, de longe, colocá-las em suas próprias páginas Web e anunciá-las em seus e-mails.

```
- - - - - BEGIN PGP MESSAGE - - - - -
Version: PGP for Personal Privacy 5.0
u2R4d+/jKmn8Bc5+hgDsqAewsDfrGdszX68liKm5F6Gc4sDfcXyt
RfdS10juHgbcfDssWe7/K=lKhnMikLoO+1/BvcX4t==Ujk9PbcD4
Thdf2awQfgHbnmK1ok8iy6gThlp
- - - - - END PGP MESSAGE
```

Figura 8.31 Uma mensagem PGP secreta

8.8.2 Camada de sockets de segurança (SSL) e segurança na camada de transporte (TLS)

Na seção anterior, consideramos como a camada de aplicação pode usar (no e-mail seguro) os vários mecanismos de segurança que estudamos antes neste capítulo: criptografia, autenticação, distribuição de chaves, integridade de mensagem e assinaturas digitais. Nesta seção, continuaremos nosso estudo de caso de vários mecanismos de segurança descendo mais uma camada na pilha de protocolos e abordando sockets seguros e uma camada de transporte segura. Consideraremos o comércio na Internet como uma aplicação motivadora, visto que transações comerciais e financeiras são uma importante força propulsora da segurança na Internet.

Vamos considerar um cenário típico de comércio pela Internet. Bob está navegando pela Web e encontra o site Alice Incorporated, que vende um produto. Esse site apresenta um formulário no qual Bob deve informar a quantidade desejada, seu endereço e o número de seu cartão de crédito. Bob registra essas informações, clica em 'apresentar' e, então, aguarda o recebimento do produto (por exemplo, pelo correio convencional); ele também espera receber a cobrança da mercadoria na próxima fatura de seu cartão de crédito. Isso tudo parece ser muito bom, mas, se não forem tomadas medidas de segurança — como criptografia ou autenticação —, Bob poderá ter algumas surpresas:

- Um intruso pode interceptar o pedido de compra, obter as informações sobre o cartão de crédito e, então, fazer compras na conta de Bob.
- O site pode apresentar o famoso logotipo da Alice Incorporated, mas, na realidade, ser mantido por Trudy, que está se fazendo passar por Alice Incorporated. Trudy poderia se apossar do dinheiro de Bob e fugir. Ou poderia fazer compras e enviar a fatura para a conta de Bob.

Muitas outras surpresas também são possíveis; discutiremos algumas delas na próxima subseção. Mas esses dois problemas apresentados são os mais sérios. O comércio pela Internet usando o protocolo SSL pode enfrentar ambos os problemas.

A SSL (*secure sockets layer* — camada de sockets de segurança), originalmente desenvolvida pela Netscape (veja também [Woo, 1994]), é um protocolo projetado para fornecer criptografia de dados e autenticação entre um cliente e um servidor Web. O protocolo começa com uma fase de apresentação mútua que negocia um algoritmo de criptografia (por exemplo, DES ou IDEA) e chaves e autentica o servidor para o cliente. Opcionalmente, o cliente pode também ser autenticado para o servidor. Uma vez concluída a apresentação mútua e iniciada a transmissão de dados da aplicação, todos os dados são criptografados usando chaves de sessão negociadas durante a fase de apresentação mútua. O protocolo SSL é amplamente usado no comércio pela Internet, sendo implementado em quase todos os browsers populares e servidores Web. Além disso, ele é a base do protocolo TLS (*transport layer security* — segurança da camada de transporte) [RFC 2246].

Os protocolos SSL e TLS não são limitados à aplicação na Web; por exemplo, eles podem ser usados de maneira semelhante para autenticação e criptografia de dados no acesso ao correio IMAP (Internet Mail Access Protocol). A camada SSL pode ser vista como uma camada que se situa entre a camada de aplicação e a camada de transporte. No lado remetente, o protocolo SSL recebe dados (como uma mensagem HTTP ou IMAP vinda de uma aplicação), criptografa-os e direciona os dados criptografados a um socket TCP. No lado receptor, o SSL lê socket TCP, decripta os dados e os direciona à aplicação. Embora possa ser usado com muitas aplicações da Internet, o protocolo SSL será discutido no contexto da Web, em que é utilizado principalmente para o comércio pela Internet.

O SSL tem as seguintes características:

- *Autenticação do servidor SSL*, que permite que um usuário confirme a identidade de um servidor. Um browser habilitado para SSL mantém uma lista de CAs de confiança, juntamente com as chaves públicas dessas CAs. Quando um browser quer fazer negócios com um servidor Web habilitado para SSL, ele obtém um certificado do servidor contendo a chave pública deste. O certifica-

do é emitido (ou seja, é assinado digitalmente) por uma CA que aparece na lista de CAs de confiança do cliente. Essa característica permite que o browser autentique o servidor antes de o usuário informar o número de seu cartão de crédito. No contexto do exemplo apresentado, essa autenticação do servidor permite que Bob verifique se ele está realmente enviando o número de seu cartão de crédito à Alice Incorporated, e não a alguém que possa estar se fazendo passar pela Alice Incorporated.

- *Autenticação do cliente SSL*, que permite que um servidor confirme a identidade de um usuário. Análoga à autenticação do servidor, a autenticação do cliente faz uso de certificados dos clientes que também foram emitidos pelas CAs. Essa autenticação é importante se o servidor, por exemplo, for um banco que está enviando informações financeiras confidenciais para um cliente e quer verificar a identidade do receptor. A autenticação do cliente, embora suportada pela SSL, é opcional. Para não desviar nosso foco de atenção, daqui em diante vamos ignorar a autenticação do cliente.
- *Uma sessão SSL criptografada*, na qual toda a informação enviada entre browser e servidor é criptografada pelo software remetente (browser ou servidor Web). Essa confidencialidade pode ser importante tanto para o cliente quanto para o comerciante. O protocolo SSL também provê um mecanismo que detecta a alteração de informação por um intruso.

Como a SSL funciona

Um usuário — digamos, Bob — navega pela Web e clica sobre um ponteiro que o transporta a uma página segura abrigada pelo servidor que utiliza SSL de Alice. A parte do URL que se refere ao protocolo para essa página é https, e não o http comum. O browser e o servidor executam o protocolo de apresentação mútua da SSL que (1) autentica o servidor e (2) gera uma chave simétrica compartilhada. Ambas as tarefas fazem uso da tecnologia de chaves públicas RSA. O fluxo principal de eventos na fase de apresentação mútua é mostrado na Figura 8.32. Durante essa fase, Alice envia a Bob seu certificado, com o qual Bob obtém a chave pública de Alice. Bob então cria uma chave simétrica aleatória, criptografa-a com a chave pública de Alice e a envia criptografada para Alice. Agora, Bob e Alice compartilham uma chave de sessão simétrica. Tão logo esse protocolo de apresentação mútua esteja concluído, todos os dados enviados entre o browser e o servidor (por conexões TCP) são criptografados usando a chave de sessão simétrica.

Agora que apresentamos uma visão de alto nível da SSL, vamos examinar melhor alguns dos detalhes mais importantes. A fase de apresentação mútua da SSL percorre as seguintes etapas:

1. O browser envia ao servidor o número da versão da SSL que utiliza e as preferências criptográficas. Ele envia essas preferências criptográficas porque browser e servidor negociam qual algoritmo de chaves simétricas vão usar.
2. O servidor envia ao browser o número da versão da SSL que utiliza, suas preferências criptográficas e seu certificado. Lembre-se de que o certificado contém a chave pública RSA do servidor e é certificado por uma CA, isto é, o certificado foi assinado com a chave privada da CA.
3. O browser tem uma lista de CAs de confiança e uma chave pública para cada CA da lista. Quando recebe o certificado do servidor, o browser verifica se a CA está na lista. Se não estiver, o usuário será avisado do problema e informado de que não pode ser estabelecida uma conexão criptografada e autenticada. Se a CA estiver na lista, o browser usará a chave pública da CA para validar o certificado e obter a chave pública do servidor.
4. O browser gera uma chave de sessão simétrica, criptografa-a com a chave pública do servidor e envia a chave de sessão criptografada ao servidor.
5. O browser envia uma mensagem ao servidor informando que as futuras mensagens do cliente serão criptografadas com a chave de sessão. Ele então envia uma mensagem separada (criptografada) indicando que sua parte na apresentação mútua está encerrada.

Figura 8.32 Visão de alto nível da fase de apresentação mútua da SSL

6. O servidor envia uma mensagem ao browser informando que as futuras mensagens do servidor serão criptografadas com a chave de sessão. Ele então envia uma mensagem separada (criptografada) indicando que sua parte na apresentação mútua está encerrada.
7. Agora, a apresentação mútua SSL está concluída e começa a sessão SSL. O browser e o servidor usam as chaves de sessão para criptografar e decriptar os dados que enviam um ao outro e para validar sua integridade.

A apresentação mútua SSL tem na verdade muito mais etapas do que as apresentadas. Você pode encontrar mais informações sobre a SSL em [Netscape Security, 1998]. Além das compras feitas com cartão de crédito, destacamos que a SSL pode ser (e é) usada para outras transações financeiras, incluindo home banking e comércio de ações.

As limitações da SSL no comércio pela Internet

Devido à sua simplicidade e ao seu desenvolvimento precoce, a SSL é amplamente implementada em browsers, servidores e produtos para comércio pela Internet. Esses servidores e browsers capacitados para SSL fornecem uma plataforma popular para transações com cartões de crédito. No entanto, devemos ter sempre em mente que a SSL não foi produzida especificamente para transações com cartões de pagamento de crédito, mas para comunicações genéricas seguras entre um cliente e um servidor. Por causa de seu projeto genérico, faltam à SSL muitas características que o setor de cartões de crédito gostaria de ver incorporadas a um protocolo de comércio pela Internet.

Considere novamente o que acontece quando Bob faz uma compra no site Alice Incorporated por meio da SSL. O certificado assinado que Bob recebe de Alice garante que ele realmente está negociando com Alice Incorporated e que Alice Incorporated é uma empresa confiável. Contudo, o certificado genérico não indica se Alice Incorporated está autorizada a aceitar compras com cartão de crédito nem se a empresa é um estabelecimento comercial de confiança. Isso abre uma porta para a fraude do comerciante. E há um problema semelhante com a autorização do cliente. Mesmo que seja usada a autenticação do cliente SSL, o certificado do cliente não vincula Bob a um determinado cartão de crédito autorizado; assim, Alice Incorporated

não pode ter certeza de que Bob está autorizado a fazer uma compra usando cartão de crédito. Isso abre a porta para todos os tipos de fraude, incluindo compras com cartões de crédito roubados e recusa de bens comprados.

8.8.3 Segurança na camada de rede: IPsec

O protocolo de **segurança IP**, mais conhecido como **IPsec**, é um conjunto de protocolos que oferece segurança na camada de rede. O IPsec é um mecanismo bastante complexo — partes dele estão descritas em mais de uma dúzia de RFCs. Nesta seção, discutiremos o IPsec em um contexto específico, a saber, admitindo que todos os hospedeiros da Internet suportam o IPsec. Esse contexto, embora esteja ainda muito distante, simplificará a discussão e nos ajudará a entender as características principais do IPsec. Dois RFCs fundamentais são o RFC 2401, que descreve a arquitetura geral de segurança IP, e o RFC 2411, que proporciona uma visão do conjunto de protocolos IPsec e apresenta os documentos que o descrevem.

Antes de examinar os aspectos específicos do IPsec, vamos dar um passo atrás e considerar o que significa oferecer segurança na camada de rede. Considere, em primeiro lugar, o que significa fornecer **confidencialidade na camada de rede**. A camada de rede ofereceria confidencialidade se todos os dados portados por todos os datagramas IP fossem criptografados. Isso significa que, sempre que um hospedeiro quisesse enviar um datagrama, ele criptografaria o campo de dados do datagrama antes de despachá-lo para a rede. Em princípio, a criptografia poderia ser feita com chaves simétricas, chaves públicas ou chaves de sessão negociadas com a utilização de criptografia de chaves públicas. O campo de dados poderia ser um segmento TCP, um segmento UDP, uma mensagem ICMP e assim por diante. Se esse serviço de camada de rede existisse, todos os dados enviados por hospedeiros — incluindo e-mail, páginas Web, mensagens de controle e mensagens de administração (como ICMP e SNMP) — ficariam ocultos de qualquer terceiro que estivesse grampeando a rede. Assim, esse serviço forneceria alguma cobertura geral para todo tráfego da Internet e daria a todos nós, por conseguinte, uma certa sensação de segurança.

Além da confidencialidade, poderíamos querer que a camada de rede também fornecesse **autenticação de fonte**. Quando um hospedeiro destinatário recebesse um datagrama IP com um endereço IP de fonte específico, ele autenticaria a fonte certificando-se de que o datagrama IP foi de fato gerado pelo hospedeiro cujo endereço IP de fonte é aquele. Esse serviço evitaria que fossem autenticados endereços IP falsificados.

No conjunto de protocolos IPsec, há dois protocolos principais: o protocolo de **Autenticação de Cabeçalho** (Authentication Header — AH) e o protocolo de **Segurança** de **Encapsulamento de Carga Útil** (Encapsulation Security Payload — ESP). Quando um hospedeiro de origem envia datagramas seguros a um hospedeiro de destino, ele o faz com o protocolo AH ou com o protocolo ESP. O protocolo AH fornece autenticação de fonte e integridade de dados, mas não oferece confidencialidade. O protocolo ESP fornece autenticação, integridade de dados e confidencialidade. Como oferece mais serviços, esse protocolo é naturalmente mais complicado e exige mais processamento do que o protocolo AH.

Tanto no protocolo AH quanto no ESP, antes de enviar datagramas seguros de um a outro, os hospedeiros de origem e de destino fazem uma apresentação mútua e criam uma ligação lógica de camada de rede. Esse canal lógico é denominado **acordo de segurança** (*security association* — AS). Assim, o IPsec transforma a tradicional camada de rede não orientada para conexão da Internet em uma rede com conexões lógicas! A conexão lógica definida por um AS é uma *conexão simplex*, isto é, unidirecional. Se ambos os hospedeiros quiserem enviar datagramas seguros reciprocamente, então será preciso estabelecer dois ASs (isto é, conexões lógicas), um em cada direção. Um AS é identificado, exclusivamente, pelos seguintes três elementos:

- um identificador de protocolo de segurança (AH ou ESP);
- o endereço IP de destino para a conexão simplex, e
- um identificador de conexão de 32 bits denominado Índice de Parâmetros de Segurança (*security parameter index* — SPI).

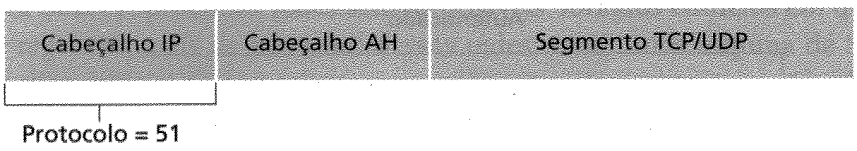


Figura 8.33 Posição do cabeçalho AH no datagrama IP

Para um dado AS (isto é, uma dada conexão lógica de hospedeiro de origem a hospedeiro de destino), cada datagrama IPsec terá um campo especial para o SPI. Todos os datagramas do AS usarão o mesmo valor de SPI nesse campo.

Protocolo de autenticação de cabeçalho (AH)

Como mencionamos anteriormente, o protocolo AH fornece identificação do hospedeiro de origem e integridade de dados, mas não confidencialidade. Quando um determinado hospedeiro de origem quer enviar um ou mais datagramas seguros a um destino particular, ele primeiramente estabelece um AS com o destino. Após ter estabelecido o AS, ele pode enviar datagramas seguros ao hospedeiro de destino. Os datagramas seguros contêm o cabeçalho AH, que é inserido entre o datagrama de dados IP original (por exemplo, um segmento TCP ou UDP) e o cabeçalho IP, como mostra a Figura 8.33. Assim, o cabeçalho AH amplia o campo de dados original, e esse campo ampliado é encapsulado como um datagrama IP padrão. Para o campo de protocolo no cabeçalho IP, o valor 51 é usado para indicar que o datagrama contém um cabeçalho AH. Quando o hospedeiro de destino recebe o datagrama IP, ele percebe o número 51 no campo de protocolo e processa o datagrama usando o protocolo AH. (Lembre-se de que o campo de protocolo do datagrama IP é usado para determinar o protocolo da camada acima — por exemplo, UDP, TCP ou ICMP — para a qual será passada a porção de dados de um datagrama IP.) Roteadores intermediários processam os datagramas exatamente como sempre o fazem — examinam o endereço de destino IP e repassam os datagramas de acordo com esse destino.

O cabeçalho AH contém diversos campos, tais como:

- *Campo Próximo Cabeçalho*, que desempenha o papel que o campo de protocolo desempenha em um datagrama comum. Ele indica se os dados que se seguem ao cabeçalho AH são um segmento TCP, um segmento UDP, um segmento ICMP e assim por diante. (Lembre-se de que o campo de protocolo do datagrama IP agora está sendo usado para indicar o protocolo AH e, assim, não pode mais ser utilizado para identificar o protocolo da camada de transporte.)
- *Campo SPI (Security Parameter Index)*, um valor arbitrário de 32 bits que, combinado ao endereço de destino IP e ao protocolo de segurança, identifica exclusivamente o AS para o datagrama.
- *Campo Número de Seqüência*, um campo de 32 bits que contém um número de seqüência para cada datagrama. Ele é inicialmente estabelecido quando da criação de um AS. O protocolo AH usa os números de seqüência para evitar os ataques de reprodução e do homem do meio (veja a Seção 8.3).
- *Campo Autenticação de Dados*, um campo de comprimento variável que contém um resumo de mensagem assinado (isto é, uma assinatura digital) para esse datagrama. O resumo de mensagem é calculado sobre o datagrama IP original, fornecendo, desse modo, autenticação do hospedeiro de origem e integridade de datagrama IP. A assinatura digital é processada usando o algoritmo de autenticação especificado pelo AS, como MD5 ou SHA.

Quando o hospedeiro de destino recebe um datagrama IP com um cabeçalho AH, ele determina o AS para o datagrama e, em seguida, autentica a integridade do datagrama processando o campo da autenticação de dados. O esquema de autenticação do IPsec (tanto para o protocolo AH como para o ESP) usa um sistema denominado HMAC, que é um resumo de mensagem criptografado descrito no RFC 2104. O HMAC usa uma chave secreta compartilhada pelas duas partes, em vez de métodos de chaves públicas para autenticação de mensagens. Para mais detalhes sobre o protocolo AH, consulte o RFC 2402.

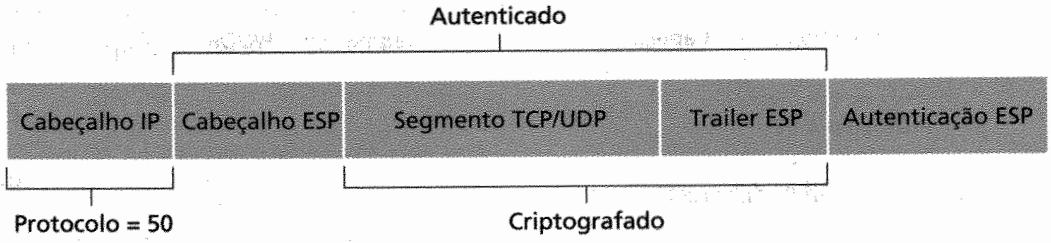


Figura 8.34 Os campos do protocolo ESP no datagrama IP

O protocolo ESP

O protocolo ESP fornece confidencialidade na camada de rede, bem como autenticação do hospedeiro de origem. Novamente, tudo começa com um hospedeiro de origem estabelecendo um AS com um hospedeiro de destino. Como mostra a Figura 8.34, um datagrama seguro é criado envolvendo o datagrama IP original com campos de cabeçalho e de trailer e, em seguida, inserindo esses dados encapsulados no campo de dados de um datagrama IP. No campo de protocolo do cabeçalho do datagrama IP, o valor 50 é usado para indicar que o datagrama contém um cabeçalho e um trailer ESP. Quando o hospedeiro de destino recebe um datagrama IP, ele percebe o número 50 no campo de protocolo e processa o datagrama usando o protocolo ESP. Como mostra a Figura 8.34, o datagrama IP original e o campo Trailer ESP são criptografados. A confidencialidade é fornecida pela criptografia DES-CBC [RFC 2405]. O cabeçalho ESP consiste em um campo de 32 bits para o SPI e um campo de 32 bits para o número de seqüência, os quais desempenham exatamente a mesma função que exercem no protocolo AH. O trailer contém o campo Próximo Cabeçalho, cuja função é exatamente a mesma realizada no protocolo AH. Note que, como o campo Próximo Cabeçalho é criptografado juntamente com os dados originais, um intruso não conseguirá descobrir qual protocolo de transporte está sendo usado. Após o trailer, há um campo Autenticação de Dados, que também desempenha o mesmo papel que o campo da autenticação de dados no protocolo AH. Para mais detalhes sobre o protocolo ESP, consulte o RFC 2406.

AS e administração de chaves

Para a disponibilização bem-sucedida do IPsec, é necessário um esquema escalável e automatizado de AS e gerenciamento de chaves. Vários protocolos foram definidos para realizar essas tarefas.

- O **Algoritmo de Troca de Chaves da Internet** (Internet Key Exchange — IKE) [RFC 2409] é o protocolo default de administração de chaves para o IPsec.
- O **Protocolo de Acordo de Segurança e de Administração de Chaves na Internet** (Internet Security Association and Key Management Protocol — ISKMP) define procedimentos para estabelecer e encerrar ASs [RFC 2407; RFC 2408]. O AS do ISKMP é completamente independente da troca de chaves do IKE.

Isso encerra nosso resumo sobre o IPsec. Discutimos o IPsec no contexto do IPv4 e do modo de transporte. O IPsec também define um modo túnel, no qual são os roteadores — e não os hospedeiros — que introduzem a funcionalidade de segurança. Por fim, ele descreve os procedimentos de criptografia para o IPv6 assim como para o IPv4.

8.8.4 Segurança em IEEE 802.11

Segurança é uma preocupação particularmente importante em redes sem fio, nas quais ondas de rádio portando quadros podem se propagar para muito além do edifício que contém a(s) estação(ões) e os hospedeiros. Um bom lembrete que nos fez abrir os olhos para esse fato foram os 18 meses de uma 'operação de guerra' executada por Peter Shipley, que, percorrendo a Baía de San Francisco de carro, munido de um

laptop e de uma placa 802.11, procurou redes sem fio 'visíveis' fora dos edifícios em que estavam instaladas [Shipley, 2001]. Ele registrou mais de nove mil dessas redes; uma única esquina de San Francisco rendeu seis redes diferentes disponíveis! O que talvez seja ainda mais alarmante é que Shipley notou que 85 por cento dessas nove mil redes não utilizavam nem mesmo os mecanismos de segurança do padrão original de redes 802.11.

A questão da segurança em redes 802.11 atraiu considerável atenção, tanto em círculos técnicos quanto na mídia. Embora a discussão tenha sido considerável, o debate nem tanto — aparentemente todos concordam que a especificação original da 802.11 contém inúmeras falhas de segurança sérias. Na verdade, agora é possível descarregar software de domínio público que explora essas falhas, o que torna quem usa os débeis mecanismos de segurança inócuos da 802.11 tão sujeito a ataques abertos à segurança quanto os 85 por cento de usuários de redes sem fio da área da Baía de San Francisco que não usam absolutamente nenhuma medida de segurança.

A seguir, discutiremos os mecanismos de segurança padronizados na especificação inicial da 802.11, conhecidos, em conjunto, como **Privacidade Equivalente Sem Fio** (Wired Equivalent Privacy — WEP). Como o nome sugere, a intenção da WEP é oferecer um nível de segurança semelhante ao que encontramos em redes com fio. Discutiremos, então, algumas das falhas de segurança da WEP e o padrão 802.11i, uma versão fundamentalmente mais segura do que a 802.11, adotada em 2004.

Privacidade Equivalente Sem Fio (Wired Equivalent Privacy — WEP)

O protocolo WEP IEEE 802.11 [IEEE 802.11 1999] fornece autenticação e criptografia de dados entre um hospedeiro e um ponto de acesso sem fio (isto é, uma estação-base) utilizando uma abordagem de chaves simétricas compartilhadas. O WEP não especifica um algoritmo de gerenciamento de chaves, portanto, admitimos que o hospedeiro e o ponto de acesso sem fio concordaram com uma chave por meio de um método fora da banda. A autenticação é executada como no protocolo pa4.0 que desenvolvemos na Seção 8.3. O processo tem quatro etapas

1. Um hospedeiro sem fio requisita autenticação por um ponto de acesso.
2. O ponto de acesso responde à requisição de autenticação com um valor de nonce de 128 bytes.
3. O hospedeiro sem fio criptografa o nonce usando a chave simétrica que compartilha com o ponto de acesso.
4. O ponto de acesso decripta o nonce critografado pelo hospedeiro.

Se o nonce decriptado corresponder ao valor enviado originalmente pelo hospedeiro, então este será autenticado pelo ponto de acesso.

O algoritmo criptográfico WEP é ilustrado na Figura 8.35. Admite-se que ambos, um hospedeiro e o ponto de acesso, conhecem uma chave simétrica secreta de 40 bits, K_S. Além disso, um Vetor de Inicialização (Initialization Vector — IV) de 24 bits é anexado à chave de 40 bits, criando uma chave de 64 bits que será utilizada para criptografar um único quadro. O IV mudará de um quadro para outro e, por conseguinte, cada quadro será criptografado com uma chave de 64 bits diferente. A criptografia é efetuada como segue. Em primeiro lugar, é calculado um valor de CRC de 4 bytes (veja a Seção 5.2) para a carga útil. A carga e os 4 bytes do CRC então são criptografados usando o codificador seqüencial RC4. Não abordaremos os detalhes do RC4 aqui (veja [Schneier, 1995] se quiser mais detalhes). Para a nossa finalidade, basta saber que, quando recebe um valor de chave (nesse caso, a chave de 64 bits (K_S,IV)), o algoritmo RC4 produz uma seqüência de valores de chaves, $k_1^{IV}, k_2^{IV}, k_3^{IV}, \ldots$ que são utilizados para criptografar os dados e o valor de CRC em um quadro. Para sermos práticos, podemos imaginar que essas operações são realizadas um byte por vez. A criptografia é efetuada fazendo OU-exclusivo no i-ésimo byte de dados, d_i, com a i-ésima chave, k_i^{IV}, na corrente de valores de chaves gerada pelo par (K_S,IV) para produzir o i-ésimo byte do texto cifrado, c_i:

$$c_i = d_i \text{ XOR } k_i^{IV}$$

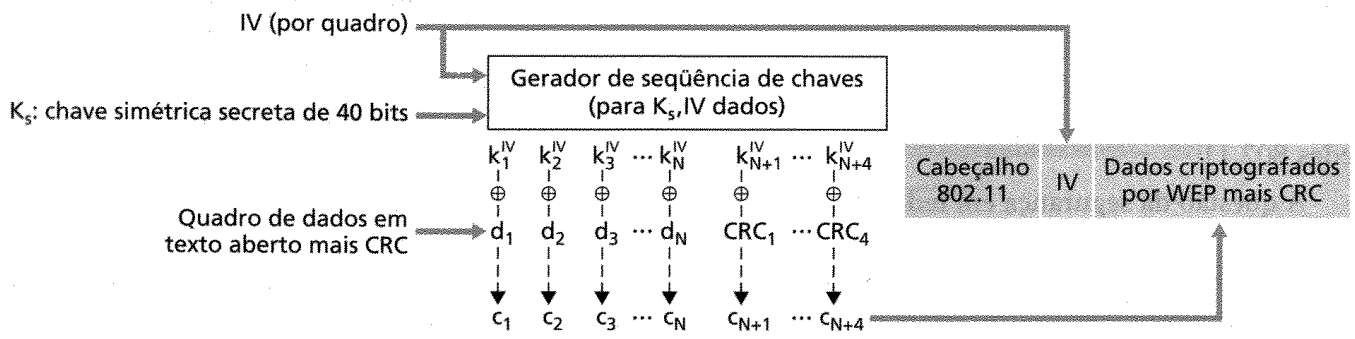


Figura 8.35 Protocolo WEP 802.11

O valor de IV muda de um quadro para o seguinte e está incluído em texto aberto no cabeçalho de cada quadro WEP 802.11 criptografado, como mostra a Figura 8.35. O receptor pega a chave simétrica secreta de 40 bits que compartilha com o emissor, anexa o IV e utiliza a chave de 64 bits resultante (que é idêntica à chave utilizada para efetuar a criptografia) para decriptar o quadro.

$$d_i = c_i \text{ XOR } k_i^{IV}$$

A utilização adequada do algoritmo RC4 requer que o mesmo valor de chave de 64 bits nunca seja utilizado mais do que uma vez. Lembre-se de que a chave WEP muda quadro a quadro. Para uma dada K_S (que raramente muda, se é que muda), isso significa que há somente 2^{24} chaves exclusivas. Se essas chaves forem escolhidas aleatoriamente, podemos demonstrar [Walker, 2000] que a probabilidade de ser escolhido o mesmo valor de IV (e, por conseguinte, a mesma chave de 64 bits) é mais de 99 por cento após apenas 12 mil quadros. Com quadros de 100 bytes de tamanho e taxa de transmissão de dados de 11 Mbps, bastam apenas alguns segundos para que os 12 mil quadros sejam transmitidos. Além do mais, visto que cada IV é transmitido em texto aberto, um bisbilhoteiro saberá sempre que for utilizado um valor duplicado de IV.

Para perceber um dos vários problemas que ocorrem quando é usada uma chave duplicada, considere o seguinte ataque a texto aberto escolhido realizado por Trudy contra Alice. Suponha que Trudy (possivelmente utilizando falsificação de IP) envie uma requisição (por exemplo, uma requisição HTTP ou FTP) a Alice, para transmitir um arquivo de conteúdo conhecido, $d_1, d_2, d_3, d_4, \ldots$. Trudy também observa os dados criptografados $c_1, c_2, c_3, c_4 \ldots$. Visto que $d_i = c_i \text{ XOR } k_i^{IV}$, se fizermos OU-exclusivo para c_i em cada lado dessa igualdade, teremos

$$d_i \text{ XOR } c_i = k_i^{IV}$$

Tendo essa relação em mãos, Trudy pode utilizar os valores conhecidos de d_i e c_i para calcular k_i^{IV}. Na próxima vez que Trudy vir o mesmo valor de IV sendo utilizado, ela ficará conhecendo a seqüência de chaves $k_1^{IV}, k_2^{IV}, k_3^{IV}, \ldots$ e, assim, poderá decriptar a mensagem criptografada.

Há também várias preocupações adicionais com o WEP. [Fluhrer, 2001] descreveu um ataque que explorava uma deficiência conhecida do RC4 quando são escolhidas certas chaves fracas. [Stubblefield, 2002] discute meios eficientes para implementar e explorar esse ataque. Uma outra preocupação com o WEP envolve os bits de CRC mostrados na Figura 8.35 e transmitidos no quadro 802.11 para detectar bits alterados na carga útil. Contudo, um atacante que modifique o conteúdo criptografado (por exemplo, substituindo os dados originais criptografados por texto sem nexo), calcule um CRC para o texto sem nexo e coloque esse CRC dentro de um quadro WEP, pode produzir um quadro 802.11 que será aceito pelo receptor. O que precisamos aqui são técnicas de integridade de mensagem tais como as estudadas na Seção 8.4 para detectar interferência ou substituição de conteúdo. Se quiser mais detalhes sobre segurança WEP, consulte [Walker, 2000; Weatherspoon, 2000; 802.11 Security, 2004] e as referências ali citadas.

IEEE802.11i

Logo após a liberação do IEEE 802.11, em 1999, foi iniciado o trabalho de desenvolvimento de uma versão nova e aprimorada do 802.11, com mecanismos de segurança mais robustos. O novo padrão, conhecido como 802.11i, estava na fase de ratificação e sua aprovação estava prevista para o início de 2004. Como veremos, enquanto o WEP oferecia criptografia relativamente fraca, somente um único modo de realizar autenticação e nenhum mecanismo de distribuição de chaves, o IEEE 802.11i fornece formas de criptografia muitos mais robustas, um conjunto extensível de mecanismos de autenticação e um mecanismo de distribuição de chaves. A seguir, apresentaremos uma visão geral do 802.11i; uma excelente exposição técnica sobre esse protocolo (em áudio em tempo real) pode ser encontrada em [TechOnline, 2004].

A Figura 8.36 dá um quadro geral da estrutura do 802.11i. Além do cliente sem fio e do ponto de acesso, o 802.11i. define um servidor de autenticação com o qual o AP pode se comunicar. A separação entre o servidor de autenticação e o AP permite que um único servidor de autenticação atenda a muitos APs, centralizando as decisões (muitas vezes delicadas) referentes à autenticação e ao acesso dentro de um servidor isolado e mantendo baixos os custos e a complexidade do AP. O funcionamento do 802.11i tem quatro fases:

1. *Descoberta.* Na fase de descoberta, o AP anuncia sua presença e as formas de autenticação e criptografia que podem ser oferecidas ao nó do cliente sem fio. O cliente, então, requisita as formas específicas de autenticação e criptografia que deseja. Embora o cliente e o AP já estejam trocando mensagens, o cliente ainda não foi autenticado, nem tem uma chave de criptografia; portanto, serão necessárias diversas outras etapas antes que o cliente possa se comunicar com um hospedeiro arbitrário pelo canal sem fio.
2. *Autenticação mútua e geração de Chave Mestra (Master Key — MK).* A autenticação ocorre entre o cliente sem fio e o servidor de autenticação. Nessa fase, o ponto de acesso age essencialmente como uma passagem, transmitindo mensagens entre o cliente e o servidor de autenticação. O **Protocolo Extensível de Autenticação** (Extensible Authentication Protocol — EAP) [RFC 2284] define os

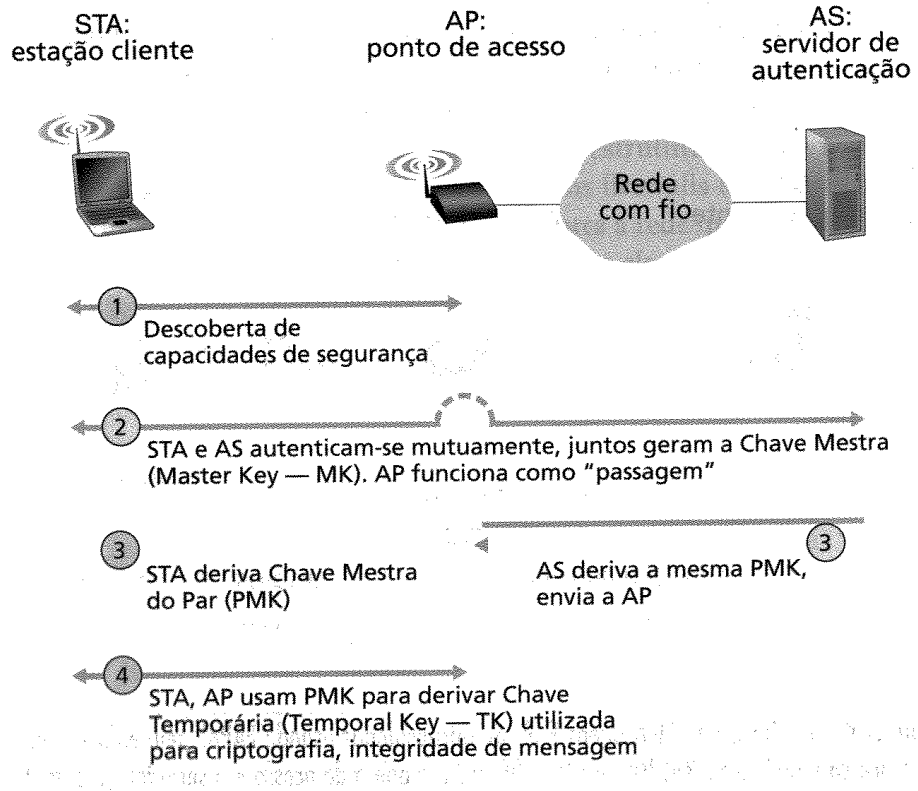


Figura 8.36 802.11i: quatro fases de operação

formatos de mensagens fim-a-fim utilizadas em um modo simples de interação requisição/resposta entre o cliente e o servidor de autenticação. Como mostra a Figura 8.37, mensagens EAPm são encapsuladas utilizando EAPoL (EAP por LAN, [IEEE 802.1X]) e enviadas pelo enlace sem fio 802.11. Essas mensagens EAP são desencapsuladas no ponto de acesso e então reencapsuladas utilizando o protocolo **RADIUS** para transmissão ao servidor de autenticação por UDP/IP. Conquanto o servidor e o protocolo RADIUS [RFC 2138] não sejam exigidos pelo protocolo 802.11i, na prática eles são componentes de fato do padrão do 802.11i. O protocolo **DIAMETER** recentemente padronizado [RFC 3588] provavelmente substituirá o RADIUS em futuro próximo.

Com EAP, o servidor de autenticação pode escolher um entre vários modos de efetuar autenticação. Conquando o 802.11i não imponha nenhum método particular de autenticação, o esquema de autenticação EAP-TLS [RFC 2716] é muito usado. O EAP-TLS utiliza técnicas de chaves públicas (incluindo criptografia de nonce e resumos de mensagens) semelhantes às estudadas nas seções 8.3, 8.4 e 8.5, para permitir que o cliente e o servidor de autenticação se autentiquem mutuamente e para derivar uma Chave Mestra (Master Key — MK) conhecida por ambos os participantes.

3. *Geração de Chave Mestra de Par (Pairwise Master Key — PMK Generation).* A MK é um segredo compartilhado que somente o cliente e o servidor de autenticação conhecem e que eles usam para gerar uma segunda chave, a Chave Mestra de Par (*Pairwise Master Key* — MK). O servidor de autenticação então envia a PMK ao AP. E é aqui que queríamos chegar! O cliente e o AP agora têm uma chave compartilhada (lembre-se de que o problema de distribuição de chaves não era abordado no WEP) e autenticaram-se mutuamente. Agora já estão quase prontos para começar a operar na rede.
4. *Geração de Chave Temporária (Temporal Key — TK Generation).* Com a PMK, o cliente sem fio e o AP agora podem gerar chaves adicionais que serão usadas para comunicação. A Chave Temporária (TK) é de particular interesse e será utilizada para realizar a criptografia na camada de enlace de dados enviados pelo enlace sem fio a um hospedeiro remoto arbitrário.

O 802.11i fornece várias formas de criptografia, incluindo um esquema criptográfico baseado em AES e uma versão reforçada da criptografia WEP.

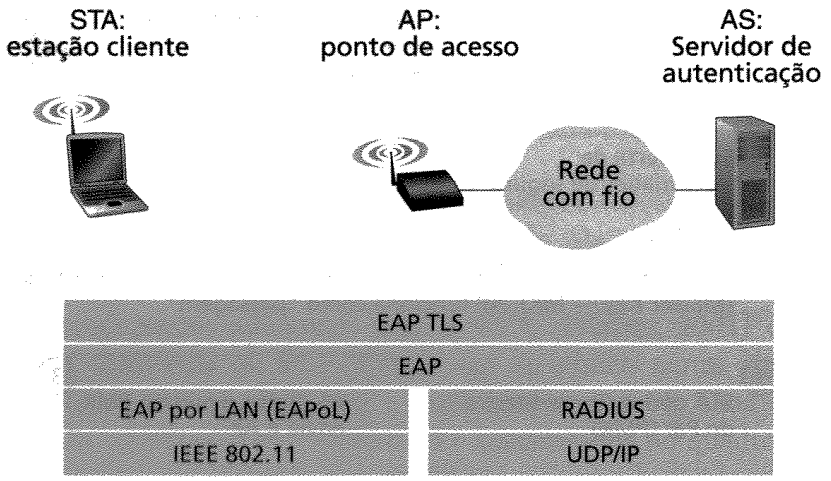


Figura 8.37 EAP é um protocolo fim-a-fim. Mensagens EAP são encapsuladas usando EAPoL pelo enlace sem fio entre o cliente e o ponto de acesso e utilizando RADIUS por UDP/IP entre o ponto de acesso e o servidor de autenticação

8.9 Resumo

Neste capítulo, examinamos os diversos mecanismos que Bob e Alice, os amantes secretos, podem usar para se comunicar com segurança. Vimos que Bob e Alice estão interessados em confidencialidade (para que somente eles possam entender o conteúdo de uma mensagem transmitida), autenticação (para que tenham a certeza de que estão falando um com o outro) e integridade de mensagem (para que tenham certeza de que suas mensagens não sejam alteradas em trânsito). É claro que a necessidade de comunicação segura não está limitada a amantes secretos. Na verdade, vimos nas seções 8.6, 8.7 e 8.8 que a segurança é necessária em várias camadas de uma arquitetura de rede para proteção contra bandidos que têm à mão um grande arsenal de ataques possíveis.

Na primeira parte deste capítulo, apresentamos vários princípios subjacentes à comunicação segura. Na Seção 8.2, examinamos as técnicas criptográficas para criptografar e decriptar dados, incluindo criptografia de chaves simétricas e criptografia de chaves públicas. O DES e o RSA foram examinados como estudos de caso específicos dessas duas classes mais importantes de técnicas de criptografia em uso nas redes de hoje. Na Seção 8.3, voltamos nossa atenção à autenticação e desenvolvemos uma série de protocolos de sofisticação crescente, para garantir que o interlocutor é, de fato, quem diz ser e para assegurar que ele está ao vivo. Vimos que a criptografia de chaves simétricas e a criptografia de chaves públicas podem desempenhar um papel importante não apenas no disfarce dos dados (criptografia/decriptação), mas também no serviço de autenticação. Técnicas para assinar um documento digital de modo que ele seja verificável, não falsificável e não repudiável foram abordadas na Seção 8.4. Mais uma vez, a aplicação de técnicas criptográficas demonstrou ser essencial. Examinamos assinaturas digitais e resumos de mensagem, sendo este último um modo abreviado de assinar um documento digital. Na Seção 8.5, analisamos protocolos de distribuição de chaves. Vimos que, para a criptografia de chaves simétricas, uma central de distribuição de chaves (KDC) — uma entidade de rede de confiança e independente — pode ser usada para distribuir uma chave simétrica compartilhada entre partes comunicantes. Para a criptografia de chaves públicas, uma CA distribui certificados de validação de chaves públicas.

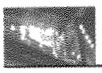
Munidos das técnicas abordadas nas seções 8.2, 8.3, 8.4 e 8.5, Bob e Alice podem se comunicar com segurança (esperamos que eles sejam estudantes de rede que aprenderam o que este livro ensinou e possam, dessa maneira, evitar que suas travessuras sejam descobertas por Trudy!) Porém, cada vez mais a confiabilidade é apenas uma pequena parte do quadro da segurança na rede. O foco da segurança em redes está se concentrando cada vez mais em garantir a segurança da infra-estrutura da rede contra o ataque potencial dos bandidos. Assim, na última parte deste capítulo, estudamos firewalls (que regulam o acesso a uma rede protegida e de uma rede protegida) e uma série de ataques contra a rede que podem ser montados por intrusos, bem como as contramedidas que podem ser tomadas contra eles. Concluímos o capítulo com estudos de casos que aplicam técnicas apresentadas nas seções 8.2 a 8.5 na camada de aplicação, na camada de transporte, na camada de rede e na camada de enlace.

Exercícios de fixação

Capítulo 8 Questões de revisão

1. Quais são as diferenças entre confidencialidade de mensagem e integridade de mensagem? É possível ter uma sem a outra? Justifique sua resposta.
2. Qual é a diferença entre um intruso ativo e um intruso passivo?
3. Qual seria uma diferença importante entre um sistema de chaves simétricas e um sistema de chaves públicas?

4. Suponha que um intruso tenha uma mensagem criptografada, bem como a versão decriptada dessa mensagem. Ele pode montar um ataque somente com texto cifrado, um ataque com texto aberto conhecido ou um ataque com texto aberto escolhido?
5. Suponha que N pessoas queiram se comunicar com cada uma das outras $N - 1$ pessoas usando criptografia de chaves simétricas. Todas as comunicações entre quaisquer duas pessoas, i e j, são visíveis para todas as outras pessoas desse grupo de N, e nenhuma outra pessoa desse grupo pode decodificar suas comunicações. O sistema, como um todo, requer quantas chaves? Agora, suponha que seja usada criptografia de chaves públicas. Quantas chaves serão necessárias nesse caso?
6. Qual é a finalidade de um nonce em um protocolo de autenticação?
7. O que significa dizer que um nonce é um valor que só ocorre uma vez na vida? Na vida de quem?
8. O que é o ataque do homem do meio? Esse ataque pode ocorrer quando são usadas chaves simétricas?
9. O que significa afirmar que um documento é verificável, não falsificável e não repudiável?
10. De que maneira um resumo de mensagem proporciona uma verificação de integridade de mensagem melhor do que uma soma de verificação, tal como a soma de verificação da Internet?
11. De que modo um resumo de mensagem criptografado por chave pública proporciona uma assinatura digital melhor do que utilizar a mensagem criptografada com chave pública?
12. A mensagem associada com um resumo de mensagem está criptografada? Justifique sua resposta.
13. O que é uma central de distribuição de chaves? E uma autoridade certificadora?
14. Resuma as principais diferenças entre os serviços fornecidos pelos protocolos de autenticação de cabeçalho (AH) e de segurança do encapsulamento de carga útil (ESP) em IPsec.



Problemas

1. Usando a cifra monoalfabética da Figura 8.3, codifique a mensagem "`This is an easy problem.`" Decodifique a mensagem "`rmij'u uamu xyj.`"
2. Mostre que o ataque com texto aberto conhecido de Trudy em que ela conhece os pares de tradução (texto cifrado, texto aberto) para sete letras reduz em aproximadamente 10^9 o número de possíveis substituições a verificar no exemplo apresentado na Seção 8.2.1.
3. Considere o sistema polialfabético mostrado na Figura 8.4. Um ataque com texto aberto escolhido que consiga obter a codificação da mensagem "`The quick fox jumps over the lazy brown dog`" é suficiente para decifrar todas as mensagens? Explique sua resposta.
4. Usando RSA, escolha $p = 3$ e $q = 11$ e codifique a palavra 'hello'. Aplique o algoritmo de decriptação à versão criptografada para recuperar a mensagem original em texto aberto.
5. Considere nosso protocolo de autenticação *ap4.0* no qual Alice se autentica para Bob, e que vimos que funciona bem (isto é, não encontramos nenhuma falha nele). Agora suponha que, enquanto Alice está se autenticando para Bob, este deve se autenticar para Alice. Apresente um cenário no qual Trudy, fazendo-se passar por Alice, agora pode se autenticar para Bob como se fosse Alice. (*Dica*: considere que a seqüência de operações de *ap4.0*, uma com Trudy iniciando a seqüência e outra com Bob iniciando a seqüência, pode ser intercalada arbitrariamente. Preste particular atenção ao fato de que Bob e Alice usarão um nonce e que, caso não se tome cuidado, o mesmo nonce pode ser utilizado com más intenções.)
6. No ataque do homem do meio mostrado na Figura 8.13, Alice não autenticou Bob. Se Alice tivesse solicitado a Bob que se autenticasse usando *ap5.0*, o ataque do homem do meio teria sido evitado? Exponha seu raciocínio.

7. O protocolo de roteamento da Internet, BGP, usa o resumo de mensagem MD5, em vez de criptografia de chaves públicas, para assinar mensagens BGP. Na sua opinião, por que o MD5 foi escolhido em vez da criptografia de chaves públicas?
8. Calcule uma terceira mensagem, diferente das duas mensagens na Figura 8.18, que tenha a mesma soma de verificação dessas duas mensagens na figura citada.
9. No protocolo e discussão da Figura 8.19, por que Alice não tem de autenticar Bob explicitamente?
10. Por que não há autenticação explícita no protocolo da Figura 8.19? A autenticação é necessária? Por quê?
11. Considere os servidores KDC e CA. Suponha que a KDC caia. Que impacto isso causaria sobre a capacidade de os participantes se comunicarem com segurança, isto é, quem pode e quem não pode se comunicar? Justifique sua resposta. Suponha agora que uma CA caia. Que impacto essa falha causaria?
12. Considere a seguinte variação do firewall de filtragem de pacotes discutido na Seção 8.6. Suponha que Alice quer desautorizar acesso à sua rede 222.22.0.0/16 a partir da Internet pública (regra R3 da primeira tabela abaixo). Alice está novamente colaborando com Bob e seus colegas que estão na universidade e, portanto, ela quer permitir que usuários da universidade de Bob (cujo endereço de rede é 111.11/16) acessem uma sub-rede específica, 222.22.22/24, dentro da rede da empresa onde ela trabalha (regra R1 abaixo). Alice sabe que Trudy, uma hacker notória, está na universidade de Bob e que o endereço de sua sub-rede, 111.11.11/24, é um porto inseguro repleto de hackers. Portanto, Alice não quer que qualquer tráfego que venha de 111.11.11/24 entre em nenhum lugar de sua rede (regra R2) exceto na sub-rede especial 222.22.22/24 (permitida sob a regra R1; esta exceção é a diferença importante em relação ao nosso exemplo na Seção 8.6). As regras de filtragem de pacote de Alice estão resumidas na tabela a seguir:

Regra	Endereço da fonte	Endereço de destino	Ação	Comentários
R1	111.11/16	222.22.22/24	permitir	Permite a entrada de datagramas que vêm da rede da universidade de Bob (incluindo a sub-rede de hackers de Trudy) em uma sub-rede restrita.
R2	111.11.11/24	222.22/16	negar	Não permite que tráfego da sub-rede de Trudy entre na rede de Alice; mas veja R1.
R3	0.0.0.0/0	0.0.0.0/0	negar	Não permite a entrada de tráfego na rede de Alice.

Preencha, na tabela abaixo, as ações executadas para esse cenário sob as regras na ordem R1, R2 e R3, e sob as regras na ordem R2, R1, R3.

Que resultado a anulação da regra R2 teria para os pacotes P1, P2, P3 e P4?

Número do Datagrama	Endereço IP de Fonte	Endereço IP de Destino	Ação desejada	Ação sob R2, R1, R3	Ação sob R1, R2, R3
P1	111.11.11.1 (sub-rede de hackers)	222.22.6.6 (rede da empresa)	negar		
P2	111.11.11.1 (sub-rede de hackers)	222.22.22.2 (sub-rede especial)	permitir		
P3	111.11.6.6 (rede da universidade, exceto a sub-rede de hackers)	222.22.22.2 (sub-rede especial)	permitir		
P4	111.11.6.6 (rede da universidade, exceto a sub-rede de hackers)	222.22.6.6 (rede da empresa)	negar		

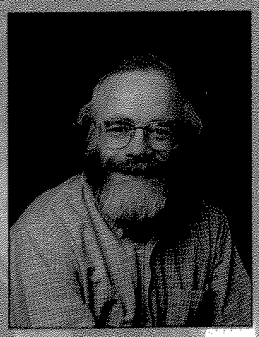
13. A Figura 8.29 mostra as operações que Alice deve realizar para fornecer confidencialidade, autenticação e integridade. Monte um diagrama demonstrando as operações correspondentes que Bob deve realizar sobre o pacote recebido de Alice.

Questões dissertativas

1. Suponha que um intruso consiga inserir e remover mensagens DNS da rede. Monte três cenários demonstrando os problemas que esse intruso poderia causar.
2. Ninguém ainda provou formalmente que o DES triplo ou o RSA são seguros. Assim, que evidência temos de que eles são, de fato, seguros?
3. Se o IPsec oferece segurança na camada de rede, por que ainda são necessários mecanismos de segurança em camadas acima do IP?
4. Consulte a home page internacional do PGP (`http://www.pgpi.org/`). Que versão do PGP você pode descarregar legalmente, informando o país em que está?

Entrevista

Steven M. Bellovin



Steven M. Bellovin é um membro AT&T (AT&T Fellow) do Network Services Research Lab nos AT&T Labs Research, em Florham Park, New Jersey. Ele trabalha especificamente com redes e segurança, e com as causas que as tornam incompatíveis. Em 1995, Steven recebeu o Usenix Lifetime Achievement Award por seu trabalho na criação da Usenet, a primeira rede de troca de informações que ligava dois ou mais computadores e permitia que os usuários compartilhassem informações e discutissem em conjunto. Steve também foi eleito membro da National Academy of Engineering. Fez mestrado na Columbia University e doutorado na University of North Carolina, em Chapel Hill.

O que o fez se decidir pela especialização na área de segurança de redes?

O que vou dizer parecerá estranho, mas a resposta é simples. Estudei programação de sistemas e administração de sistemas, o que levou naturalmente à segurança. E sempre me interessei por comunicações, desde a época em que trabalhava em tempo parcial em programação de sistemas, quando ainda estava na universidade.

Meu trabalho na área de segurança continua a ser motivado por duas coisas — um desejo de manter a utilidade dos computadores, o que significa impedir que sua função seja corrompida por atacantes, e um desejo de proteger a privacidade.

Qual era sua visão sobre a Usenet na época em que o senhor a estava desenvolvendo? Qual é sua visão agora?

No início considerávamos a Usenet como um meio para discutir ciência da computação e programação de computadores em todo o país e para utilizar em questões administrativas locais, como recurso de vendas, e assim por diante. Na verdade, minha previsão original era de uma ou duas mensagens por dia, de 50 a 100 sites no máximo. Mas o crescimento real ocorreu em tópicos relacionados a pessoas, incluindo — mas não limitado a — interações de seres humanos com computadores. Meus grupos de bate-papo preferidos foram, durante muitos anos, coisas como rec.woodworking (marcenaria), bem como sci.crypt (criptografia).

Até certo ponto, as redes de notícias foram desbancadas pela Web. Se eu fosse iniciar o projeto hoje, ele seria muito diferente. Mas ainda é um excelente meio para alcançar um público muito amplo interessado no assunto, sem ter de passar por sites Web particulares.

Quais pessoas o inspiraram profissionalmente? De que modo?

O professor Fred Brooks — fundador e primeiro chefe do departamento de ciência da computação da University of North Carolina, em Chapel Hill, gerente da equipe que desenvolveu o IBM S/360 e o OS/360, e autor do livro *The Mythical Man-Month* — exerceu uma tremenda influência sobre minha carreira. Acima de tudo, ele me ensinou observação e compromissos — como considerar problemas no contexto do mundo real (e como o mundo real era muito mais confuso do que qualquer teórico gostaria que fosse), e como equilibrar interesses conflitantes ao elaborar uma solução. Grande parte do trabalho com computadores é engenharia — a arte de fazer os compromissos certos de modo a satisfazer muitos objetivos contraditórios

Em sua opinião, qual é o futuro das redes e da segurança?

Até agora, grande parte da segurança que temos vem do isolamento. Um firewall, por exemplo, funciona impedindo o acesso a certas máquinas e serviços. Mas vivemos em uma época na qual a conectividade é cada vez maior — ficou mais difícil isolar coisas. Pior ainda, nossos sistemas de produção requerem um número muito maior de componentes isolados, interconectados por redes. Garantir a segurança de tudo isso é um de nossos maiores desafios.

Em que projetos especiais o senhor está trabalhando agora?

Estou me concentrando muito em questões operacionais. Não basta que um sistema seja seguro; ele também tem de ser utilizável. Isso significa, por sua vez, que operadores de redes têm de poder monitorar coisas. Mas monitorar pode conflitar com segurança — a segurança tende a ocultar e proteger computadores e informações, mas operadores precisam ver certas coisas. Além disso, sistemas têm de continuar funcionando mesmo que componentes cruciais falhem. Como compartilhar informações com os interessados corretos sem permitir a entrada dos bandidos? Essa é a pergunta que estou tentando responder agora.

Na sua opinião, têm havido grandes avanços na área de segurança? Até onde teremos de ir?

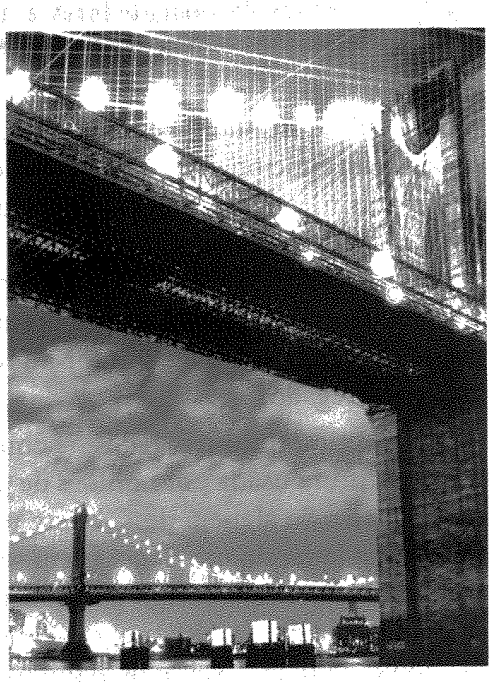
Ao menos do ponto de vista científico, sabemos como fazer criptografia. E isso é uma grande ajuda. Mas a maioria dos problemas de segurança se deve a códigos defeituosos e este é um problema muito mais sério. Na verdade, é o problema mais antigo da ciência da computação que ainda não foi resolvido — e acho que continuará sendo. O desafio é descobrir como manter a segurança de sistemas quando temos de construí-los com componentes inseguros. Hoje já podemos fazer isso no que tange às falhas de hardware; mas podemos fazer o mesmo em relação à segurança?

O senhor pode dar algum conselho aos estudantes sobre a Internet e segurança em redes?

Aprender os mecanismos é a parte fácil. Aprender a "pensar como um paranóico" é mais difícil. Você tem de lembrar que as distribuições de probabilidade não se aplicam — os atacantes podem encontrar e encontrarão condições improváveis. E os detalhes são importantes — e muito!

Capítulo 9

Gerenciamento de Rede



Após termos percorrido nosso caminho pelos oito primeiros capítulos deste livro, estamos agora conscientes de que uma rede consiste em muitas peças complexas de hardware e software que interagem umas com as outras — desde os enlaces, comutadores, roteadores, hospedeiros e outros dispositivos, que são os componentes físicos da rede, até os muitos protocolos (tanto em hardware quanto em software) que controlam e coordenam esses componentes. Quando centenas ou milhares desses componentes são montados em conjunto por alguma organização para formar uma rede, não é nada surpreendente que ocasionalmente eles apresentem defeitos, que elementos da rede sejam mal configurados, que recursos da rede sejam utilizados excessivamente ou que componentes da rede simplesmente 'quebrem' (por exemplo, um cabo pode ser cortado, uma latinha de refrigerante pode ser derramada sobre um roteador). O administrador de rede, cuja tarefa é mantê-la 'viva e atuante', deve estar habilitado a reagir a esses contratempos (e, melhor ainda, a evitá-los). Com potencialmente milhares de componentes de rede espalhados por uma grande área, ele, em sua central de operações (*network operations center* — NOC), evidentemente necessita de ferramentas que o auxiliem a monitorar, administrar e controlar a rede. Neste capítulo, examinaremos a arquitetura, protocolos e bases de informação que um administrador de rede utiliza para realizar seu trabalho.

9.1 O que é gerenciamento de rede?

Antes de discutir o gerenciamento de redes em si, vamos considerar alguns cenários ilustrativos do 'mundo real' que não são de redes, mas nos quais um sistema complexo com muitos componentes em interação deve ser monitorado, gerenciado e controlado por um administrador. As usinas de geração de energia elétrica (ao menos como são retratadas na mídia popular em filmes como *Síndrome da China*) têm uma sala de controle, onde mostradores, medidores e lâmpadas monitoram o estado (temperatura, pressão, vazão) de válvulas, tubulações, vasos e outros componentes remotos da instalação industrial. Esses dispositivos permitem que o operador monitore os muitos componentes da planta e podem alertá-lo (o famoso pisca-pisca vermelho de alerta) caso haja algum problema iminente. O operador responsável executa certas ações para controlar esses componentes. De maneira semelhante, a cabine de um avião é equipada com instrumentos para que o piloto possa monitorar e controlar os muitos componentes de uma aeronave. Nesses dois exem-

plos, o 'administrador' *monitora* equipamentos remotos e *analisa* os dados para garantir que os equipamentos estejam funcionando e operando dentro dos limites especificados (por exemplo, que a fusão do núcleo de uma usina nuclear não esteja na iminência de acontecer ou que o combustível do avião não esteja prestes a acabar), *controla reativamente* o sistema fazendo ajustes de acordo com as modificações ocorridas no sistema ou em seu ambiente e *gerencia pró-ativamente* o sistema (por exemplo, detectando tendências ou comportamentos anômalos que permitem executar uma ação antes que surjam problemas sérios). Nesse mesmo sentido, o administrador de rede vai monitorar, gerenciar e controlar ativamente o sistema do qual está encarregado.

Nos primórdios das redes de computadores, quando elas ainda eram artefatos de pesquisa, e não uma infra-estrutura usada por milhões de pessoas por dia, 'gerenciamento de rede' era algo de que nunca se tinha ouvido falar. Se alguém descobrisse um problema na rede, poderia realizar alguns testes, como o *ping*, para localizar a fonte do problema e, em seguida, modificar os ajustes do sistema, reiniciar o software ou o hardware ou chamar um colega para fazer isso. (O RFC 789 traz uma discussão de fácil leitura sobre a primeira grande 'queda' da ARPAnet em 27 de outubro de 1980, muito antes da existência de ferramentas de gerenciamento de rede, e sobre os esforços realizados para entender os motivos dessa queda e recuperar a rede.) Como a Internet pública e as intranets privadas cresceram e se transformaram de pequenas redes em grandes infra-estruturas globais, a necessidade de gerenciar mais sistematicamente a enorme quantidade de componentes de hardware e software dentro dessas redes também se tornou mais importante.

Com o intuito de motivar nosso estudo de gerenciamento de redes, vamos começar com um exemplo simples. A Figura 9.1 ilustra uma pequena rede constituída de três roteadores e alguns hospedeiros e servidores. Mesmo para uma rede tão simples, há muitos cenários em que o administrador de rede muito se beneficiará por ter à mão as ferramentas de gerenciamento adequadas:

- *Detecção de falha em uma placa de interface em um hospedeiro ou roteador.* Com ferramentas de gerenciamento apropriadas, uma entidade de rede (por exemplo, o roteador A) pode indicar ao administrador de rede que uma de suas interfaces não está funcionando. (Isso certamente é melhor do que receber um telefonema, no NOC (centro de operações), de um usuário zangado dizendo que a conexão com a rede caiu!) Um administrador de rede que monitora e analisa de maneira ativa o tráfego da rede pode até impressionar o usuário (aquele, o zangado) detectando problemas na interface bem antes e substituindo a placa de interface antes que ela caia. Isso poderá ser feito, por exemplo, se o administrador notar um aumento de erros de somas de verificação em quadros que estão sendo enviados por uma placa de interface que está prestes a falhar.
- *Monitoração de hospedeiro.* O administrador de rede pode verificar periodicamente se todos os hospedeiros da rede estão ativos e operacionais. Mais uma vez, ele pode, de fato, impressionar um usuário da rede, reagindo pró-ativamente a um problema (falha em um hospedeiro) antes de o defeito ser apontado pelo usuário.
- *Monitoração de tráfego para auxiliar o oferecimento de recursos.* Um administrador de rede pode monitorar padrões de tráfego entre fontes e destinos e notar, por exemplo, que, comutando servidores entre segmentos de LAN, o total de tráfego que passa por várias LANs poderia ser reduzido de maneira significativa. Imagine a felicidade geral (especialmente na alta administração) com um desempenho melhor sem o custo de novos equipamentos. De modo similar, monitorando a utilização de um enlace, um administrador de rede pode determinar que um segmento de LAN ou o enlace com o mundo externo está sobrecarregado e que um enlace de maior largura de banda deve ser providenciado (ainda que com um custo mais alto). Ele também poderia ser alertado automaticamente quando o nível de congestionamento de um enlace ultrapassasse determinado limite, para providenciar um enlace de maior largura de banda antes que o congestionamento se tornasse sério.

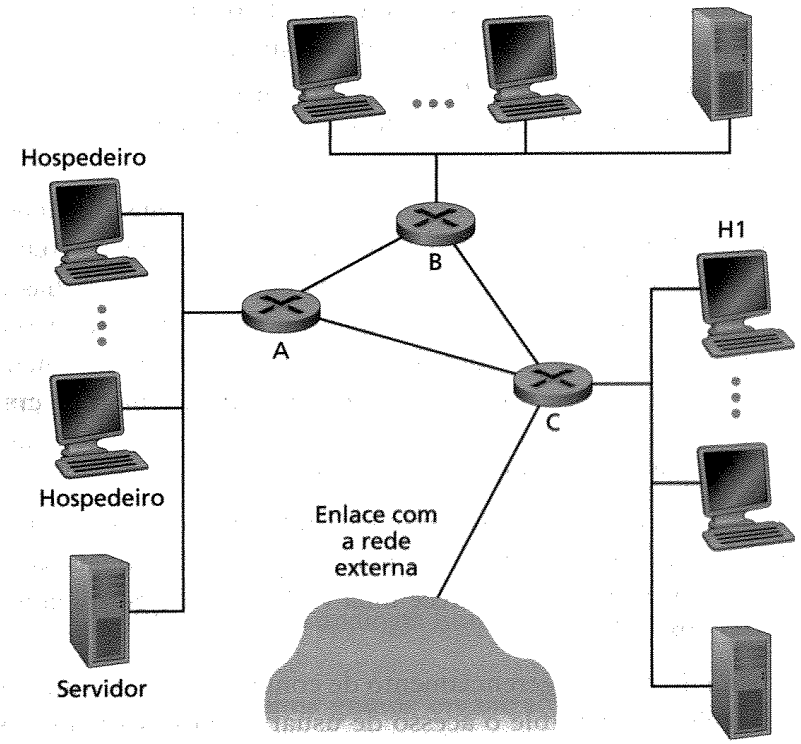


Figura 9.1 Um cenário simples que ilustra a utilização do gerenciamento de rede

- *Detecção de mudanças rápidas em tabelas de roteamento.* A alternância de rotas — mudanças freqüentes em tabelas de roteamento — pode indicar instabilidades no roteamento ou um roteador mal configurado. Evidentemente, um administrador de rede que configurou um roteador de modo inapropriado prefere ele mesmo descobrir o erro antes que a rede caia.
- *Monitoração de SLAs.* Com o advento dos **Acordos de Nível de Serviços** (Service Level Agreements — SLAs) — contratos que definem parâmetros específicos de medida e níveis aceitáveis de desempenho do provedor de rede em relação a essas medidas —, o interesse na monitoração do tráfego cresceu significativamente nos últimos anos [Larsen, 1997; Huston, 1999a]. MCI e Sprint são apenas dois dos muitos provedores de rede que garantem SLAs [MCI, 2004; Sprint, 2004] a seus usuários. Alguns desses SLAs são: disponibilidade de serviço (interrupção de serviços), latência, vazão e requisitos para notificação da ocorrência de serviço interrompido. É claro que, se um contrato especificar critérios de desempenho de prestação de serviços entre um provedor de rede e seus usuários, então a medição e o gerenciamento do desempenho do sistema também serão de grande importância para o administrador de rede.
- *Detecção de intrusos.* Um administrador de rede provavelmente vai querer ser avisado quando chegar tráfego de uma fonte suspeita ou quando se destinar tráfego a ela (por exemplo, hospedeiro ou número de porta). Da mesma maneira, ele pode querer detectar (e, em muitos casos, filtrar) a existência de certos tipos de tráfego (por exemplo, pacotes roteados pela fonte ou um grande número de pacotes SYN dirigidos a um dado hospedeiro) que são característicos de determinados tipos de ataque à segurança que consideramos no Capítulo 8.

A International Organization for Standardization (ISO) criou um modelo de gerenciamento de rede que é útil para situar os cenários apresentados em um quadro mais estruturado. São definidas cinco áreas de gerenciamento de rede:

- *Gerenciamento de desempenho.* A meta do gerenciamento de desempenho é quantificar, medir, informar, analisar e controlar o desempenho (por exemplo, utilização e vazão) de diferentes com-

ponentes da rede. Entre esses componentes estão dispositivos individuais (por exemplo, enlaces, roteadores e hospedeiros), bem como abstrações fim-a-fim, como um trajeto pela rede. Veremos, em breve, que padrões de protocolo como o SNMP (Simple Network Management Protocol — Protocolo Simples de Gerenciamento de Rede) [RFC 3410] desempenham um papel fundamental no gerenciamento de desempenho da Internet.

- *Gerenciamento de falhas.* O objetivo do gerenciamento de falhas é registrar, detectar e reagir às condições de falha da rede. A linha divisória entre gerenciamento de falha e gerenciamento de desempenho é bastante indefinida. Podemos considerar o gerenciamento de falhas como o tratamento imediato de falhas transitórias da rede (por exemplo, interrupção de serviço em enlaces, hospedeiros, ou em hardware e software de roteadores), enquanto o gerenciamento de desempenho adota uma abordagem de longo prazo em relação ao desempenho da rede em face de demandas variáveis de tráfego e falhas ocasionais na rede. Como acontece no gerenciamento de desempenho, o SNMP tem um papel fundamental no gerenciamento de falhas.
- *Gerenciamento de configuração.* O gerenciamento de configuração permite que um administrador de rede saiba quais dispositivos fazem parte da rede administrada e quais são suas configurações de hardware e software. O [RFC 3139] oferece uma visão geral de gerenciamento e requisitos de configuração para redes IP.
- *Gerenciamento de contabilização.* O gerenciamento de contabilização permite que o administrador da rede especifique, registre e controle o acesso de usuários e dispositivos aos recursos da rede. Quotas de utilização, cobrança por utilização e alocação de acesso privilegiado a recursos fazem parte do gerenciamento de contabilização.
- *Gerenciamento de segurança.* A meta do gerenciamento de segurança é controlar o acesso aos recursos da rede de acordo com alguma política definida. As centrais de distribuição de chaves e as autoridades certificadoras que estudamos na Seção 7.5 são componentes do gerenciamento de segurança. O uso de firewalls para monitorar e controlar pontos externos de acesso à rede, um tópico que estudamos na Seção 8.6, é outro componente crucial.

Neste capítulo, abordaremos apenas os rudimentos do gerenciamento de rede. Nosso foco é propositalmente limitado — examinaremos somente a infra-estrutura do gerenciamento de rede: a arquitetura geral, os protocolos de gerenciamento de rede e a base de informações que servem para um administrador de rede mantê-la em pé e em funcionamento. Não abordaremos os processos de tomada de decisão do administrador de rede, que é quem deve planejar, analisar e reagir às informações gerenciais dirigidas à NOC. Nessa área estão alguns tópicos, como identificação e gerenciamento de falhas [Katzela, 1995; Medhi, 1997; Steinder, 2002], detecção pró-ativa de anomalias [Thottan, 1998], correlação entre alarmes [Jakobson, 1993] e outros mais. Tampouco abordaremos o tópico mais amplo do gerenciamento de serviços [Saydam, 1996; RFC 3053; AT&T SLM, 2004] — o fornecimento de recursos, como largura de banda, capacidade do servidor e outros recursos computacionais e de comunicação necessários para cumprir os requisitos de serviço específicos da missão de uma empresa. Nessa última área, padrões como o TMN [Glitho, 1995; Sidor, 1998] e o TINA [Hamada, 1997], que são mais amplos, mais abrangentes (e alguns consideravelmente muito mais pesados), abordam essa questão mais geral. O TINA, por exemplo, é descrito como "um conjunto de metas, princípios e conceitos comuns que abrangem o gerenciamento de serviços, de recursos e de partes do Ambiente de Processamento Distribuído" [Hamada, 1997]. É claro que cada um desses tópicos é material suficiente para um outro livro e nos afastaria muito dos aspectos mais técnicos das redes de computadores. Assim, como dissemos anteriormente, nosso objetivo mais modesto é examinar os detalhes importantes da infra-estrutura do gerenciamento de redes, por meio da qual um administrador de rede mantém os bits fluindo com suavidade.

"O que é gerenciamento de rede?" Essa pergunta é feita com freqüência. Nossa discussão anterior expôs e ilustrou a necessidade de alguns usos do gerenciamento de rede. Concluiremos esta seção com uma defini-

ção de gerenciamento de rede em uma única sentença (embora um tanto longa e encadeada), dada por [Saydam, 1996]:

"Gerenciamento de rede inclui o oferecimento, a integração e a coordenação de elementos de hardware, software e humanos, para monitorar, testar, consultar, configurar, analisar, avaliar e controlar os recursos da rede, e de elementos, para satisfazer às exigências operacionais, de desempenho e de qualidade de serviço em tempo real a um custo razoável."

É uma sentença de perder o fôlego, mas é uma definição viável. Nas seções seguintes, acrescentaremos um pouco de recheio a essa definição bastante enxuta de gerenciamento de rede.

9.2 A infra-estrutura do gerenciamento de rede

Vimos, na seção anterior, que o gerenciamento de rede exige a capacidade de "monitorar, testar, consultar, configurar (...) e controlar" os componentes de hardware e software de uma rede. Como os dispositivos da rede são distribuídos, fazer isso exigirá, no mínimo, que o administrador possa coletar dados (por exemplo, para a finalidade de monitoração) de uma entidade remota e efetuar mudanças nessa entidade (por exemplo, controle). Nesta seção, uma analogia humana será útil para entender a infra-estrutura necessária para gerenciamento de rede.

Imagine que você seja o chefe de uma grande organização que tem filiais em todo o mundo. É tarefa sua assegurar que as diversas partes de sua organização operem sem percalços. Como você o fará? No mínimo, você coletará dados periodicamente de suas filiais por meio de relatórios e de várias medições quantitativas de atividade, produtividade e orçamento. De vez em quando (mas não sempre), você será explicitamente notificado da existência de um problema em uma das filiais; o gerente da filial que quer galgar a escada corporativa (e talvez ficar com seu cargo) pode enviar relatórios não solicitados mostrando que as coisas estão correndo muito bem na filial a seu cargo. Você examinará os relatórios que receber esperando encontrar operações regulares em todos os lugares, mas, sem dúvida, encontrará problemas que precisarão de sua atenção. Você poderá iniciar um diálogo pessoal com uma de suas filiais problemáticas, coletar mais dados para entender o problema e, então, passar uma ordem executiva ("Faça essa mudança!") ao gerente da filial.

Implícita nesse cenário humano muito comum está uma infra-estrutura para controlar a organização — o chefe (você), os locais remotos que estão sendo controlados (as filiais), seus agentes remotos (os gerentes das filiais), protocolos de comunicação (para transmitir relatórios e dados padronizados e para diálogos pessoais) e dados (o conteúdo dos relatórios e as medições quantitativas de atividade, produtividade e orçamento). Cada um desses componentes do gerenciamento organizacional humano tem uma contrapartida no gerenciamento de rede.

A arquitetura de um sistema de gerenciamento de rede é conceitualmente idêntica a essa analogia simples com uma organização humana. O campo do gerenciamento de rede tem sua terminologia específica própria para os vários componentes de uma arquitetura de gerenciamento de rede; portanto, adotamos aqui essa terminologia. Como mostra a Figura 9.2, há três componentes principais em uma arquitetura de gerenciamento de rede: uma entidade gerenciadora (o chefe, na analogia apresentada), os dispositivos gerenciados (as filiais) e um protocolo de gerenciamento de rede.

A **entidade gerenciadora** é uma aplicação que em geral tem um ser humano no circuito e que é executada em uma estação central de gerência de rede na NOC. Ela é o lócus da atividade de gerenciamento de rede; ela controla a coleta, o processamento, a análise e/ou a apresentação de informações de gerenciamento de rede. É nela que são iniciadas ações para controlar o comportamento da rede e é aqui que o administrador humano interage com os dispositivos da rede.

Um **dispositivo gerenciado** é um equipamento de rede (incluindo seu software) que reside em uma rede gerenciada. Ele corresponde à filial de nossa analogia humana. Um dispositivo gerenciado pode ser um hospedeiro, um roteador, uma ponte, um hub, uma impressora ou um modem. No interior de um disposi-

tivo gerenciado pode haver diversos **objetos gerenciados**. Estes são, na verdade, as peças de hardware propriamente ditas que estão dentro do dispositivo gerenciado (por exemplo, uma placa de interface de rede) e os conjuntos de parâmetros de configuração para as peças de hardware e software (por exemplo, um protocolo de roteamento intradomínio, como o RIP). Em nossa analogia humana, os objetos gerenciados podem ser os departamentos existentes na filial. Esses objetos gerenciados têm informações associadas a eles que são coletadas dentro de uma **Base de Informações de Gerenciamento** (Management Information Base — MIB); veremos que os valores dessas informações estão disponíveis para a entidade gerenciadora (e, em muitos casos, podem ser ajustados por ela). Em nossa analogia humana, a MIB corresponde aos dados quantitativos (medições de atividade, produtividade e orçamento, podendo este último ser estabelecido pela entidade gerenciadora!) que são trocados entre o escritório central e a filial. Estudaremos as MIBs detalhadamente na Seção 9.3. Por fim, reside também em cada dispositivo gerenciado um **agente de gerenciamento de rede**, um processo que é executado no dispositivo gerenciado, que se comunica com a entidade gerenciadora e que executa ações locais nos dispositivos gerenciados sob o comando e o controle da entidade gerenciadora. O agente de gerenciamento de rede é o gerente da filial em nossa analogia humana.

O terceiro componente de uma arquitetura de gerenciamento de rede é o **protocolo de gerenciamento de rede**. Esse protocolo é executado entre a entidade gerenciadora e o agente de gerenciamento de rede dos dispositivos gerenciados, o que permite que a entidade gerenciadora investigue o estado dos dispositivos gerenciados e, indiretamente, execute ações sobre eles mediante seus agentes. Agentes podem usar o protocolo de gerenciamento de rede para informar à entidade gerenciadora a ocorrência de eventos excepcionais (por exemplo, falhas de componentes ou violação de patamares de desempenho). É importante notar que o protocolo de gerenciamento de rede em si não gerencia a rede. Em vez disso, ele fornece uma ferramenta com a qual o administrador de rede pode gerenciar ("monitorar, testar, consultar, configurar, analisar, avaliar e controlar") a rede. Essa é uma distinção sutil, mas importante.

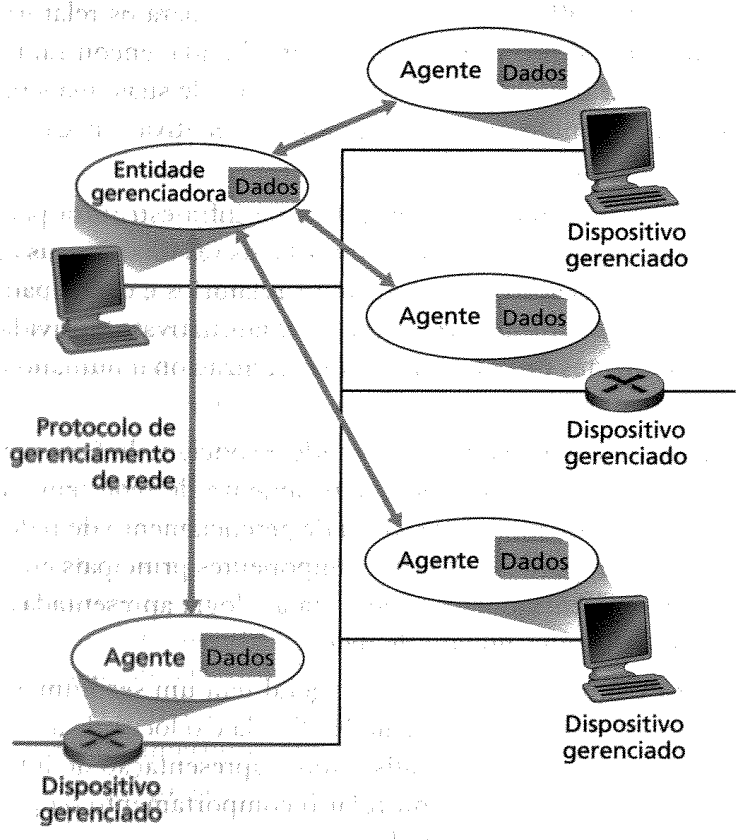


Figura 9.2 Principais componentes de uma arquitetura de gerenciamento de rede

Embora a infra-estrutura do gerenciamento de rede seja conceitualmente simples, podemos nos atrapalhar com o vocabulário especial de gerenciamento de rede, como os termos 'entidade gerenciadora', 'dispositivo gerenciado', 'agente de gerenciamento' e 'base de informações de gerenciamento'. Na terminologia do gerenciamento de rede, em nosso cenário simples de monitoração de hospedeiros, 'agentes de gerenciamento' localizados em 'dispositivos gerenciados' são periodicamente examinados pela 'entidade gerenciadora' — uma idéia simples, mas um problema lingüístico! Mas, com um pouco de sorte, lembrar sempre da analogia com uma organização humana e seus óbvios paralelos com o gerenciamento de rede nos ajudará neste capítulo.

Nossa discussão anterior sobre arquitetura de gerenciamento de rede foi genérica e se aplica, em geral, a vários padrões e esforços de gerenciamento de rede que vêm sendo propostos há anos. Os padrões de gerenciamento de rede começaram a amadurecer no final da década de 1980, sendo que o OSI **CMISE/CMIP** (Common Management Service Element/Common Management Information Protocol — **Elemento de Serviço de Gerenciamento Comum/Protocolo de Informação de Gerenciamento Comum**) [Piscatello, 1993; Stallings, 1993; Glitho, 1998] e o **SNMP** (Simple Network Management Protocol — **Protocolo Simples de Gerenciamento de Rede**) da Internet [RFC 3410; Stallings, 1999; Rose, 1996] emergiram como os dois padrões mais importantes [Miller, 1997; Subramanian, 2000]. Ambos foram projetados para ser independentes de produtos ou de redes de fabricantes específicos. Como o SNMP foi projetado e oferecido rapidamente em uma época em que a necessidade de gerenciamento de rede começava a ficar premente, ele encontrou uma ampla aceitação. Hoje, esse protocolo é a estrutura de gerenciamento de rede mais amplamente usada e disseminada. Abordaremos o SNMP em detalhes na seção seguinte.

A Central de Operações de Rede da Sprintlink

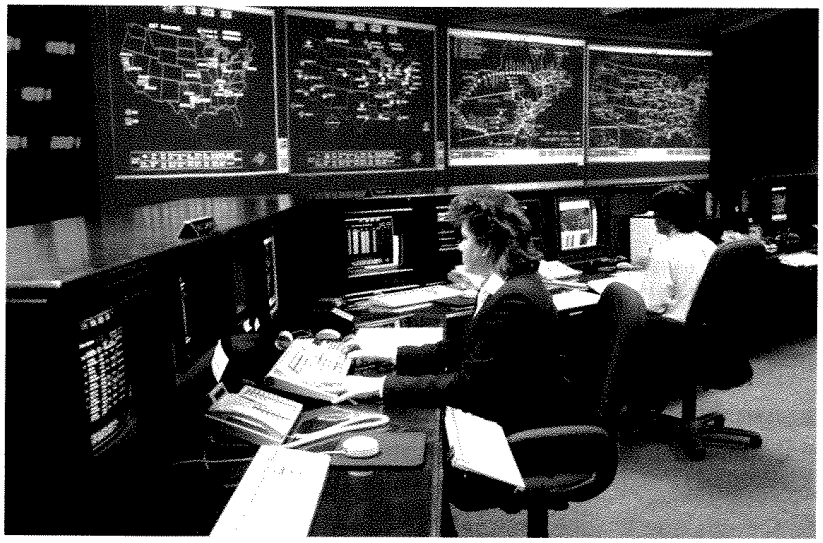
Há redes de todos os tamanhos e, desde a menor das redes residenciais até o maior dos ISPs de nível 1, cabe ao operador (ou aos operadores) da rede garantir que sua operação seja tranqüila. Mas o que acontece em uma central de operações de rede (NOC) e o que um operador de rede faz, na realidade?

A Sprint, que tem uma rede global IP (conhecida como http://www.sprintlink.net/) constituída de aproximadamente 65 POPs (Pontos de Presença, locais que contêm roteadores IP Sprintlink) de nível IP e 700 roteadores, é um dos maiores ISPs de nível 1 do planeta. A NOC da Sprintlink está localizada em Reston, no Estado de Virgínia. A Sprint também mantém NOCs para sua rede ATM, para a rede de frame-relay e para sua instalação subjacente de cabos de fibra. A qualquer instante, uma pequena equipe de quatro operadores de rede gerencia o núcleo da rede IP Sprintlink, e mais uma dezena de indivíduos trabalha na garantia de serviços IP. A automação — da monitoração, do gerenciamento da configuração, da geração e do gerenciamento de bilhetes (problemas relatados), da correlação entre alarmes, da identificação de falhas e da restauração de serviços — possibilita que esse grupo extraordinariamente reduzido de operadores gerencie essa rede tão grande e complexa.

Quando ocorre um problema, a preocupação principal do operador da Sprintlink é manter, ou restaurar rapidamente, o serviço oferecido aos clientes. Os operadores da NOC fazem triagem, diagnóstico e restauração seguindo um conjunto de procedimentos bem definidos e bem documentados correspondentes a um conjunto conhecido de problemas e/ou sintomas. Problemas que não podem ser diagnosticados imediatamente, ou que não podem ser resolvidos por operadores dentro de um período de tempo específico que depende do nível de seriedade do problema (por exemplo, 15 minutos), são passados para membros da National Technical Assistance Center (NTAC) da Sprint. Os técnicos da NTAC dedicam-se a estudar a fundo as causas dos problemas; eles também redigem procedimentos para os operadores da NOC e, quando necessário, trabalham com fornecedores de equipamentos (por exemplo, fabricantes de roteadores) para diagnosticar e resolver problemas relacionados a equipamentos específicos. Aproximadamente 70% dos problemas são tratados diretamente por operadores da

Continua...

NOC. O pessoal da NOC e da NTAC também interage com outros indivíduos, por exemplo, o pessoal das NOCs de redes de clientes da Sprint, o pessoal da Sprint que trabalha em campo e que são os 'olhos, as mãos e os ouvidos' da empresa e o pessoal que trabalha em NOCs de redes parceiras.



Técnicos da Sprint monitoram a saúde da rede em centros de operação como o mostrado acima.

Como discutimos anteriormente neste capítulo, o 'gerenciamento de rede' na Sprintlink (bem como em outros ISPs) evoluiu do gerenciamento de falhas para o gerenciamento de desempenho para o gerenciamento de serviços, com ênfase cada vez maior nas necessidades do cliente. Embora dêem a máxima atenção às necessidades dos clientes, os operadores da Sprint também se orgulham muito de sua excelência operacional e do papel que desempenham na manutenção e na segurança da infra-estrutura da rede global de um dos maiores ISPs do mundo.

9.3 A estrutura de gerenciamento padrão da Internet

Ao contrário do que o nome SNMP (protocolo simples de gerenciamento de rede) possa sugerir, o gerenciamento de rede na Internet é muito mais do que apenas um protocolo para transportar dados de gerenciamento entre uma entidade gerenciadora e seus agentes, e o SNMP passou a ser muito mais complexo do que a palavra 'simples' (o 'S') possa sugerir. As raízes da atual estrutura de gerenciamento padrão da Internet (Internet Standard Management Framework) remontam ao SGMP (Simple Gateway Monitoring Protocol — protocolo de monitoramento do gateway simples) [RFC 1028]. O SGMP foi projetado por um grupo de pesquisadores, usuários e administradores universitários de rede, cuja experiência com esse protocolo permitiu que eles projetassem, implementassem e oferecessem o SNMP em poucos meses [Lynch, 1993] — um feito muito distante dos processos de padronização atuais, que são bastante prolongados. Desde então, o SNMP evoluiu do SNMPv1 para o SNMPv2 e chegou à sua versão mais recente, o SNMPv3 [RFC 3410], lançada em abril de 1999 e atualizada em dezembro de 2002.

Na descrição de qualquer estrutura para gerenciamento de rede, certas questões devem inevitavelmente ser abordadas:

- O que está sendo monitorado (de um ponto de vista semântico)? E que tipo de controle pode ser exercido pelo administrador de rede?
- Qual é o modelo específico das informações que serão relatadas e/ou trocadas?
- Qual é o protocolo de comunicação para trocar essas informações?

Lembre-se de nossa analogia humana apresentada na seção anterior. O chefe e os gerentes das filiais precisarão acertar entre eles as medições de atividade, produtividade e orçamento que serão usadas para relatar a situação das filiais. De maneira semelhante, eles terão de concordar sobre que tipo de ações o chefe poderá realizar (por exemplo, cortar o orçamento, ordenar que o gerente da filial modifique alguma característica da operação do escritório ou demitir o pessoal e fechar a filial). Em um nível de maior detalhamento, eles precisarão concordar sobre o modo como esses dados serão relatados. Por exemplo, em que moeda (dólares, reais?) será apresentado o relatório de orçamento? Em que unidades será medida a produtividade? Embora talvez pareçam triviais, esses detalhes terão de ser acertados. Por fim, a maneira pela qual a informação trafegará entre o escritório central e as filiais (isto é, o protocolo de comunicação) deve ser especificada.

A estrutura de gerenciamento padrão da Internet aborda essas questões. A estrutura é constituída de quatro partes:

- Definições dos *objetos de gerenciamento de rede*, conhecidos como objetos MIB. Na Estrutura de Gerenciamento de Rede da Internet, as informações de gerenciamento são representadas como uma coletânea de objetos gerenciados que, juntos, formam um banco virtual de informações virtuais conhecido como MIB. Um objeto MIB pode ser um contador, tal como o número de datagramas IP descartados em um roteador devido a erros em cabeçalhos de datagramas IP ou o número de erros de detecção de portadora em uma placa de interface Ethernet; um conjunto de informações descritivas, como a versão do software que está sendo executado em um servidor DNS; informações de estado, como se um determinado dispositivo está funcionando corretamente; ou informações específicas sobre protocolos, como um caminho de roteamento até um destino. Assim, os objetos MIB definem as informações de gerenciamento mantidas por um dispositivo gerenciado. Objetos MIB relacionados são reunidos em **módulos MIB**. Em nossa analogia com uma organização humana, a MIB define a informação transportada entre a filial e a sede.
- Uma *linguagem de definição de dados*, conhecida como SMI (Structure of Management Information — Estrutura de Informação de Gerenciamento), que define os tipos de dados, um modelo de objeto e regras para escrever e revisar informações de gerenciamento. Objetos MIB são especificados nessa linguagem de definição de dados. Em nossa analogia humana, a SMI é usada para definir os detalhes do formato das informações que serão trocadas.
- *Um protocolo, SNMP*, para transmitir informações e comandos entre uma entidade gerenciadora e um agente que os executa em nome da entidade dentro de um dispositivo de rede gerenciado.
- *Capacidades de segurança e de administração*. A adição dessas capacidades representa o aprimoramento mais importante do SNMPv3 em comparação com o SNMPv2.

Assim, a arquitetura de gerenciamento da Internet é modular por projeto, com uma linguagem de definição de dados e de MIB independente de protocolo e um protocolo independente de MIB. O interessante é que essa arquitetura modular foi primeiramente criada para facilitar a transição de um gerenciamento de rede baseado em SNMP para uma estrutura de gerenciamento de rede que estava sendo desenvolvida pela ISO, que era a arquitetura de gerenciamento que concorria com o SNMP quando este foi projetado — uma transição que nunca aconteceu. Com o tempo, contudo, a modularidade do projeto do SNMP permitiu que ele evoluísse mediante três importantes revisões, com cada uma das quatro partes do SNMP discutidas anteriormente se desenvolvendo independentemente. Ficou bem claro que a decisão pela modularidade foi correta, ainda que tenha sido tomada pela razão errada!

Nas subseções seguintes, examinaremos com mais detalhes os quatro componentes mais importantes da estrutura de gerenciamento padrão da Internet.

9.3.1 SMI (Estrutura de Informações de Gerenciamento)

A **Estrutura de Informações de Gerenciamento** (Structure of Management Information — SMI) (um componente da estrutura de gerenciamento de rede cujo nome é bastante estranho e não dá nenhuma pista quanto à sua funcionalidade) é a linguagem usada para definir as informações de gerenciamento que residem em uma entidade gerenciada de rede. Essa linguagem de definição é necessária para assegurar que a sintaxe e a semântica dos dados de gerenciamento de rede sejam bem definidas e não apresentem ambigüidade. Note que a SMI não define uma instância específica para os dados em uma entidade gerenciada de rede, mas a linguagem na qual a informação está especificada. Os documentos que descrevem a SMI para o SNMPv3 (confusamente denominada SMIv2) são [RFC 2578; RFC 2579; RFC 2580]. Vamos examinar a SMI de baixo para cima, começando com seus tipos de dados básicos. Em seguida, examinaremos como os objetos gerenciados são descritos em SMI e, então, como os objetos gerenciados relacionados entre si são agrupados em módulos.

Tipos de dados básicos da SMI

O RFC 2578 especifica os tipos de dados básicos da linguagem SMI de definição de módulos MIB. Embora a SMI seja baseada na linguagem de definição de objetos ASN.1 (Abstract Syntax Notation One — notação de sintaxe abstrata 1) [ISO, 1987; ISO X.680, 1998], tantos foram os tipos de dados específicos da SMI acrescentados que esta deve ser considerada uma linguagem de definição de dados por direito próprio. Os 11 tipos de dados básicos definidos no RFC 2578 são mostrados na Tabela 9.1. Além desses objetos escalares, também é possível impor uma estrutura tabular sobre um conjunto ordenado de objetos MIB usando a construção SEQUENCE OF; consulte o RFC 2578 para mais detalhes. Grande parte dos tipos de dados da Tabela 9.1 será familiar (ou auto-explicativa) para a maioria dos leitores. O único tipo de dado que discutiremos com mais detalhes será o OBJECT IDENTIFIER, que é usado para dar nome a um objeto.

Construções SMI de nível mais alto

Além dos tipos de dados básicos, a linguagem de definição de dados SMI também fornece construções de linguagem de nível mais alto.

Tipo de dado	Descrição
INTEGER	Número inteiro de 32 bits, como definido em ASN.1, com valor entre – 231 e 231 – 1, inclusive, ou um valor de uma lista de valores constantes possíveis, nomeados.
Integer32	Número inteiro de 32 bits, com valor entre – 231 e 231 – 1, inclusive.
Unsigned32	Número inteiro de 32 bits sem sinal na faixa de 0 a 232 – 1, inclusive.
OCTET STRING	Cadeia de bytes de formato ASN.1 que representa dados binários arbitrários ou de texto de até 65.535 bytes de comprimento.
OBJECT IDENTIFIER	Formato ASN.1 atribuído administrativamente (nome estruturado); veja a Seção 9.3.2.
Endereço IP	Endereço Internet de 32 bits, na ordem de bytes de rede.
Counter32	Contador de 32 bits que cresce de 0 a 232 – 1 e volta a 0.
Counter64	Contador de 64 bits.
Gauge32	Número inteiro de 32 bits que não faz contagens além de 232 – 1 nem diminui para menos do que 0.
TimeTicks	Tempo, medido em centésimos de segundo, transcorrido a partir de algum evento.
Opaque	Cadeia ASN.1 não interpretada, necessária para compatibilidade com versões anteriores.

Tabela 9.1 Tipos de dados básicos da SMI

A construção OBJECT-TYPE é utilizada para especificar o tipo de dado, o status e a semântica de um objeto gerenciado. Esses objetos gerenciados contêm, coletivamente, os dados de gerenciamento que estão no cerne do gerenciamento de rede. Há mais de dez mil objetos definidos em diversos RFCs da Internet [RFC 2570]. A construção OBJECT-TYPE tem quatro cláusulas. A cláusula SYNTAX de uma definição OBJECT-TYPE especifica o tipo de dado básico associado ao objeto. A cláusula MAX-ACCESS especifica se o objeto gerenciado pode ser lido, escrito, criado ou ter seu valor incluído em uma notificação. A cláusula STATUS indica se a definição do objeto é atual e válida, obsoleta (caso em que não deve ser implementada, pois sua definição está incluída por motivos históricos apenas) ou depreciada (obsoleta, mas implementável por causa de sua interoperabilidade com implementações mais antigas). A cláusula DESCRIPTION contém uma definição textual e legível do objeto; ela 'documenta' a finalidade do objeto gerenciado e deve fornecer todas as informações semânticas necessárias para implementá-lo.

Como exemplo de uma construção OBJECT-TYPE, considere a definição do tipo de objeto `ipInDelivers` do RFC 2011. Esse objeto define um contador de 32 bits que monitora o número de datagramas IP que foram recebidos no dispositivo gerenciado e entregues com sucesso a um protocolo de camada superior. A última linha dessa definição diz respeito ao nome desse objeto, um tópico que consideraremos na seção seguinte.

```
ipInDelivers OBJECT-TYPE
SYNTAX          Counter32
MAX-ACCESS      read-only
STATUS          current
DESCRIPTION
        "The total number of input datagrams
        successfully delivered to IP user-protocols
        (including ICMP)."
::= { ip 9 }
```

A construção MODULE-IDENTITY permite que objetos relacionados entre si sejam agrupados, como conjunto, dentro de um 'módulo'. Por exemplo, o RFC 2011 especifica o módulo MIB que define objetos gerenciados (incluindo `ipInDelivers`) para gerenciar implementações do IP e de seu protocolo associado, o ICMP. O RFC 2012 especifica o módulo MIB para TCP, e o RFC 2013 especifica o módulo MIB para UDP. O RFC 2021 define o módulo MIB para monitoração remota, RMON. Além de conter as definições OBJECT-TYPE dos objetos gerenciados dentro do módulo, a construção MODULE-IDENTITY contém cláusulas para documentar informações de contato do autor do módulo, a data da última atualização, um histórico de revisões e uma descrição textual do módulo. Como exemplo, considere o módulo de definição para gerenciamento do protocolo IP:

```
ipMIB MODULE-IDENTITY
    LAST-UPDATED "9411010000Z"
    ORGANIZATION "IETF SNMPv2 Working Group"
    CONTACT-INFO
              "      Keith McCloghrie
             Postal: Cisco Systems, Inc.
                     170 West Tasman Drive
                     San Jose, CA 95134-1706
                     US
```

```
    Phone:          +1 408 526 5260
    E-mail:         kzm@cisco.com"
DESCRIPTION
        "The MIB module for managing IP and ICMP
        implementations, but excluding their
        management of IP routes."
REVISION "9103310000Z"
DESCRIPTION
        "The initial revision of this MIB module was part of MIB-II."
::= { mib-2 48}
```

A construção NOTIFICATION-TYPE é usada para especificar informações referentes a mensagens 'SNMPv2-Trap' e 'Information Request' (solicitação de informações) geradas por um agente ou por uma entidade gerenciadora; veja a Seção 9.3.3. Entre essas informações estão um texto de descrição (DESCRIPTION) que especifica quando tais mensagens devem ser enviadas, bem como uma lista de valores que devem ser incluídos na mensagem gerada; veja o RFC 2578 para mais detalhes. A construção MODULE-COMPLIANCE define o conjunto de objetos gerenciados dentro de um módulo que um agente deve implementar. A construção AGENT-CAPABILITIES especifica as capacidades dos agentes relativas às definições de notificação de objetos e de eventos.

9.3.2 Base de informações de gerenciamento: MIB

Com dissemos anteriormente, a **Base de Informações de Gerenciamento** (Management Information Base — MIB) pode ser imaginada como um banco virtual de informações que guarda objetos gerenciados cujos valores, coletivamente, refletem o 'estado' atual da rede. Esses valores podem ser consultados e/ou definidos por uma entidade gerenciadora por meio do envio de mensagens SNMP ao agente que está rodando em um dispositivo gerenciado em nome da entidade gerenciadora. Objetos gerenciados são especificados utilizando a construção OBJECT-TYPE da SMI discutida anteriormente e agrupados em **módulos MIB** utilizando a construção MODULE-IDENTITY.

A IETF tem estado muito atarefada com a padronização de módulos MIB associados a roteadores, hospedeiros e outros equipamentos de rede, o que inclui dados básicos de identificação sobre um determinado componente de hardware e informações de gerenciamento sobre as interfaces e os protocolos de dispositivos da rede. Até 2004 havia mais de cem módulos MIB baseados em padrões e um número ainda maior de módulos MIB especificados por fabricantes privados. Com todos esses padrões, a IETF precisava encontrar um modo de identificar e dar nome aos módulos padronizados, bem como aos objetos gerenciados específicos dentro de um módulo. Em vez de começar do nada, a IETF adotou uma estrutura padronizada de identificação de objetos (nomeação) que já tinha sido publicada pela ISO. Como acontece com muitas entidades dedicadas à padronização, a ISO tinha 'grandes planos' para sua estrutura padronizada de identificação de objetos — identificar todo e qualquer objeto padronizado possível (por exemplo, formato de dados, protocolo ou informação) em qualquer rede, independentemente das organizações dedicadas à padronização das redes (por exemplo, IETF, ISO, IEEE ou ANSI), do fabricante do equipamento ou do proprietário da rede. Um objetivo bem grandioso mesmo! A estrutura de identificação de objeto adotada pela ISO é parte da linguagem de definição de objetos ASN.1 [ISO, 1987; ISO X.680, 1998], que discutiremos na Seção 9.4. Os módulos MIB padronizados já têm seu próprio cantinho confortável dentro dessa estrutura de nomeação vastamente abrangente, como veremos a seguir.

Como mostra a Figura 9.3, pela estrutura de nomeação da ISO, os objetos são nomeados hierarquicamente. Note que cada ponto de ramo da árvore tem um nome e um número (entre parênteses); assim, qual-

quer ponto da árvore pode ser identificado pela seqüência de nomes ou números que especificam o trajeto da raiz até aquele ponto da árvore de identificação. Um programa baseado na Web — divertido, mas incompleto e não oficial — que percorre parte da árvore de identificação de objetos (usando informações sobre os ramos oferecidas por voluntários) pode ser encontrado em [Alvestrand, 1997].

No topo da hierarquia estão a ISO e o ITU-T, as duas principais entidades de padronização que tratam da ASN.1, bem como um ramo para o esforço conjunto realizado por essas duas organizações. No ramo ISO da árvore, encontramos registros para todos os padrões ISO (1.0) e para os padrões emitidos por entidades padronizadoras de vários países membros da ISO (1.2). Embora não apareça na Figura 9.3, logo abaixo desse ramo da árvore (ISO/países membros, também conhecido como 1.2), encontraríamos os Estados Unidos (1.2.840), embaixo dos quais encontraríamos um número para os padrões IEEE e ANSI e para os padrões específicos de empresas privadas. Entre as empresas privadas, estão a RSA (1.2.840.11359) e a Microsoft (1.2.840.113556), sob a qual encontraríamos os Microsoft File Formats (1.2.840.113556.4) para vários produtos da Microsoft, como o Word (1.2.840.113556.4.2). Mas estamos interessados em redes (e *não* nos arquivos do Microsoft Word), portanto, vamos voltar nossa atenção ao ramo denominado 1.3, os padrões emitidos por entidades reconhecidas pela ISO. Entre estas estão o Departamento de Defesa dos Estados Unidos (6) (sob o qual encontraremos os padrões da Internet), a Open Software Foundation (22), a associação de empresas aéreas SITA (69) e as entidades identificadas pela OTAN (57), bem como muitas outras organizações.

Sob o ramo `Internet` da árvore (1.3.6.1), há sete categorias. Sob o ramo `private` (1.3.6.1.4), encontramos uma lista [IANA, 2004b] dos nomes e códigos de empresas privadas para as mais de quatro mil empresas privadas que se registraram na Internet Assigned Numbers Authority (IANA) [IANA, 1999]. Sob o ramo `management` (1.3.6.1.2) e `MIB-2` (1.3.6.1.2.1) da árvore, encontramos a definição dos módulos MIB padronizados. Puxa! É uma longa jornada até chegarmos ao nosso cantinho no espaço de nomes da ISO!

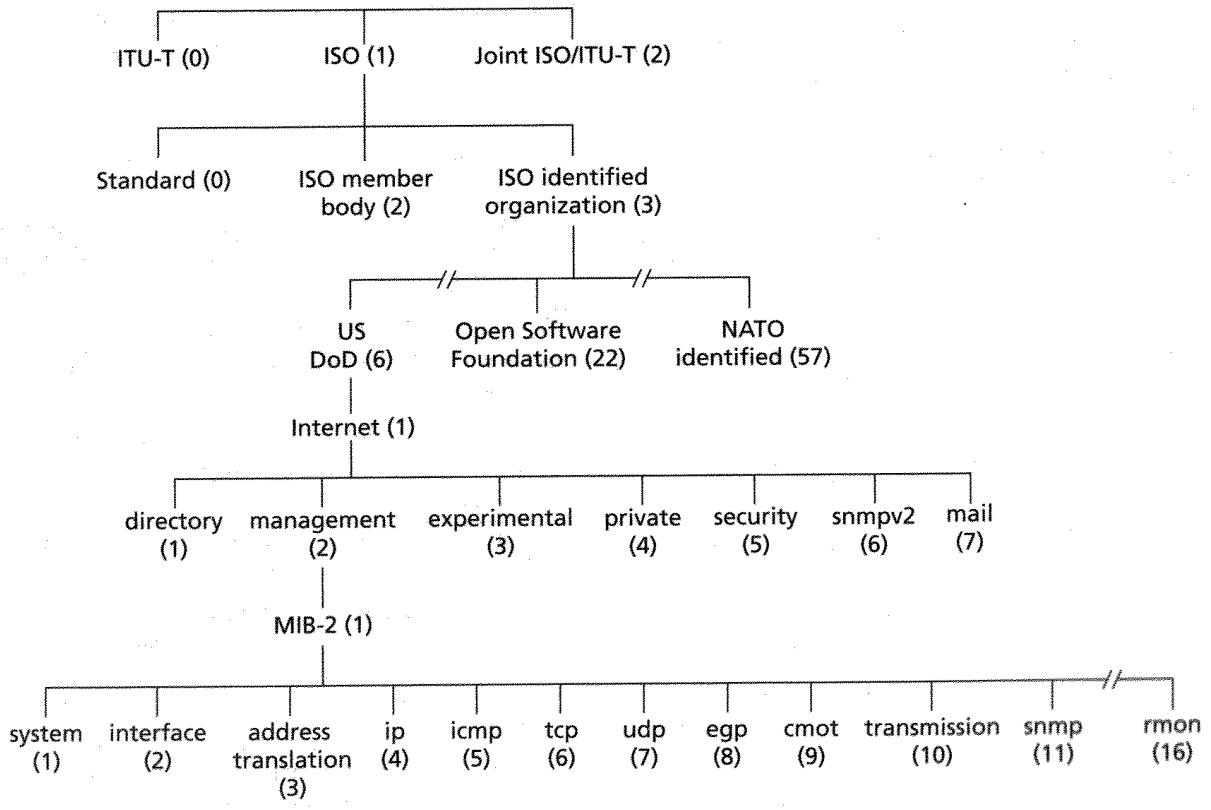


Figura 9.3 Árvore de identificadores de objetos ASN.1

Módulos MIB padronizados

O ramo mais baixo da árvore da Figura 9.3 mostra alguns dos módulos MIB orientados para hardware importantes (`system` e `interface`), bem como módulos associados a alguns dos protocolos mais importantes da Internet. O RFC 3600 relaciona todos os módulos MIB padronizados. Embora os RFCs referentes às MIBs sejam uma leitura tediosa e difícil, é muito instrutivo (isto é, assim como comer verduras, "é bom para você") considerar algumas definições de módulos MIB para ter uma idéia do tipo de informação que existe dentro de um módulo.

Os objetos gerenciados que ficam sob o título `system` contêm informações gerais sobre o dispositivo que está sendo gerenciado; todos os dispositivos gerenciados devem suportar os objetos MIB do grupo system. A Tabela 9.2 define os objetos gerenciados no grupo system, de acordo com o RFC 1213. A Tabela 9.3 define os objetos gerenciados no módulo MIB para o protocolo UDP em uma entidade gerenciada.

9.3.3 Operações do protocolo SNMP e mapeamentos de transporte

O SNMPv2 [RFC 3416] é usado para transportar informações da MIB entre entidades gerenciadoras e agentes, executando em nome das entidades gerenciadoras. A utilização mais comum do SNMP é em um **modo comando-resposta**, no qual a entidade gerenciadora SNMPv2 envia uma requisição a um agente SNMPv2, que a recebe, realiza alguma ação e envia uma resposta à requisição. Em geral, uma requisição é usada para consultar (recuperar) ou modificar (definir) valores de objetos MIB associados a um dispositivo gerenciado. Um segundo uso comum do SNMP é para um agente enviar uma mensagem não solicitada, conhecida como **mensagem trap**, à entidade gerenciadora. As mensagens trap são usadas para notificar uma entidade gerenciadora de uma situação excepcional que resultou em mudança nos valores dos objetos MIB. Vimos anteriormente, na Seção 9.1, que o administrador de rede pode querer receber uma mensagem trap, por exemplo, quando uma interface cai, quando o congestionamento atinge um nível predefinido em um enlace ou quando ocorre qualquer outro evento notável. Observe que há uma série de compromissos importantes entre consulta de objetos (interação comando-resposta) e envio assíncrono de mensagens não solucionadas; veja os exercícios ao final deste capítulo.

Identificador de objeto	Nome	Tipo	Descrição (segundo o RFC 1213)
1.3.6.1.2.1.1.1	`sysDescr`	OCTET STRING	"Nome completo e identificação da versão do tipo de hardware do sistema, do sistema operacional do software e do software de rede."
1.3.6.1.2.1.1.2	`sysObjectID`	OBJECT IDENTIFIER	ID atribuído pelo fabricante do objeto que "fornece um meio fácil e não ambíguo para determinar 'que tipo de caixa' está sendo gerenciada."
1.3.6.1.2.1.1.3	`sysUpTime`	TimeTicks	"O tempo (em centésimos de segundo) desde que a porção de gerenciamento de rede do sistema foi reinicializada pela última vez."
1.3.6.1.2.1.1.4	`sysContact`	OCTET STRING	"A pessoa de contato para esse nó gerenciado, juntamente com a informação sobre como contatá-la."
1.3.6.1.2.1.1.5	`sysName`	OCTET STRING	"Um nome atribuído administrativamente para esse nó gerenciado. Por convenção, esse é o nome de domínio totalmente qualificado do nó."
1.3.6.1.2.1.1.6	`sysLocation`	OCTET STRING	"A localização física do nó."
1.3.6.1.2.1.1.7	`sysServices`	Integer32	Um valor codificado que indica o conjunto de serviços disponível no nó: aplicações físicas (por exemplo, um repetidor), de enlace/sub-rede (por exemplo, ponte), de Internet (por exemplo, gateway IP), fim-a-fim (por exemplo, hospedeiro).

Tabela 9.2 Objetos gerenciados no grupo system da MIB-2

Identificador de objeto	Nome	Tipo	Descrição (segundo o RFC 2013)
1.3.6.1.2.1.7.1	udpInDatagrams	Counter32	"número total de datagramas UDP entregues a usuários UDP"
1.3.6.1.2.1.7.2	udpNoPorts	Counter32	"número total de datagramas UDP recebidos para os quais não havia nenhuma aplicação na porta de destino"
1.3.6.1.2.1.7.3	udpInErrors	Counter32	"número de datagramas UDP recebidos que não puderam ser entregues por outras razões que não a falta de uma aplicação na porta de destino"
1.3.6.1.2.1.7.4	udpOutDatagrams	Counter32	"número total de datagramas UDP enviados dessa entidade"
1.3.6.1.2.1.7.5	udpTable	SEQUENCE de UdpEntry	"uma seqüência de objetos UdpEntry, um para cada porta que esteja correntemente aberta por uma aplicação, dando o endereço IP e o número da porta usada por essa aplicação"

Tabela 9.3 Objetos gerenciados no módulo MIB-2 udp

O SNMPv2 define sete tipos de mensagens, conhecidas genericamente como PDUs (*protocol data units*), como mostra a Tabela 9.4. O formato da PDU é mostrado na Figura 9.4.

As PDUs `GetRequest`, `GetNextRequest` e `GetBulkRequest` são enviadas de uma entidade gerenciadora a um agente para requisitar o valor de um ou mais objetos MIB no dispositivo gerenciado do agente. Os identificadores de objeto dos objetos MIB cujos valores estão sendo requisitados são especificados na porção de vinculação de variáveis da PDU. `GetRequest`, `GetNextRequest` e `GetBulkRequest` diferem no grau de especificidade de seus pedidos de dados. `GetRequest` pode requisitar um conjunto arbitrário de valores MIB; múltiplas `GetNextRequest` podem ser usadas para percorrer a seqüência de uma lista ou tabela de objetos MIB, e `GetBulkRequest` permite que um grande bloco de dados seja devolvido, evitando a sobrecarga incorrida quando tiverem de ser enviadas múltiplas mensagens `GetRequest` ou `GetNextRequest`. Em todos os três casos, o agente responde com uma PDU `Response` que contém os identificadores de objetos e seus valores associados.

Tipo de SNMPv2-PDU	Remetente–receptor	Descrição
`GetRequest`	gerente a agente	pega o valor de uma ou mais instâncias de objetos MIB
`GetNextRequest`	gerente a agente	pega o valor da próxima instância de objeto MIB na lista ou tabela
`GetBulkRequest`	gerente a agente	pega valores em grandes blocos de dados, por exemplo, valores em uma grande tabela
`InformRequest`	gerente a gerente	informa à entidade gerenciadora remota valores da MIB que são remotos para seu acesso
`SetRequest`	gerente a agente	define valores de uma ou mais instâncias de objetos MIB
`Response`	agente a gerente ou gerente a gerente	gerada em resposta a
		`GetRequest`,
		`GetNextRequest`,
		`GetBulkRequest`,
		`SetRequest` PDU ou `InformRequest`
`SNMPv2-Trap`	agente a gerente	informa ao gerente um evento excepcional

Tabela 9.4 Tipos de SNMPv2-PDUs

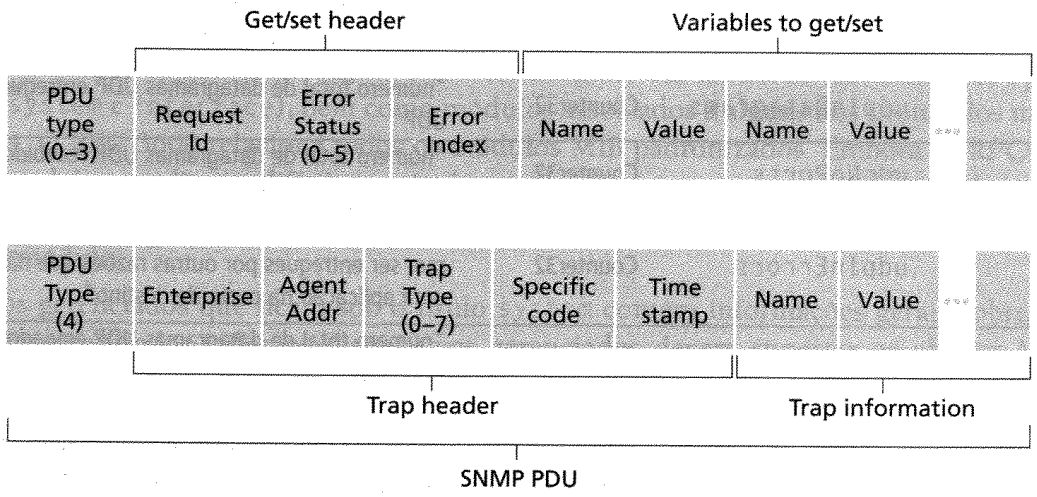


Figura 9.4 Formato da SNMP-PDU

- A PDU `SetRequest` é usada por uma entidade gerenciadora para estabelecer o valor de um ou mais objetos MIB em um dispositivo gerenciado. Um agente responde com uma PDU `Response` que contém uma mensagem Error Status 'noError' para confirmar que o valor na verdade foi estabelecido.
- A PDU `InformRequest` é usada por uma entidade gerenciadora para comunicar a outra entidade gerenciadora informações MIB remotas à entidade receptora. A entidade receptora responde com uma PDU `Response` com a mensagem Error Status 'noError' para reconhecer o recebimento da PDU `InformRequest`.
- O tipo final de SNMPv2-PDU é a mensagem trap. Mensagens trap são geradas assincronamente, isto é, não são geradas em resposta a uma requisição recebida, mas em resposta a um evento para o qual a entidade gerenciadora requer notificação. O RFC 1907 define tipos conhecidos de trap que incluem uma partida a frio ou a quente realizada por um dispositivo, a ativação ou interrupção de um enlace, a perda de um vizinho ou um evento de falha de autenticação. Uma requisição de trap recebida não exige resposta de uma entidade gerenciadora.

Dada a natureza comando-resposta do SNMPv2, convém observar que, embora as SNMP-PDUs possam ser transportadas por muitos protocolos de transporte diferentes, elas são tipicamente transportadas na carga útil de um datagrama UDP. Na verdade, o RFC 1906 estabelece que o UDP é o 'mapeamento de transporte preferencial'. Uma vez que o UDP é um protocolo de transporte não confiável, não há garantia de que um comando ou sua resposta será recebido no destino pretendido. O campo Request Id da PDU é usado pela entidade gerenciadora para numerar as requisições que faz a um agente; a resposta de um agente adota a Request ID do comando recebida. Assim, o campo Request Id pode ser usado pela entidade gerenciadora para detectar comandos ou respostas perdidos. A entidade gerenciadora é quem decidirá se retransmitirá um comando se nenhuma resposta correspondente for recebida após um dado período de tempo. Em particular, o padrão SNMP não impõe nenhum procedimento específico de retransmissão, nem mesmo diz que o comando deve ser enviado, em primeiro lugar. Ele requer apenas que a entidade gerenciadora "aja com responsabilidade em relação à freqüência e à duração das retransmissões". Isso, é claro, nos leva a pensar como deve agir um protocolo 'responsável'!

9.3.4 Segurança e administração

Os projetistas do SNMPv3 têm dito que o "SNMPv3 pode ser imaginado como um SNMPv2 com capacidades adicionais de segurança e de administração" [RFC 3410]. Certamente, há mudanças no SNMPv3 em relação ao SNMPv2, mas em nenhum lugar essas mudanças são mais evidentes do que nas áreas da admi-

nistração e da segurança. O papel central da segurança no SNMPv3 era particularmente importante, já que a falta de segurança adequada resultava no uso do SNMP primordialmente para monitorar, em vez de controlar (por exemplo, `SetRequest` é raramente usada no SNMPv1).

À medida que o SNMP amadurecia, passando por três versões, sua funcionalidade crescia, mas infelizmente crescia também o número de documentos de padronização relacionados a ele. Isso fica evidenciado pelo fato de que há agora um RFC [RFC 3411] que "descreve uma arquitetura para descrever os Ambientes de Gerenciamento do SNMP"! Embora a idéia de uma 'arquitetura' para 'descrever um ambiente' possa ser um pouco excessiva para nossa cabeça, o objetivo do RFC 3411 é admirável — introduzir uma linguagem comum para descrever a funcionalidade e as ações executadas por um agente ou entidade gerenciadora SNMPv3. A arquitetura de uma entidade SNMPv3 é direta, e viajar por ela servirá para solidificar nosso entendimento do SNMP.

As denominadas **aplicações SNMP** consistem em um gerador de comandos, um receptor de notificações e um transmissor proxy (todos normalmente encontrados em uma entidade gerenciadora); um elemento respondedor de comandos e um originador de notificações (ambos tipicamente encontrados em um agente), e na possibilidade de outras aplicações. O gerador de comandos gera as PDUs `GetRequest`, `GetNextRequest`, `GetBulkRequest` e `SetRequest`, que examinamos na Seção 9.3.3, e processa as respostas recebidas dessas PDUs. O elemento respondedor de comandos executa em um agente e recebe, processa e responde (usando a mensagem Response) às PDUs `GetRequest`, `GetNextRequest`, `GetBulkRequest` e `SetRequest` recebidas. A aplicação originadora de notificações de um agente gera PDUs `Trap`; essas PDUs podem ser recebidas e processadas em uma aplicação receptora de notificações em uma entidade gerenciadora. A aplicação do transmissor proxy repassa as PDUs de requisição, notificação e resposta.

Uma PDU enviada por uma aplicação SNMP passa, em seguida, por um 'processador' SNMP, antes de ser enviada via protocolo de transporte apropriado. A Figura 9.5 mostra como uma PDU gerada pela aplicação geradora de comandos entra primeiramente no módulo de despacho, onde é determinada a versão do SNMP. A PDU é então processada no sistema de processamento de mensagens, no qual é envelopada em um cabeçalho de mensagem que contém o número da versão do SNMP, uma mensagem Id e informações sobre o tamanho da mensagem. Caso seja necessária criptografia ou autenticação, são incluídos também os campos de cabeçalho apropriados para essas informações; veja o RFC 3411 para mais detalhes. Por fim, a mensagem SNMP (a PDU gerada pela aplicação e mais as informações do cabeçalho de mensagem) é passada ao protocolo de transporte apropriado. O protocolo de transporte preferencial para transportar mensagens SNMP é o UDP (isto é, as mensagens SNMP são transportadas como carga útil de um datagrama UDP), e o número de porta preferencial para o SNMP é a porta 161. A porta 162 é usada para mensagens trap.

Vimos anteriormente que as mensagens SNMP são usadas não somente para monitorar, mas também para controlar (por exemplo, por meio do comando `SetRequest`) elementos da rede. É claro que um intruso que conseguisse interceptar mensagens SNMP e/ou gerar seus próprios pacotes SNMP na infra-estrutura de gerenciamento poderia criar um grande tumulto na rede. Assim, é crucial que mensagens SNMP sejam transmitidas com segurança. Surpreendentemente, foi somente na versão mais recente do SNMP que a segurança recebeu a atenção merecida. Sua segurança é conhecida como **segurança baseada no usuário** [RFC 3414], pois utiliza o conceito tradicional de um usuário, identificado por um nome de usuário, ao qual as informações de segurança — uma senha, um valor de chave ou acessos privilegiados — são associadas. O SNMPv3 fornece criptografia, autenticação, proteção contra ataques de reprodução (veja as seções 8.2. e 8.3) e controle de acesso.

Criptografia. As SNMP-PDUs podem ser criptografadas usando o DES no modo encadeamento de blocos de cifras; consulte a Seção 8.2 para uma discussão sobre o DES. Note que, uma vez que o DES é um sistema de chaves compartilhadas, o valor da chave secreta dos dados de codificação do usuário deve ser conhecido pela entidade receptora que deve decriptar os dados.

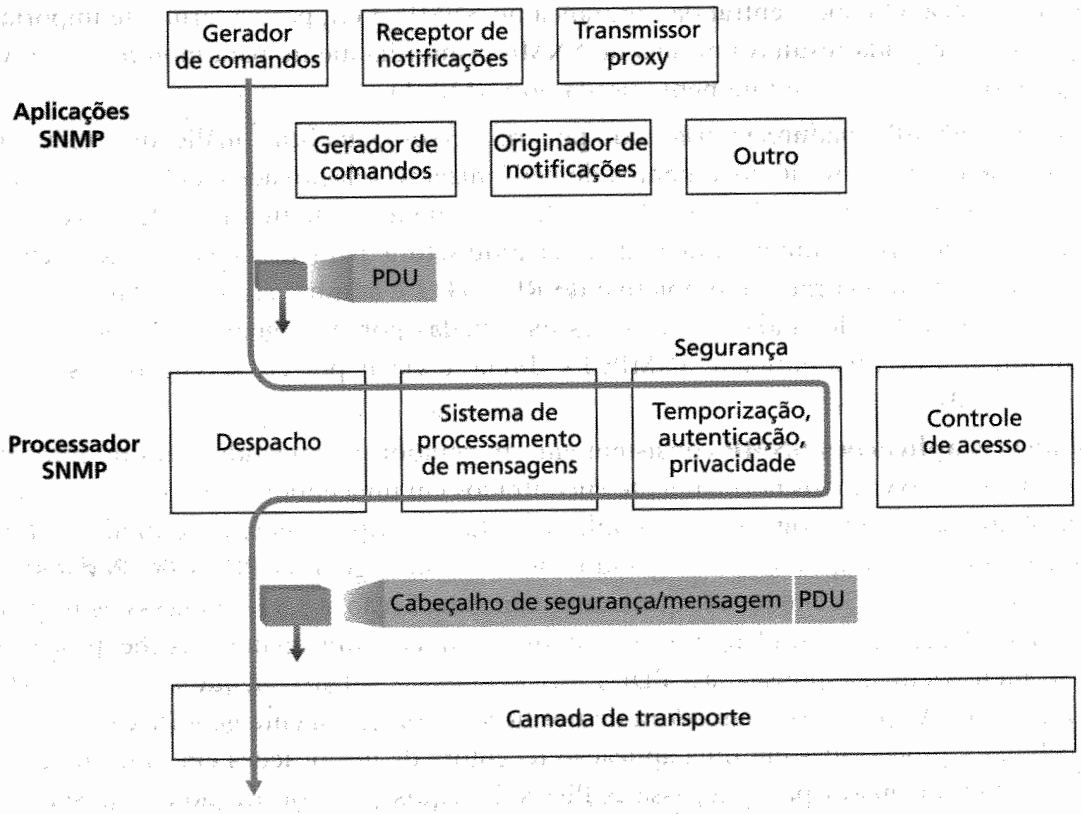


Figura 9.5 Processador e aplicações do SNMPv3

Autenticação. O SNMP combina o uso de uma função de hash, como o algoritmo MD5 que foi estudado na Seção 8.4, com um valor de chave secreta para fornecer tanto autenticação como proteção contra adulterações. A abordagem, conhecida como HMAC (Hashed Message Authentication Codes — Códigos de Autenticação de Mensagens Resumidas) [RFC 2104], é conceitualmente simples. Suponha que o remetente tenha uma SNMP-PDU, m, que ele quer enviar ao receptor. Essa PDU pode já ter sido criptografada. Suponha também que tanto o remetente quanto o receptor conheçam uma chave secreta compartilhada, K, que não precisa ser a mesma chave usada para a criptografia. O remetente enviará m ao receptor. Contudo, em vez de enviar um simples resumo de mensagem (veja Seção 8.4.2) ou código de integridade de mensagem (*message integrity code* — MIC), $MIC(m)$, que foi computado sobre m para se proteger contra adulteração, o remetente anexa a chave compartilhada secreta a m e calcula um MIC, $MIC(m,K)$ sobre a PDU e a chave combinadas. O valor $MIC(m,K)$ (mas não a chave secreta!) é, então, transmitido juntamente com m. Quando o receptor recebe m, ele anexa a chave secreta K e calcula $MIC(m,K)$. Se esse valor computado combinar com o valor transmitido de $MIC(m,K)$, então o receptor saberá não somente que a mensagem não foi adulterada, como também que foi enviada por alguém que conhece o valor de K, isto é, por um remetente de confiança e, agora, autenticado. Quando em operação, o HMAC na verdade realiza a operação de anexar e calcular o hash duas vezes, usando, cada vez, um valor de chave ligeiramente diferente; consulte o RFC 2104 para mais detalhes.

Proteção contra ataques de reprodução. Lembre-se de que podem ser usados nonces (veja o Capítulo 8) para proteção contra ataques de reprodução. O SNMPv3 adota uma abordagem relacionada. Para garantir que uma mensagem recebida não seja uma reprodução de alguma mensagem anterior, o receptor exige que o remetente inclua em cada mensagem um valor baseado em um contador no *receptor*. Esse contador, que funciona como um nonce, reflete o período de tempo decorrido entre a última reinicialização do software de gerenciamento de rede do remetente e o número total de reinicializações desde a última vez que o software de gerenciamento de rede do receptor

foi configurado. Contanto que o contador em uma mensagem recebida esteja dentro de uma determinada margem de erro em relação ao próprio valor do receptor, a mensagem é aceita como uma mensagem original e a partir daí pode ser autenticada e/ou decriptada. Consulte o RFC 3414 para mais detalhes.

- *Controle de acesso.* O SNMPv3 fornece um controle de acesso baseado em visões [RFC 2575] que controla quais das informações de gerenciamento de rede podem ser consultadas e/ou definidas por quais usuários. Uma entidade SNMP armazena informações sobre direitos de acesso e políticas em um banco de dados de configuração local (*local configuration datastore* — LCD). Partes do LCD são acessíveis elas próprias como objetos gerenciados, definidos na MIB de Configuração do Modelo de Controle de Acesso Baseado em Visões (View-Based Access Control Model Configuration) [RFC 3415] e, portanto, podem ser gerenciados e manipulados remotamente via SNMP.

9.4 ASN.1

Neste livro, examinamos uma série de assuntos interessantes sobre redes de computadores. Esta seção sobre a ASN.1, contudo, pode não fazer parte da lista dos dez tópicos mais interessantes. Assim como as verduras, conhecer a ASN.1 e a questão mais ampla de serviços de apresentação é algo que "é bom para você". A ASN.1 é um padrão originado na ISO, usado em uma série de protocolos relacionados à Internet, particularmente na área de gerenciamento de rede. Por exemplo, vimos na Seção 9.3 que as variáveis MIB do SNMP estão inextricavelmente ligadas à ASN.1. Portanto, embora o material sobre a ASN.1 apresentado nesta seção seja bastante árido, esperamos que o leitor nos dê um voto de confiança e acredite que o material é importante.

Princípios na prática

Há centenas (se não milhares) de produtos de gerenciamento de rede à disposição atualmente — todos incorporando até certo ponto os fundamentos de estrutura de gerenciamento de rede e SNMP que estudamos nesta seção. Um levantamento desses produtos está bem além do escopo deste livro e (sem dúvida) do âmbito da atenção do leitor. Assim, fornecemos aqui indicações de alguns dos produtos mais proeminentes. Um bom ponto de partida para se ter uma visão geral da abrangência das ferramentas de gerenciamento de rede é o Capítulo 12 do livro de [Subramanian, 2000].

As ferramentas de gerenciamento de rede podem ser divididas em duas classes amplas: as ferramentas dos fabricantes de equipamentos de rede, que são destinadas ao gerenciamento dos equipamentos desses fabricantes, e as ferramentas que visam ao gerenciamento de redes com equipamentos heterogêneos. Entre as ofertas específicas de fabricantes está o CiscoWorks 2000 [Cisco CiscoWorks, 2000], para o gerenciamento de LANs e WANs construídas sobre a plataforma Cisco. O Optivity Network Management System da Nortel [Nortel, 2000] oferece gerenciamento de rede, gerenciamento de serviço e gerenciamento de políticas (gerenciamento de largura de banda, QoS, segurança de aplicação e endereço IP).

Entre as ferramentas populares para o gerenciamento de redes heterogêneas, citamos o Openview, da Hewlett-Packard [Openview, 2004], o Spectrum, da Aprisma [Aprisma, 2004], e o sistema de gerenciamento de rede Solstice, da Sun [Sun, 2004]. Esses três sistemas adotam uma arquitetura de sistema distribuído, na qual vários servidores coletam informações de gerenciamento de rede do domínio que gerenciam. A estação de gerenciamento de rede então coleta resultados desses servidores, apresenta informações e executa ações de controle. Os três produtos suportam os protocolos SNMP e CMIP e fornecem assistência automática para a correlação entre eventos e alarmes.

Para motivar nossa discussão, faça o seguinte exercício mental: suponha que fosse possível copiar dados, de maneira confiável, da memória de um computador diretamente para a memória de outro computador. Se isso fosse possível, o problema da comunicação estaria 'resolvido'? A resposta a essa pergunta depende de como se define 'problema de comunicação'. Com certeza, uma cópia perfeita, memória a memória, comunicaria com exatidão os bits e os bytes de uma máquina para outra. Mas essa cópia exata de bits e bytes significa que, quando o software que estiver sendo executado no computador receptor acessar esses dados, ele verá os mesmos valores que estavam armazenados na memória do computador remetente? A resposta a essa pergunta é "não necessariamente"! O nó da questão é que diferentes arquiteturas de computadores, diferentes sistemas operacionais e compiladores têm diferentes convenções de armazenamento e representação de dados. Quando se trata de comunicar e armazenar dados entre vários computadores (como acontece em todas as redes de comunicação), fica evidente que esse problema da representação de dados tem de ser resolvido.

Como exemplo desse problema, considere o fragmento simples em linguagem C a seguir. Como essa estrutura deveria ser disposta na memória?

```
struct {
    char code;
    int x;
} test;
test.x = 259;
test.code = 'a';
```

O lado esquerdo da Figura 9.6 mostra uma disposição possível para esses dados em uma arquitetura hipotética: há um único byte de memória que contém o caractere 'a', seguido de uma palavra de 16 bits que contém o valor inteiro 259, armazenado com o byte mais significativo antes. A disposição na memória de um outro computador é mostrada no lado direito da Figura 9.6. O caractere 'a' é seguido pelo valor inteiro armazenado com o byte menos significativo antes e com o inteiro de 16 bits alinhado para começar em uma fronteira de palavra de 16 bits. Certamente, se quiséssemos executar uma cópia ao pé da letra entre as memórias desses dois computadores e usar a mesma definição de estrutura para acessar os valores armazenados, veríamos resultados diferentes nos dois computadores!

O fato de diferentes arquiteturas terem diferentes formatos para os dados internos é um problema real e universal. O problema específico do armazenamento de inteiros em diferentes formatos é tão comum que tem até um nome. A ordem de armazenamento de inteiros 'big-endian' (extremidade maior) armazena primeiramente os bytes mais significativos (nos endereços de armazenamento mais baixos). A ordem 'little-endian' (extremidade menor) armazena primeiramente os bytes menos significativos. Os processadores Sparc da Sun e os da Motorola são do tipo 'big-endian', ao passo que os processadores da Intel e da DEC/Compac Alpha são do tipo 'little-endian'. Como curiosidade, os termos 'big-endian'e 'little-endian' vêm do livro *Viagens de Gulliver*, de Jonathan Swift, no qual dois grupos de pessoas insistem, dogmaticamente, em fazer uma coisa simples de duas maneiras diferentes (esperamos que a analogia com a comunidade da arquitetura de computadores fique clara). Um grupo da Terra de Lilliput insiste em quebrar os ovos pela extremidade

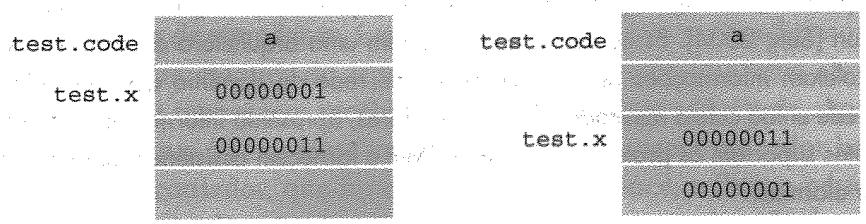


Figura 9.6 Duas organizações diferentes de dados em duas arquiteturas diferentes

maior (os 'big-endians'), enquanto o outro grupo insiste em quebrá-los pela extremidade menor. Essa diferença foi a causa de um grande conflito e de uma rebelião civil.

Visto que diferentes computadores armazenam e representam dados de modo diferente, como os protocolos de rede devem enfrentar o problema? Por exemplo, se um agente SNMP estiver prestes a enviar uma mensagem Response que contém um número inteiro que representa a contagem do número de datagramas UDP recebidos, como ele deveria representar o valor inteiro a ser enviado à entidade gerenciadora — na ordem 'big-endian' ou na ordem 'little-endian'? Uma alternativa seria o agente enviar os bytes do inteiro na mesma ordem em que serão armazenados na entidade gerenciadora. Outra alternativa seria o agente enviar o inteiro em sua própria ordem de armazenamento, deixando à entidade gerenciadora a responsabilidade de reordenar os bytes quando necessário. Qualquer uma das alternativas exigiria que o remetente ou o receptor soubesse qual o formato de representação de inteiros do outro.

Uma terceira alternativa é ter um método independente de máquina, de sistema operacional e de linguagem para descrever números inteiros e outros tipos de dados (isto é, uma linguagem de descrição de dados) e regras que estabeleçam a maneira como cada um desses tipos de dados deve ser transmitido pela rede. Quando forem recebidos dados de um determinado tipo, eles estarão em um formato conhecido e, assim, poderão ser armazenados em qualquer formato específico que um dado computador exija. Tanto a SMI, que estudamos na Seção 9.3, quanto a ASN.1 adotam essa terceira alternativa. No jargão da ISO, esses dois padrões descrevem um **serviço de apresentação** — o serviço de transmitir e traduzir informações de um formato específico de uma máquina para outro. A Figura 9.7 ilustra um problema de apresentação no mundo real; nenhum dos receptores entende a idéia essencial que está sendo comunicada — que o interlocutor gosta de algo. Como mostrado na Figura 9.8, um serviço de apresentação pode resolver esse problema traduzindo a idéia para uma linguagem inteligível (pelo serviço de apresentação), independentemente de quem fala, enviando essa informação ao receptor e, em seguida, traduzindo-a para uma linguagem que o receptor entende.

A Tabela 9.5 mostra alguns tipos de dados definidos pela ASN.1. Lembre-se de que encontramos os tipos de dados INTEGER, OCTET STRING e OBJECT IDENTIFIER em nosso estudo anterior da SMI. Como nossa meta aqui (felizmente) não é fornecer uma introdução completa à ASN.1, remetemos o leitor aos padrões ou ao livro impresso e on-line de [Larmouth, 1996], para a descrição de tipos e construtores ASN.1, como SEQUENCE e SET, que permitem definir os tipos de dados estruturados.

Além de fornecer uma linguagem de descrição de dados, a ASN.1 oferece **Regras Básicas de Codificação** (Basic Encoding Rules — BERs), que especificam como instâncias de objetos que foram definidas usando a linguagem de descrição de dados ASN.1 devem ser enviadas pela rede. A BER adota a abordagem **TLV** (Type, Length, Value — **Tipo, Comprimento, Valor**) para a codificação de dados para transmissão. Para cada item de dados a ser remetido, são enviados o tipo dos dados, o comprimento do item de dados e o valor do item de dados, nessa ordem. Com essa simples convenção, os dados recebidos se auto-identificam.

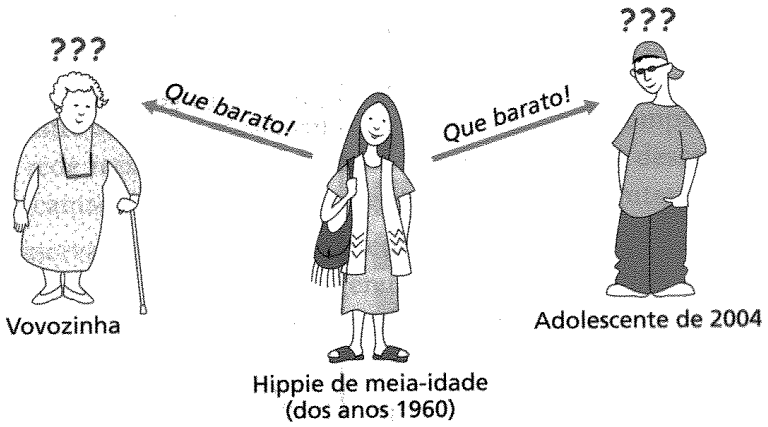


Figura 9.7 O problema da apresentação

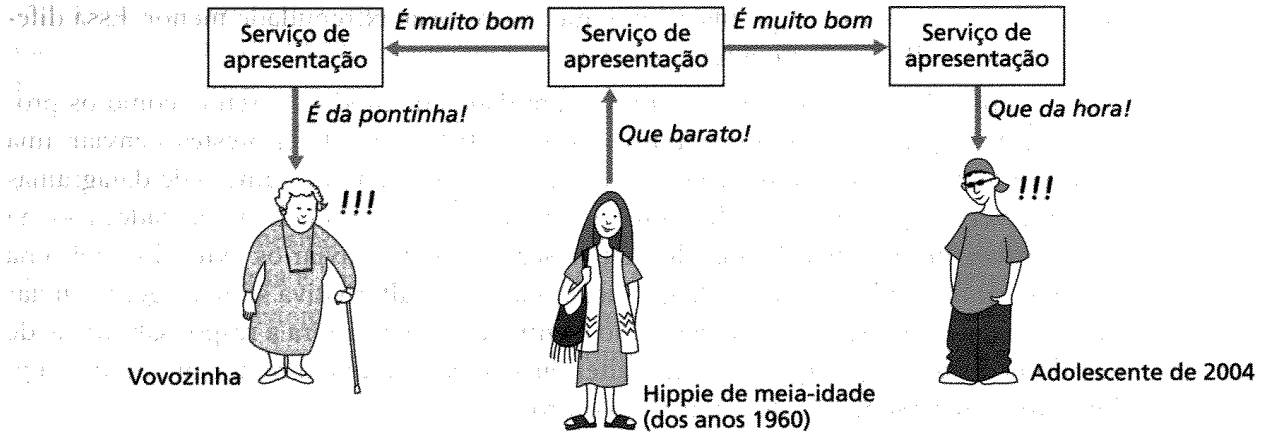


Figura 9.8 Resolução do problema da apresentação

A Figura 9.9 mostra como os dois itens de dados de um exemplo simples seriam enviados. Nesse exemplo, o remetente quer enviar a cadeia de caracteres 'smith' seguida de um valor decimal 259 (que é igual a 00000001 00000011 em linguagem binária ou a um valor de byte 1 seguido de um valor de byte 3) adotando a ordem 'big-endian'. O primeiro byte da cadeia transmitida tem o valor 4, que indica que o tipo do item de dado seguinte é um OCTET STRING; esse é o 'T' do código TLV. O segundo byte da cadeia contém o comprimento do OCTET STRING, nesse caso, 5. O terceiro byte da cadeia transmitida inicia o OCTET STRING de comprimento 5; ele contém a representação ASCII da letra 's'. Os valores T, L e V dos dados seguintes são 2 (o tag do tipo INTEGER), 2 (isto é, um inteiro de comprimento de 2 bytes) e a representação 'big-endian' de 2 bytes do valor decimal 259.

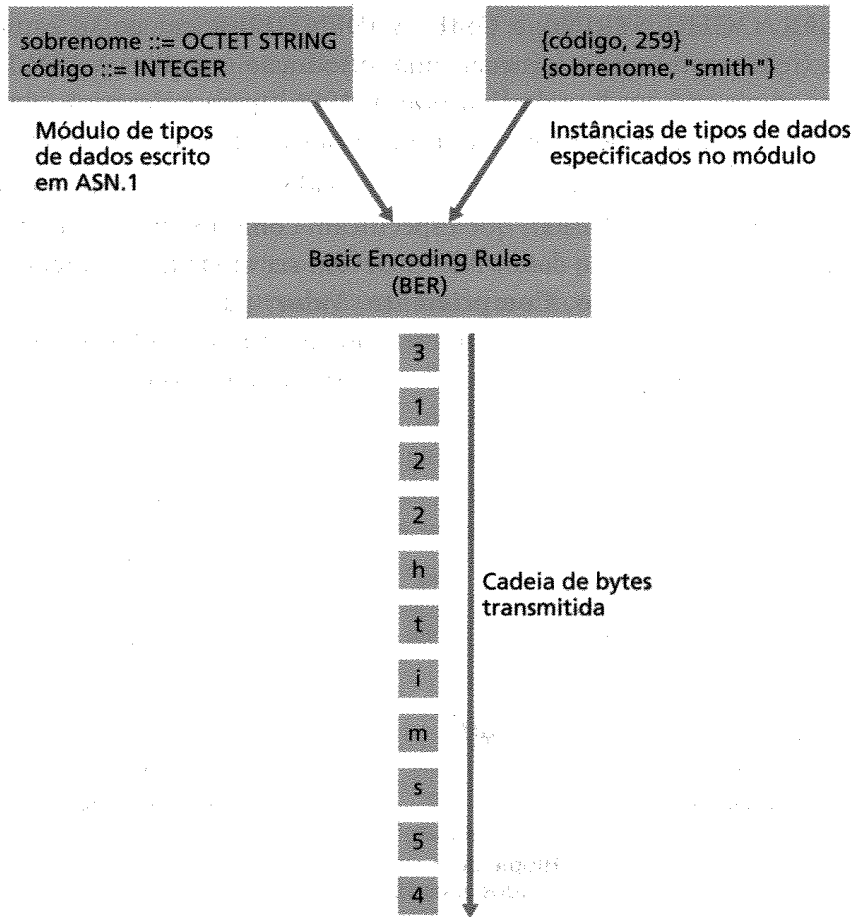


Figura 9.9 Exemplo de codificação BER

Tag	Tipo	Descrição
1	BOOLEAN	valor é 'verdadeiro' ou 'falso'
2	INTEGER	pode ser arbitrariamente grande
3	BITSTRING	lista de um ou mais bits
4	OCTET STRING	lista de um ou mais bytes
5	NULL	sem valor
6	OBJECT IDENTIFIER	nome, na árvore de nomeação padrão ASN.1; veja a Seção 9.2.2
9	REAL	ponto flutuante

Tabela 9.5 Tipos de dados ASN.1 selecionados

Em nossa discussão, abordamos apenas um subconjunto pequeno e simples da ASN.1. Entre os recursos para aprender mais sobre a ASN.1 estão os documentos padronizados da ASN.1 [ISO, 1987; ISO X.680, 1998], o livro on-line de [Larmouth, 1996], relativo à OSI e [OSS, 2004].

9.5 Conclusão

Nosso estudo sobre o gerenciamento de redes — e, na verdade, sobre toda a rede — agora está completo!

Neste capítulo final sobre gerenciamento de redes, começamos apresentando os motivos da necessidade de fornecer ferramentas adequadas ao administrador de rede — a pessoa que tem a tarefa de manter a rede 'ligada e funcionando' — para monitorar, testar, consultar, configurar, analisar, avaliar e controlar a operação da rede. Nossas analogias com a administração de sistemas complexos, como usinas elétricas, aviões e organizações humanas, nos ajudaram a fornecer motivos para essa necessidade. Vimos que a arquitetura do sistema de gerenciamento de rede gira em torno de cinco componentes fundamentais: (1) um gerenciador de rede, (2) um conjunto de dispositivos gerenciados remotamente (pelo gerenciador de rede), (3) as bases de informações de gerenciamento (MIBs) existentes nesses dispositivos, contendo dados sobre seu estado e sua operação, (4) os agentes remotos que reportam informação das MIBs e executam ações sob o controle do gerenciador de rede e (5) um protocolo para a comunicação entre o gerenciador de rede e os dispositivos remotos.

Em seguida, examinamos em detalhes a estrutura de gerenciamento de rede da Internet e, em particular, o protocolo SNMP. Vimos como o SNMP apresenta os cinco componentes fundamentais de uma arquitetura de gerenciamento de rede e gastamos um tempo considerável examinando objetos MIB, a SMI — a linguagem de definição de dados para especificação das MIBs — e o protocolo SNMP em si. Observamos que a SMI e a ASN.1 estão inextricavelmente interligadas e que a ASN.1 desempenha um papel fundamental na camada de apresentação do modelo de referência de sete camadas ISO/OSI, e então fizemos um estudo rápido da ASN1. Talvez mais importante do que os detalhes da ASN.1 em si foi a necessidade percebida de fornecer a tradução entre formatos de dados específicos de cada computador de uma rede. Enquanto algumas arquiteturas de rede reconhecem explicitamente a importância desse serviço por terem uma camada de apresentação, essa camada não existe na pilha de protocolos da Internet.

Convém também observar que há muitos tópicos no gerenciamento de rede que preferimos *não* abordar — tópicos como as falhas de identificação e gerenciamento, a detecção pró-ativa de anomalias, a correlação entre os alarmes e os itens mais amplos do gerenciamento de serviço (por exemplo, como oposto do gerenciamento de rede). Embora sejam importantes, esses tópicos merecem um livro dedicado a eles. O leitor pode consultar as referências apresentadas na Seção 9.1.

Exercícios de fixação

Capítulo 9 Questões de revisão

Seção 9.1

1. Por que um administrador de rede necessita de ferramentas de gerenciamento de rede? Descreva cinco cenários.
2. Quais são as cinco áreas de gerenciamento de rede definidas pela ISO?
3. Qual a diferença entre gerenciamento de rede e gerenciamento de serviço?

Seção 9.2

4. Defina os seguintes termos: entidade gerenciadora, dispositivo gerenciado, agente de gerenciamento de rede, MIB e protocolo de gerenciamento de rede.

Seção 9.3

5. Qual é o papel da SMI no gerenciamento de rede?
6. Cite uma diferença importante entre uma mensagem de comando-resposta e uma mensagem trap no SNMP.
7. Quais são os sete tipos de mensagens usados no SNMP?
8. O que significa um 'processador do SNMP'?

Seção 9.4

9. Qual é a finalidade da árvore de identificadores de objetos ASN.1?
10. Qual é o papel da ASN.1 na camada de apresentação nos modelos de referência ISO/OSI?
11. A Internet tem uma camada de apresentação? Se não tiver, como são tratadas as questões de diferenças entre arquiteturas de máquinas — por exemplo, a representação diferente de números inteiros em máquinas diferentes?
12. O que significa codificação TLV?

Problemas

1. Considere as duas maneiras pelas quais ocorrem as comunicações entre uma entidade gerenciadora e um dispositivo gerenciado: modo comando-resposta e trapping. Quais são os prós e os contras dessas duas abordagens, em termos de (1) sobrecarga, (2) tempo de notificação quando ocorrem eventos excepcionais e (3) robustez quanto às mensagens perdidas entre a entidade gerenciadora e o dispositivo gerenciado?
2. Na Seção 9.3 vimos que era preferível transportar mensagens SNMP em datagramas UDP não confiáveis. Na sua opinião, por que os projetistas do SNMP preferiram o UDP ao TCP como protocolo de transporte para o SNMP?
3. Qual é o identificador de objeto ASN.1 para o protocolo ICMP (veja a Figura 9.3)?
4. Suponha que você trabalhe para uma empresa com sede nos Estados Unidos que quer desenvolver sua própria MIB para o gerenciamento de uma linha de produtos. Em que lugar da árvore de identificadores de objetos (veja a Figura 9.3) esse produto seria registrado? (*Dica:* você deve recorrer a alguns RFCs e a outros documentos similares para responder a essa pergunta.)
5. Suponha que a Microsoft desenvolva um novo formato de arquivo para um novo produto de software. Em que lugar da árvore de identificadores de objetos ele seria registrado?

6. Considere a Figura 9.9. Qual seria a codificação BER para {código, 271} {sobrenome, "jackson"}?
7. Considere a Figura 9.9. Qual seria a codificação BER para {código, 296} {sobrenome, "browne"}?

Questões dissertativas

1. Além das analogias com uma usina elétrica e uma cabine de avião, cite uma analogia para um sistema complexo distribuído que precisa ser controlado.
2. Considere o cenário de motivação da Figura 9.1. Na sua opinião, que outras atividades um administrador de rede gostaria de monitorar? Por quê?
3. Leia o RFC 789. Como a queda da ARPAnet ocorrida em 1980 poderia ter sido evitada (ou sua recuperação simplificada) se seus administradores tivessem as ferramentas de gerenciamento de rede existentes hoje?

Entrevista

Jeff Case

Jeff Case é o fundador e diretor técnico da SNMP Research, Inc., empresa líder na produção de padrões da Internet e de produtos para gerenciamento de rede baseados em padrões. Ele é bacharel em educação industrial e engenharia elétrica pela Universidade de Purdue. É também doutor em educação técnica pela Universidade de Illinois, em Urbana-Champaign.

O que o fez se decidir pela especialização em tecnologia de rede?

Sempre fui fascinado por interligar coisas, desde quando era criança e enfiava grampos de cabelo nas tomadas. Essa fascinação se desenvolveu e se transformou em interesse por equipamentos de áudio durante a adolescência, quando eu montava sistemas de som para bandas de rock, cuja potência parecia ser suficiente para pulverizar concreto. Enquanto estudava e trabalhava como técnico de TV e rádio (em geral, equipamentos de áudio), fui mordido pelo bicho do computador e comecei a me interessar por tudo o que era digital, incluindo hardware e software para computadores. Comecei a me interessar pela construção de interfaces entre um componente esquisito e outro componente esquisito. Primeiro montei uma interface entre periféricos e processadores. Mais tarde, uma interface entre dois sistemas. A rede é a interface definitiva. E, até agora, a Internet é a rede definitiva.

Qual foi seu primeiro emprego no setor de computação? O que implicava?

Passei a maior parte de meus primeiros anos como profissional na Universidade de Purdue. Em determinados anos, dava aula para quase todos os cursos do currículo de graduação em eletricidade e em tecnologia da computação. Isso incluía criar novos cursos para os então emergentes tópicos de hardware e software para microprocessadores. Durante um semestre, a classe projetava um computador a partir de chips e o construía na parte laboratorial do curso, com uma equipe trabalhando na CPU, outra no subsistema de memória, outra no subsistema de E/S e assim por diante. No semestre seguinte, escrevíamos o software para o hardware que tínhamos construído.

Foi nessa mesma época que chefiei a área de computação do campus, reportando-me ao reitor como diretor de serviços acadêmicos de computação aos usuários.

Qual parte de seu trabalho lhe apresenta mais desafios?

Ficar atualizado com todas as mudanças, tanto técnicas quanto empresariais. Sou um administrador muito técnico, e está ficando cada vez mais difícil ficar atualizado com os avanços tecnológicos de nosso setor. Meu papel também exige que eu me informe sobre as mudanças no espaço empresarial, como fusões e aquisições.

Em sua opinião, qual é o futuro das redes e da Internet?

Mais, mais, mais. Mais velocidade. Mais ubiqüidade. Mais conteúdo. Mais tensão entre anarquia e governança. Mais spam. Mais movimentos anti-spam. Mais problemas de segurança. Mais soluções de segurança. E, por fim, devemos esperar o inesperado.

Quais pessoas o inspiraram profissionalmente?

Meu falecido pai, que era um homem de negócios bem-sucedido; Dilbert; dr. Vint Cerf, dr. Jon Postel, dr. Marshall Rose e Chuck Davin, que são figuras muito conhecidas no setor da Internet; Bill Seifert, agora um sócio vice-presidente da empresa; dr. Rupert Evans, meu orientador; minha esposa, que trabalha comigo na empresa, e, finalmente, mas não menos importante, Jesus.

Li que o senhor tem uma coletânea notável de 'provérbios'. Quando era professor de ciência da computação, quais deles costumava dizer a seus alunos?

"Um exemplo vale dois livros" (acho que é de Gauss).

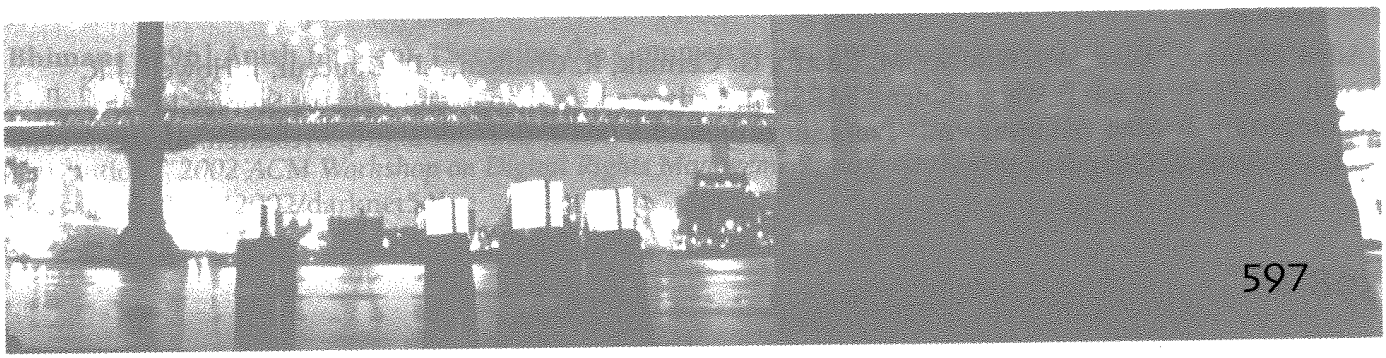
"Às vezes, há uma lacuna entre teoria e prática. A lacuna entre teoria e prática não é tão grande teoricamente quanto a lacuna entre teoria e prática na prática" (não tenho idéia de onde veio esse provérbio).

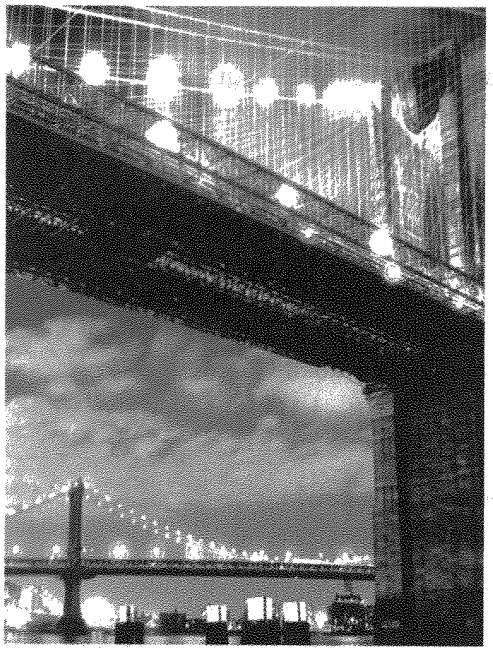
Quais têm sido os maiores obstáculos na criação de padrões para a Internet?

Dinheiro. Política. Egos. Fracassos de liderança.

Qual tem sido o uso mais surpreendente da tecnologia SNMP?

Todos. Na realidade, eu me envolvi no gerenciamento da Internet por causa das minhas necessidades de sobrevivência. Precisava ter algumas ferramentas adequadas para gerenciar a infra-estrutura de rede da minha organização. O sucesso estrondoso que aconteceu foi obra do acaso, porque muitas outras pessoas precisavam resolver problemas semelhantes; foi também sorte, e muito trabalho árduo. O importante é que conseguimos montar a arquitetura correta logo de início.





Referências

Sempre que possível, informamos um URL como referência. Todos os URLs foram validados (e testados) em abril de 2004. Infelizmente, os URLs poderão estar desatualizados. Por favor, consulte a bibliografia atualizada na versão on-line deste livro em inglês (http://www.awl.com/kurose-ross).

Uma nota sobre os pedidos de comentários (*request for comments* — RFCs) da Internet: há uma cópia dos RFCs em diversos sites. Todos os URLs de RFCs a seguir levam ao arquivo RFC do Information Sciences Institute (ISI) mantido pelo Editor de RFC (RFC Editor) da Internet Society (a organização que supervisiona os RFCs). Outros sites de RFC são: http://www.faqs.org/rfcs e http://www.csl.sony.co.jp/rfc/ (localizado no Japão).

Os RFCs da Internet podem ficar desatualizados com a publicação de outros RFCs. Aconselhamos que você acesse os sites relacionados anteriormente para obter informações mais atualizadas. A funcionalidade de busca dos RFCs no ISI (http://www.rfc-editor.org/rfc.html) permitirá que você procure um RFC e apresentará suas atualizações.

[3Com 2004] 3Com Corporation. "Network Interface Cards." http://www.3com.com/products/nics.html

[3GPP 2004] Third Generation Partnership Project. http://www.3gpp.org/

[802.11 Security] The Unofficial 802.11 Security Web Page. http://www.drizzle.com/~aboba/IEEE/

[Abitz 1993] P. Albitz e C. Liu. *DNS and BIND*. Petaluma: O'Reilly & Associates, 1993.

[Abramson 1970] N. Abramson "The Aloha System – Another Alternative for Computer Communications." *Proceedings of Fall Joint Computer Conference, AFIPS Conference*, 1970, p. 37.

[Abramson 1985] N. Abramson. "Development of the ALOHAnet." *IEEE Transactions on Information Theory*, vol. IT-31, n. 3, mar. 1985, p. 119-123.

[Adler 2002] M. Adler. "Tradeoffs in Probabilistic Packet Marking for IP Traceback." *Proceedings of 34th ACM Symposium on Theory of Computing (STOC)*, maio 2002. http://www.cs.umass.edu/~micah/pubs/traceback.ps

[Adya 2004] A. Adya, W.J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J.R. Douceur, J. Howell, J.R. Lorch, M. Theimer e R.P. Wattenhofer. "FARSITE: Federated, Available, and Reliable Storage for an Incompletely Trusted Environment." *Proceedings of the 5th OSDI*, dez. 2002. http://research.microsoft.com/~adya/pubs/osdi2002.pdf

[Ahn 1995] J.S. Ahn, P.B. Danzig, Z. Liu e Y. Yan. "Experience with TCP Vegas: Emulation and Experiment". *Proceedings of ACM SIGCOMM '95*. Boston, ago. 1995, p. 185-195. http://www.acm.org/sigcomm/sigcomm95/papers/ahn.html

[Akamai 2004] Akamai. http://www.akamai.com

[Alvestrand 1997] H. Alvestrand. "Object Identifier Registry." http://www.alvestrand.no/harald/objectid/top.html

[Aprisma 2004] Aprisma. http://www.aprisma.com/

[ARIN 1996] ARIN. "IP Allocation Report." ftp://rs.arin.net/netinfo/ip_network_allocations

[ASO-ICANN 2004] The Address Supporting Organization home page. http://www.aso.icann.org

[Ash 1998] G.R. Ash. *Dynamic routing in telecommunications networks*. McGraw Hill: Nova York, 1998.

[AT&T SLM 2004] AT&T Business. "AT&T Enterprise Hosting Services Service Guide." http://www.att.com/abs/serviceguide/docs/eh_sg.pdf

[ATM Forum 2004] ATM Forum. http://www.atmforum.com/

[Ayanoglu 1995] E. Ayanoglu, S. Paul, T.F. La Porta, K.K. Sabnani e R.D. Gitlin. "AIRMAIL: A Link-Layer Protocol for Wireless Networks." *ACM ACM/Baltzer Wireless Networks Journal*, 1, fev. 1995, p. 47-60. http://www.bell-labs.com/user/sanjoy/airmail.ps.Z

[Bakre 1995] A. Bakre e B.R. Badrinath. "I-TCP: Indirect TCP for Mobile Hosts." *Proceedings of the 15th International Conf. on Distributed Computing Systems (ICDCS)*, maio 1995, p. 136-143. ftp://paul.rutgers.edu/pub/badri/itcp-tr314.ps.Z

[Balakrishnan 1995] H. Balakrishnan, S. Seshan e R.H. Katz. "Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks." *ACM Wireless Networks*, 1, n. 4, dez. 1995. http://nms.lcs.mit.edu/~hari/papers/winet.ps

[Balakrishnan 1997] H. Balakrishnan, V. Padmanabhan, S. Seshan e R. Katz. "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links." *IEEE/ACM Transactions on Networking* 5, n. 6, dez. 1997. http://nms.lcs.mit.edu/~hari/papers/ton.ps

[Baptista 2003] A. Baptista, T. Leen, Y. Zhang, A. Chawla, D. Maier, W. Feng, W. Feng, J. Walpole, C. Silva e J. Freire. "Environmental Observation and Forecasting Systems: Vision, Challenges and Successes of a Prototype." In: R.A. Meyers (ed.) *Encyclopedia of Physical Science and Technology*, 3 ed. Academic Press, vol. 5, p. 565-581.

[Baran 1964] P. Baran. "On Distributed Communication Networks." *IEEE Transactions on Communication Systems*, mar. 1964. Relatório da Rand Corporation Technical com o mesmo título. Memorando RM-3420-PR, 1964. http://www.rand.org/publications/RM/RM3420/

[BBC 2001] BBC news online. "A Small Slice of Design", abr. 2001. http://news.bbc.co.uk/1/low/sci/tech/1264205.stm

[Berners-Lee 1989] T. Berners-Lee, Cern. "Information Management: A Proposal", mar. 1989, maio 1990. http://www.w3.org/History/1989/proposal.html

[Berners-Lee 1994] T. Berners-Lee, R. Cailliau, A. Luotonen, H. Frystyk Nielsen e A. Secret. "The World-Wide Web." *Communications of the ACM*, vol. 37, n. 8, ago. 1994, p. 76-82.

[Bertsekas 1991] D. Bertsekas e R. Gallagher. *Data Networks*, 2 ed. Englewood Cliffs: Prentice Hall, 1991.

[Bhagwat 2003] P. Bhagwat, B. Raman e D. Sanghi. "Turning 802.11 Inside Out." *Proceedings of the 2003 ACM Hotnets II Workshop*. Cambridge, nov. 2003. http://nms.lcs.mit.edu/HotNets-II/papers/inside-out.pdf

[Bhimani 1996] Anish Bhimani. "Securing the Commercial Internet." *Communications of the ACM*, vol. 39, n. 6, mar. 1996, p. 29-35.

[Biddle 2003] P. Biddle, P. England, M. Peinado e B. Willman. "The Darknet and the Future of Content Distribution." *2002 ACM Workshop on Digital Rights Management*, nov. 2002, Washington, D.C. http://crypto.stanford.edu/DRM2002/darknet5.doc

[Biersack 1992] E.W. Biersack. "Performance Evaluation of Forward Error Correction in ATM networks." *Proceedings of ACM SIGCOMM'92*. Baltimore, 1992, p. 248-257. http://www.acm.org/pubs/articles/proceedings/comm/144179/p248-biersack/p248-biersack.pdf

[BIND 2004] Internet Software Consortium page on BIND. http://www.isc.org/bind.html

[Bisdikian 2001] C. Bisdikian. "An Overview of the Bluetooth Wireless Technology." *IEEE Communications Magazine*, n. 12, dez. 2001, p. 86-94.

[Bishop 2003] M. Bishop. *Computer Security: Art and Science*. Boston: Addison Wesley, 2003.

[Black 1995] U. Black. *ATM Volume I: Foundation for Broadband Networks*. Prentice Hall, 1995.

[Black 1997] U. Black. *ATM, Volume II: Signaling in Broadband Networks*. Prentice Hall, 1997.

[Blaze 1996] M. Blaze, W. Diffie, R. Rivest, B. Schneier, T. Shimomura, E. Thompson e M. Weiner. "Minimal Key Lengths for Symmetric Ciphers to Provide Adequate Commercial Security." http://www.counterpane.com/keylength.html

[Bluetooth 2002] R. Morrow. *Bluetooth: Operation and Use*. Nova York: McGraw-Hill, 2002.

[Bochman 1984] G.V. Bochmann e C.A. Sunshine. "Formal Methods in Communication Protocol Design." *IEEE Transactions on Communications*, vol. COM-28, n. 4, abril 1980, p. 624-631.

[Bolot 1994] J-C. Bolot e T. Turletti. "A Rate Control Scheme for Packet Video in the Internet." *Proceedings of IEEE Infocom*, 1994, p. 1216-1223. ftp://ftp-sop.inria.fr/rodeo/bolot/94.Video_control.ps.gz

[Bolot 1996] J-C. Bolot e Andreas Vega-Garcia. "Control Mechanisms for Packet Audio in the Internet." *Proceedings of IEEE Infocom*, 1996, p. 232-239. ftp://ftp-sop.inria.fr/rodeo/bolot/96.Audio_ctl.ps.gz

[Boutremans 2002] C. Boutremans, G. Iannaccone e C. Diot. "Impact of Link Failures on VoIP Performance." *12th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, Miami, maio 2002. http://ipmon.sprint.com/pubs_trs/pubs/gianluca/voip.pdf

[Bradner 1996] S. Bradner e A. Mankin. *IPng: Internet Protocol Next Generation*. Reading: Addison Wesley, 1996.

[Brakmo 1995] L. Brakmo e L. Peterson. "TCP Vegas: End to End Congestion Avoidance on a Global Internet." *IEEE Journal of Selected Areas in Communications*, vol. 13, n. 8, out. 1995, p. 1465-1480. ftp://ftp.cs.arizona.edu/xkernel/Papers/jsac.ps.Z

[Brodnik 1997] A. Brodnik, S. Carlsson, M. Degemark e S. Pink. "Small Forwarding Tables for Fast Routing Lookups." *Proceedings of ACM SIGCOMM '97*. Cannes, out. 1997, p. 3-15. http://www.acm.org/sigs/sigcomm/sigcomm97/papers/p192.html

[Brown 1997] K. Brown e S. Singh. "M-TCP: TCP for Mobile Cellular Networks." *ACM CCR 27*, n. 5, 1997. http://www.cs.pdx.edu/~singh/ftp/mtcp.ps.gz

[Bryant 1988] B. Bryant. "Designing an Authentication System: A Dialogue in Four Scenes." http://web.mit.edu/kerberos/www/dialogue.html

[Bush 1945] V. Bush. "As We May Think." *The Atlantic Monthly*, jul. 1945. http://www.theatlantic.com/unbound/flashbks/computer/bushf.htm

[Byers 1998] J. Byers, M. Luby, M. Mitzenmacher e A. Rege. "A Digital Fountain Approach to Reliable Distribution of Bulk Data." *Proceedings of ACM SIGCOMM '98*. Vancouver, ago. 1998, p. 56-67. http://www.acm.org/sigcomm/sigcomm98/tp/abs_05.html

[Cablelabs 2004] CableLabs homepage. http://www.cablelabs.com

[Caldwell 2004] C. Caldwell. The Prime Pages. http://www.utm.edu/research/primes/prove

[Cardwell 2000] N. Cardwell, S. Savage e T. Anderson. "Modeling TCP Latency." *Proceedings of the 2000 IEEE Infocom Conference*. Tel-Aviv, mar. 2000. http://www.cs.ucsd.edu/users/savage/papers/Infocom2000tcp.ps

[CASA 2004] Center for Collaborative Adaptive Sensing of the Atmosphere. http://www.casa.umass.edu

[Casner 1992] S. Casner e S. Deering. "First IETF Internet Audiocast." *ACM SIGCOMM Computer Communications Review*, vol. 22, n. 3, jul. 1992, p. 92-97. http://citeseer.nj.nec.com/casner92first.html

[Ceiva 2004] Ceiva homepage. http://www.ceiva.com/

[CENS 2004] Center for Embedded Network Sensing. http://www.cens.ucla.edu/

[Cerf 1974] V. Cerf e R. Kahn. "A Protocol for Packet Network Interconnection." *IEEE Transactions on Communications Technology*, vol. COM-22, n. 5, p. 627-641.

[CERT 1999-04] CERT. "Advisory CA-1999-04: Melissa Macro Virus." http://www.cert.org/advisories/CA-1999-04.html

[CERT 2001-09] CERT. "Advisory 2001-09: Statistical Weaknesses in TCP/IP Initial Sequence Numbers." http://www.cert.org/advisories/CA-2001-09.html

[CERT 2001-19] CERT. "Advisory CA-2001-19: "Code Red" Worm Exploiting Buffer Overflow In IIS Indexing Service DLL." http://www.cert.org/advisories/CA-2001-19.html

[CERT 2003-04] CERT. "CERT Advisory CA-2003-04 MS-SQL Server Worm." http://www.cert.org/advisories/CA-2003-04.html

[CERT 2004] CERT Coordination Center. http://www.cert.org/advisories

[CERT Filtering 2002] CERT. "Packet Filtering for Firewall Systems." http://www.cert.org/tech_tips/packet_filtering.html

[CERT Smurf 1998] CERT(r). "Advisory CA-98.01: Smurf IP Denail-of-Service Attacks." http://www.cert.org/advisories/CA-1998-01.html

[Cert SYN 1996] CERT. "Advisory CA-96.21: TCP SYN Flooding and IP Spoofing Attacks." http://www.cert.org/advisories/CA-1998-01.html

[CERT 2004 Summaries] CERT. "CERT Summaries." http://www.cert.org/summaries/

[Chao 2001] H.J. Chao, C. Lam e E. Oki. *Broadband Packet Switching Technologies – A Practical Guide to ATM Switches and IP Routers*. John Wiley & Sons, 2001.

[Chapman 1992] B. Chapman. "Network (In)Security Through Packet Filtering." *Third UNIX Security Symposium*, sponsored by USENIX Association, Baltimore, MD, 1992. http://www.greatcircle.com/pkt_filtering.html

[Checkpoint 2004] Checkpoint Web site. http://www.checkpoint.com

[Chen 2000] G. Chen e D. Kotz. "A Survey of Context-Aware Mobile Computing Research." *Technical Report TR2000-381*, Dept. of Computer Science, Dartmouth College, nov. 2000. http://www.cs.dartmouth.edu/~dfk/papers/chen:survey-tr.pdf

[Cheswick 2000] B. Cheswick, H. Burch e S. Branigan. "Mapping and Visualizing the Internet." *Proceedings of the 2000 Usenix Conference*. San Diego, jun. 2000. http://www.usenix.org/publications/library/proceedings/usenix2000/general/full_papers/cheswick/cheswick_html/mapping.html

[Chiu 1989] D. Chiu e R. Jain. "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks." *Computer Networks and ISDN Systems*, vol. 17, n. 1, p. 1-14. http://www.cis.ohio-state.edu/~jain/papers/cong_av.htm

[Christiansen 2001] M. Christiansen, K. Jeffay, D. Ott e F.D. Smith. "Tuning Red for Web Traffic." *IEEE/ACM Transactions on Networking*, vol. 9, n. 3, jun. 2001, p. 249-264. http://www.cs.unc.edu/~jeffay/papers/IEEE-ToN-01.pdf

[Chu 2000] Y. Chu, S. Rao e H. Zhang. "The Case for End System Multicast." *Proceedings of ACM SIGMETRICS 2000*. Santa Clara, ago. 2000. http://www.cs.cmu.edu/~sanjay/Papers/sigmetrics-2000.ps.gz

[Cisco 12000 1998] Cisco Systems. "Cisco 12000 Series Gigabit Switch Routers."

http://www.cisco.com/univercd/cc/td/doc/pcat/12000.htm

[Cisco 8500 1999] Cisco Systems Inc. "Catalyst 8500 Campus Switch Router Architecture." http://www.cisco.com/univercd/cc/td/doc/product/l3sw/8540/rel_12_0/w5_6f/softcnfg/1cfg8500.pdf

[Cisco CiscoWorks 2000] Cisco Systems, Cisco Works 2000 homepage. http://www.cisco.com/warp/public/cc/pd/wr2k/index.shtml

[Cisco NAT 2004] Cisco Systems Inc. "How NAT Works." http://www.cisco.com/warp/public/556/nat-cisco.shtml

[Cisco QoS 2002] Cisco Systems Inc. "Advanced QoS Services for the Intelligent Internet." http://www.cisco.com/warp/public/cc/pd/iosw/ioft/ioqo/tech/qos_wp.htm

[Cisco Queue 1995] Cisco Systems Inc. "Interface Queue Management." http://www.cisco.com/warp/public/614/16.html

[Cisco Security 2004] Cisco Systems Inc. "Why You Need a Firewall." http://www.cisco.com/en/US/products/sw/secursw/ps743/products_user_guide_chapter09186a008007f303.html

[Cisco Switches 1999] Cisco Systems Inc. "Cisco Catalyst 1900/2820 – Affordable Switching Solutions." http://www.cisco.com/warp/public/cc/pd/si/index.shtml

[Cisco Switches 2004] Cisco Systems Inc. "Cisco Switches." http://www.cisco.com/warp/public/cc/pd/si/index.shtml

[CISN 2004] California Integrated Seismic Network. http://www.cisn.org/

[Claffy 1998] K. Claffy, G. Miller e K. Thompson. "The Nature of the Beast: Recent Traffic Measurements from an Internet Backbone." *Proceedings of Inet '98*. Geneva, jul. 1998. http://www.caida.org/outreach/papers/1998/Inet98/

[Clark 1988] D. Clark. "The Design Philosophy of the DARPA Internet Protocols." *Proceedings of ACM SIGCOMM'88*. Stanford, vol. 18, n. 4, ago. 1988. http://www.acm.org/sigcomm/ccr/archive/1995/jan95/ccr-9501-clark.html

[Clarke 2002] I. Clarke, T.W. Hong, S.G. Miller, O. Sandberg e B. Wiley. "Protecting Free Expression Online with Freenet." *IEEE Internet Computing*, jan./fev. 2002, p. 40-49. http://freenet.sourceforge.net/papers/freenet-ieee.pdf

[Cnet 2000] Cnet news.com. "Leading Web Sites Under Attack." http://news.com.com/2100-1017236683.html

[Cohen 1977] D. Cohen. "Issues in Transnet Packetized Voice Communication." *Proceedings of the Fifth Data Communications Symposium*. Snowbird, set. 1977, p. 6-13.

[Cookie Central 2004] Cookie Central homepage. http://www.cookiecentral.com

[Cormen 2001] T.H. Cormen. *Introduction to Algorithms*, 2 ed. Cambridge: MIT Press, 2001.

[Crow 1997] B. Crow, I. Widjaja, J. Kim e P. Sakai. "IEEE 802.11 Wireless Local Area Networks." *IEEE Communications Magazine*, set. 1997, p. 116-126.

[Crowcroft 1995] J. Crowcroft, Z. Wang, A. Smith e J. Adams. "A Comparison of the IETF and ATM Service Models." *IEEE Communications Magazine*, nov./dez. 1995, p. 12-16. http://citeseer.nj.nec.com/crowcroft95rough.html

[Crowcroft 1999] J. Crowcroft, M. Handley e I. Wakeman. *Internetworking Multimedia*. São Francisco: Morgan-Kaufman, 1999.

[Cusumano 1998] M.A. Cusumano e D.B. Yoffie. *Competing on Internet Time: Lessons from Netscape and its Battle with Microsoft*. Nova York: Free Press, 1998.

[Daemen 2000] J. Daemen e V. Rijmen. "The Block Cipher Rijndael". In: J.J. Quisquater, B. Schneier (eds.) *Smart Card Research and Applications*. LNCS 1820, Springer-Verlag, 2000, p. 288-296.

[Daigle 1991] J.N. Daigle. *Queuing Theory for Telecommunications*. Reading, MA: Addison Wesley, 1991.

[Dalal 1978] Y. Dalal e R. Metcalfe. "Reverse Path Forwarding of Broadcast Packets." *Communications of the ACM*, vol. 21, n. 12, dez. 1978, p. 1040-1048.

[Danielyan 2001] E. Danielyan. "Goodbye DES, Welcome AES." *Internet Protocol Journal*, vol. 4, n. 2, jun. 2001. http://www.cisco.com/en/US/about/ac123/ac147/ac174/about_cisco_ipj_archive_issues_list.html

[DEC 1990] Digital Equipment Corporation. "In Memoriam: J.C.R. Licklider 1915–1990." *SRC Research Report 61*, ago. 1990. http://www.memex.org/licklider.pdf

[DeClercq 2002] J. DeClercq e O. Paridaens. "Scalability Implications of Virtual Private Networks." *IEEE Communications Magazine*, vol. 40, n. 5, maio 2002, p. 151-157.

[Deering 1990] S. Deering e D. Cheriton. "Multicast Routing in Datagram Internetworks and Extended LANs." *ACM Transactions on Computer Systems*, vol. 8, n. 2, 1990, p. 85-110.

[Deering 1996] S. Deering, D. Estrin, D. Faranacci, V. Jacobson, C. Liu e L. Wei. "The PIM Architecture for Wide Area Multicasting." *IEEE/ACM Transactions on Networking*, vol. 4, n. 2, abr. 1996, p. 153-162.

[Demers 1990] A. Demers, S. Keshav e S. Shenker. "Analysis and Simulation of a Fair Queuing Algorithm." *Internetworking: Research and Experience*, vol. 1, n. 1, 1990, p. 3-26.

[Denning 1997] D. Denning (editor) e P. Denning (prefácio). *Internet Besieged: Countering Cyberspace Scofflaws*. Reading: Addison Wesley, 1997.

[dhc 2004] IETF Dynamic Host Configuration working group. http://www.ietf.org/html.charters/dhc-charter.html

[Dialpad 2004] Dialpad homepage. http://www.dialpad.com

[Diffie 1976] W. Diffie e M.E. Hellman. "New Directions in Cryptography." *IEEE Transactions on Information Theory*, vol. IT-22, 1976, p. 644-654.

[Diffie 1998] W. Diffie e S. Landau. *Privacy on the Line, The Politics of Wiretapping and Encryption*. Cambridge: MIT Press, 1998.

[Digital Signature 2004] Digital Signature Trust Company. http://www.trustdst.com/

[Diot 2000] C. Diot, B.N. Levine, B. Lyles, H. Kassem e D. Balensiefen. "Deployment Issues for the IP Multicast Service and Architecture." *IEEE Network*, vol. 14, n. 1, jan./fev. 2000, p. 78-88. http://signl.cs.umass.edu/pubs/brian.ieeenetwork00.ps.gz

[Dodge 2004] M. Dodge. "An Atlas of Cyberspaces." http://www.cybergeography.org/atlas/isp_maps.html

[Donahoo 2000] M. Donahoo e K. Calvert. *TCP/IP Sockets in C: Practical Guide for Programmers*. Morgan Kaufman, 2000.

[Dornan 2001] A. Dornan. *The Essential Guide to Wireless Communications Applications: From Cellular Systems to WAP and M-Commerce*. Upper Saddle River: Prentice Hall, 2001.

[Droms 1999] R. Droms e T. Lemon. *The DHCP Handbook*. Indianapolis: Macmillan Technical Publishing, 1999.

[DSL 2004] DSL Forum. http://www.dslforum.org/

[EFF 1999] Electronic Frontier Foundation. "Frequently Asked Questions (FAQ) About the Electronic Frontier Foundation's DES Cracker Machine." http://www.eff.org/pub/Privacy/Crypto/Crypto_misc/DESCracker/HTML/19980716_eff_des_faq.html

[Elgamal 2001] A. Elgamal, F. Seible, F. Vernon, M. Trivedi e M. Fraser. "On-Line Structural Monitoring and Data Management." *Proceedings, 6th Seismic Research Workshop*. California Department of Transportation, Sacramento, jun. 12-13, 2001. http://www.calit2.net/eci/caltrans_health_monitoring_paper.pdf

[Ellis 1987] H. Ellis. "The Story of Non-Secret Encryption." http://www.cesg.gov.uk/site/publications/media/ellis.pdf

[Ericsson EDGE 2004] Ericsson. "EDGE: Introduction of High-Speed Data in GSM/GPRS Networks." http://www.ericsson.com/products/white_papers_pdf/edge_wp_technical.pdf

[Estrin 1997] D. Estrin, M. Handley, A. Helmy, P. Huang e D. Thaler. "A Dynamic Bootstrap Mechanism for Rendezvous-based Multicast Routing." *Proceedings of IEEE Infocom '98*. Nova York, abr. 1998. http://ceng.usc.edu/~helmy/infocom-bootstrap-99.pdf

[Estrin 1998b] D. Estrin, V. Jacobson, D. Farinacci, L. Wei, S. Deering, M. Handley, D. Thaler, Ching-Gung Liu, Puneet Sharma, A. Helmy. "Protocol Independent Multicast-Sparse Mode (PIM-SM): Motivation and Architecture." Trabalho em desenvolvimento. http://netweb.usc.edu/pim/pimsm/PIM-Arch.ps.gz

[Estrin 2002] D. Estrin, D. Culler e K. Pister. "Connecting the Physical World with Pervasive Networks." *IEEE Pervasive Computing*, 1, 1, jan./mar. 2002.

[Ethereal 2004] Ethereal homepage. http://www.ethereal.com

[Faloutsos 1999] C. Faloutsos, M. Faloutsos e P. Faloutsos. "What Does the Internet Look Like? Empirical Laws of the Internet Topology." *Proceedings of ACM SIGCOMM 1999*. Boston, set. 1999.

[Feamster 2004] N. Feamster, J. Winick e J. Rexford. "A Model for BGP Routing for Network Engineering." *Proceedings of 2004 ACM Sigmetrics*. Nova York, jun. 2004. http://www.research.att.com/~jrex/papers/whatifatron.pdf

[Feldmeier 1988] D. Feldmeier. "Improving Gateway Performance with a Routing Table Cache." *Proceedings of 1988 IEEE Infocom Conference*. Nova Orleans, mar. 1988.

[Feldmeier 1995] D. Feldmeier. "Fast Software Implementation of Error Detection Codes." *IEEE/ACM Transactions on Networking*, vol. 3, n. 6, dez. 1995, p. 640-652.

[FIPS 1995] Federal Information Processing Standard. "Secure Hash Standard." *FIPS Publication 180-1*. http://www.itl.nist.gov/fipspubs/fip180-1.htm

[FIPS-46-1 1988] US National Bureau of Standards. "Data Encryption Standard." *Federal Information Processing Standard (FIPS) Publication*, 46-1, jan. 1988. http://www.itl.nist.gov/fipspubs/fip46-2.htm

[Fletcher 1982] J.G. Fletcher. "An Arithmetic Checksum for Serial Transmissions." *IEEE Transactions on Communications*, vol. 30, n. 1, jan. 1982, p. 247-253.

[Floyd 1999] S. Floyd e K. Fall. "Promoting the Use of End-to-End Congestion Control in the Internet." *IEEE/ACM Transactions on Networking*, vol. 6, n. 5, out. 1998, p. 458-472. http://www.icir.org/floyd/end2end-paper.html

[Floyd 2000] S. Floyd, M. Handley, J. Padhye e J. Widmer. "Equation-Based Congestion Control for Unicast Applications." *Proceedings 2000 ACM Sigcomm Conference*. Estocolmo, ago. 2000. http://www.icir.org/tfrc/tcp-friendly.pdf

[Floyd 2001] S. Floyd. "A Report on Some Recent Developments in TCP Congestion Control." *IEEE Communications Magazine*, abr. 2001. http://www.aciri.org/floyd/papers/report_Jan01.pdf

[Floyd 2004] S. Floyd. "References on RED (Random Early Detection) Queue Management." http://www.icir.org/floyd/red.html

[Floyd Synchronization 1994] S. Floyd e V. Jacobson. "Synchronization of Periodic Routing Messages." *IEEE/ACM Transactions on Networking*, vol. 2, n. 2, abr. 1997, p. 122-136. http://www.aciri.org/floyd/papers/sync_94.ps.Z

[Floyd TCP 1994] S. Floyd. "TCP and Explicit Congestion Notification." *ACM Computer Communication Review*, vol. 24, n. 5, out. 1994, p. 10-23. http://www.aciri.org/floyd/papers/tcp_ecn.4.ps.Z

[Fluhrer 2001] S. Fluhrer, I. Mantin e A. Shamir. "Weaknesses in the Key Scheduling Algorithm of RC4." *Eighth Annual Workshop on Selected Areas in Cryptography*, Toronto, ago. 2002. http://www.drizzle.com/~aboba/IEEE/rc4_ksaproc.pdf

[Fortz 2000] B. Fortz e M. Thorup. "Internet Traffic Engineering by Optimizing OSPF Weights." *Proceedings of 2000 IEEE Infocom*. http://www.ieee-infocom.org/2000/papers/165.ps

[Fortz 2002] B. Fortz, J. Rexford e M. Thorup. "Traffic Engineering with Traditional IP Routing Protocols." *IEEE Communication Magazine*, out. 2002. http://www.research.att.com/~jrex/papers/ieeecomm02.ps

[Foster 2002] I. Foster. "The Grid: A New Infrastructure for 21st Century Science." *Physics Today*, vol. 55, n. 2, 2002, p. 42-47. http://www.aip.org/pt/vol-55/iss-2/p42.html.

[Freephone 2004] "Freephone: Why Use the Plain Old Telephone When You Can Get so Much Better on the Internet?" http://www-sop.inria.fr/rodeo/fphone/

[Friedman 1999] T. Friedman e D. Towsley "Multicast Session Membership Size Estimation." *Proceedings of the IEEE Infocom '99*. Nova York, mar. 1999. ftp://gaia.cs.umass.edu/pub/Friedman99_Infocom99.ps.gz

[Frost 1994] J. Frost. "BSD Sockets: A Quick and Dirty Primer." http://world.std.com/~jimf/papers/sockets/sockets.html

[Gallager 1983] R.G. Gallager, P.A. Humblet e P.M. Spira. "A Distributed Algorithm for Minimum Weight-Spanning Trees." *ACM Trans. on Programming Languages and Systems*, vol. 1, n. 5, jan. 1983, p. 66-77.

[Gao 2001] L. Gao e J. Rexford. "Stable Internet Routing Without Global Coordination." *IEEE/ACM Trans. Networking*, vol. 9, n. 6, dez. 2001, p. 681-692. http://www.research.att.com/~jrex/papers/sigmetrics00. long.pdf

[Garces-Erce 2003] L. Garces-Erce, K.W. Ross, E. Biersack, P. Felber e G. Urvoy-Keller. "TOPLUS: Topology Centric Lookup Service." *Fifth International Workshop on Networked Group Communications (NGC'03)*, Munique, set. 2003. http://cis.poly.edu/~ross/papers/TOPLUS.pdf

[Gartner 2003] F.C. Gartner. "A Survey of Self-Stabilizing Spanning-Tree Construction Algorithms." *Technical Report IC/2003/38*, Swiss Federal Institute of Technology (EPFL), School of Computer and Communication Sciences, 10 jun. 2003. http://ic2.epfl.ch/publications/documents/IC_TECH_REPORT_200338.pdf.

[Gauthier 1999] L. Gauthier, C. Diot e J. Kurose. "End-to-end Transmission Control Mechanisms for Multiparty Interactive Applications on the Internet." *Proceedings of IEEE Infocom '99*. Nova York, abr. 1999. ftp://ftp.sprintlabs.com/diot/infocom99-mimaze.zip

[Giacopelli 1990] J. Giacopelli, M. Littlewood e W.D. Sincoskie. "Sunshine: A High Performance Self-Routing Broadband Packet Switch Architecture." *1990 International Switching Symposium*. Uma versão ampliada desse documento pode ser encontrada em *IEEE J. Sel. Areas in Commun.*, vol. 9, n. 8, out. 1991, p. 1289-1298.

[Girard 1990] A. Girard. *Routing and Dimensioning in Circuit-Switched Networks*. Reading: Addison Wesley, 1990.

[Glitho 1995] R. Glitho e S. Hayes (eds.). "Special Issue on Telecommunications Management Network." *IEEE Communications Magazine*, vol. 33, n. 3, mar. 1995.

[Glitho 1998] R. Glitho. "Contrasting OSI Systems Management to SNMP and TMN." *Journal of Network and Systems Management*, vol. 6, n. 2, jun. 1998, p. 113-131.

[Gnutella 2004] "The Gnutella Protocol Specification, v0.4." http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf

[Goodman 1997] D.J. Goodman. *Wireless Personal Communications Systems*. Prentice-Hall, 1997.

[Goodman 1997b] D. Goodman (Chair). *The Evolution of Untethered Communications*. Washington: National Academy Press, dez. 1997. http://www.nap.edu/readingroom/books/evolution/index.html

[Goralski 1999] W. Goralski. *Frame Relay for High-Speed Networks*. Nova York: John Wiley, 1999.

[Goralski 2001] W. Goralski. *Optical Networking and WDM*. Berkeley: Osborne/McGraw-Hill, 2001.

[Griffin 2002] T. Griffin. "Interdomain Routing Links." http://www.research.att.com/~griffin/interdomain.html

[Gummadi 2003] K.P. Gummadi, R.J. Dunn, S. Saroiu, S.D. Gribble, H.M. Levy e J. Zahorjan. "Measurement, Modeling, and Analysis of a Peer-to-Peer File-Sharing Workload." *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP-19)*, out. 2003. http://www.cs.washington.edu/homes/tzoompy/publications/sosp/2003/abstract.html

[Gupta 1998] P. Gupta, S. Lin e N. McKeown. "Routing Lookups in Hardware at Memory Access Speeds." *Proceedings of IEEE Infocom 1998*. São Francisco, abr. 1998, p. 1241-1248. http://tiny-tera.stanford.edu/~nickm/papers/Infocom98_lookup.pdf

[Gupta 2001] P. Gupta e N. McKeown. "Algorithms for Packet Classification." *IEEE Network Magazine*, vol. 15, n. 2, mar./abr. 2001, p. 24-32. http://klamath.stanford.edu/~pankaj/paps/ieeenetwork_tut_01.pdf

[Halabi 2000] S. Halabi. *Internet Routing Architectures*, 2 ed. Cisco Press, 2000.

[Hamada 1997] T. Hamada, H. Kamata e S. Hogg. "An Overview of the TINA Management Architecture." *Journal of Network and Systems Management*, vol. 5, n. 4, dez. 1997, p. 411-435.

[Heidemann 1997] J. Heidemann, K. Obraczka e J. Touch. "Modeling the Performance of HTTP over Several Transport Protocols." *IEEE/ACM Transactions on Networking*, vol. 5, n. 5, out. 1997, p. 616-630. http://www.isi.edu/~johnh/PAPERS/Heidemann96a.html

[Held 2001] G. Held. *Data Over Wireless Networks: Bluetooth, WAP, and Wireless LANs*. McGraw-Hill, 2001.

[Hersent 2000] O. Hersent, D. Gurle e J-P. Petit. *IP Telephony: Packet-Based Multimedia Communication Systems*." Edinburgh: Pearson Education Limited, 2000.

[Hinden 2004] R. Hinden. "IP Next Generation (IP ng)." http://playground.sun.com/pub/ipng/html/ipng-main.html

[Holbrook 1999] H. Holbrook e D. Cheriton. "IP Multicast Channels: EXPRESS Support for Large-Scale Single-Source Applications." *Proceedings of ACM SIGCOMM '99*. Boston, ago. 1999. http://www.acm.org/sigs/sigcomm/sigcomm99/papers/session2-3.html

[Hollot 2002] C.V. Hollot, V. Misra, D. Towsley e W. Gong. "Analysis and Design of Controllers for AQM Routers Supporting TCP Flows." *IEEE Transactions on Automatic Control*, vol. 47, n. 6, jun. 2002, p. 945-959. http://www1.cs.columbia.edu/~misra/pubs/TAC_special.pdf

[Huang 2002] C. Haung, V. Sharma, K. Owens e V. Makam. "Building Reliable MPLS Networks Using a Path Protection Mechanism." *IEEE Communications Magazine*, vol. 40, n. 3, mar. 2002, p. 156-162.

[Huitema 1998] C. Huitema. *IPv6: The New Internet Protocol*, 2 ed. Englewood Cliffs: Prentice Hall, 1998.

[Huston 1999a] G. Huston. "Interconnection, Peering, and Settlements – Part I." *The Internet Protocol Journal*, vol. 2, n. 1, mar. 1999. http://www.cisco.com/warp/public/759/ipj_2-1/ipj_2-1_ps1.html

[Huston 1999b] G. Huston. "Interconnecting, Peering, and Settlements – Part II." *The Internet Protocol Journal*, vol. 2, n. 2, jun. 1999. http://www.cisco.com/warp/public/759/ipj_2-2/ipj_2-2_ps1.html

[Huston 2001] G. Huston. "Analyzing the Internet BGP Routing Table." *The Internet Protocol Journal*, vol. 4, n. 1, mar. 2001. http://www.cisco.com/warp/public/759/ipj_4-1/ipj_4-1_bgp.html

[IAB 2004] Internet Architecture Board. http://www.iab.org/iab/

[IANA 2004] Internet Assigned Number Authority homepage. http://www.iana.org/

[ICANN 2004] The Internet Corporation for Assigned Names and Numbers. http://www.icann.org

[IEC Optical 2003] IEC Online Education. "Optical Access." http://www.iec.org/online/tutorials/opt_acc/

[IEEE 802 2004] "IEEE 802 LAN/MAN Standards Committee." http://www.ieee802.org/

[IEEE 802.11 1999] IEEE 802.11, 1999 Edition (ISO/IEC 8802-11: 1999) IEEE Standards for Information Technology –Telecommunications and Information Exchange Between Systems – Local and Metropolitan Area Network – Specific Requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification. http://standards.ieee.org/getieee802/download/802.11-1999.pdf

[IEEE 802.15 2004] IEEE 802.15 Working Group for WPAN. http://grouper.ieee.org/groups/802/15/

[IEEE 802.1X] IEEE Std 802.1X-2001 Port-Based Network Access Control. http://standards.ieee.org/reading/ieee/std_public/description/lanman/802.1x-2001_desc.html

[IETF 2004] Internet Engineering Task Force homepage. http://www.ietf.org

[IETF dnsext 2004] IETF DNS Extensions Working Group. http://www.ietf.org/html.charters/dnsext-charter.html

[IMAP 2002] The IMAP Connection. http://www.imap.org/

[Interlinknetworks 2004] Interlinknetworks. "Introduction to 802.1x for Wireless Local Area Networks." http://www.interlinknetworks.com/resource/wp5-1-1.htm

[Interlinknetworks 2004] Internlinknetworks. "Introduction to 802.1x for Wireless Local Area Networks." http://www.interlinknetworks.com/resource/wp5-1-1.htm

[Ioannidis 2000] S. Ioannidis, A. Keromytis, S. Bellovin e J.M. Smith. "Implementing a Distributed Firewall." *Proceedings of the ACM Computer and Communications Security (CCS) 2000*. Atenas, p. 190-199. http://www.cis.upenn.edu/~strongman/Papers/df.pdf

[Iren 1999] S. Iren, P. Amer e P. Conrad. "The Transport Layer: Tutorial and Survey." *ACM Computing Surveys*, vol 31, n. 4, dez. 1999. http://www.cis.udel.edu/~amer/PEL/survey/

[ISC 2004] Internet Systems Consortium. http://www.isc.org

[ISO 1987] International Organization for Standardization. "Information processing systems – Open Systems Interconnection." *International Standard 8824*, dez. 1987. http://asn1.elibel.tm.fr/en/standards/index.htm

[ISO X.680 1998] International Organization for Standardization. "X.680: ITU-T Recommendation X.680 (1997) | ISO/IEC 8824-1:1998, Information Technology – Abstract Syntax Notation One (ASN.1): Specification of Basic Notation." http://asn1.elibel.tm.fr/en/standards/index.htm

[ITU 2000] International Telecommunication Union. "Recommendation X.509 (11/93) Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks." http://www.itu.int/rec/recommendation.asp?type=items&lang=e&parent=T-REC-X.509-200003-I

[ITU 2004] The ITU Web site. http://www.itu.int/

[ITU Statistics 2004] International Telecommunication Union. "Key Global Telecom Indicators for the World Telecommunication Service Sector." http://www.itu.int/ITU-D/ict/statistics/at_glance/KeyTelecom99.html

[ITU-T Q.2931 1994] "Broadband Integrated Service Digital Network (B-ISDN) Digital Subscriber Signaling System no.2 (DSS2) User Network Interface Layer 3 Specification for Basic Call/Connection Control." *ITU-T Recommendation Q.2931*. Geneva: International Telecommunication Union, 1994.

[Iyer 2002] S. Iyer, R. Zhang e N. McKeown. "Routers with a Single Stage of Buffering." *Proceedings 2002 ACM Sigcomm Conference*. http://www.acm.org/sigs/sigcomm/sigcomm2002/papers/routersingle.pdf.

[Jacobson 1988] V. Jacobson. "Congestion Avoidance and Control." *Proceedings of ACM SIGCOMM '88*. Stanford, ago. 1988, p. 314-329. ftp://ftp.ee.lbl.gov/papers/congavoid.ps.Z

[Jain 1989] R. Jain. "A Delay-Based Approach for Congestion Avoidance in Interconnected Heterogeneous Computer Networks." *ACM Computer Communications Review*, vol. 19, n. 5, 1989, p. 56-71. http://www.cis.ohio-state.edu/~jain/papers/delay.htm

[Jain 1994] R. Jain. *FDDI Handbook: High-Speed Networking Using Fiber and Other Media*. Reading: Addison-Wesley, 1994.

[Jain 1996] R. Jain, S. Kalyanaraman, S. Fahmy, R. Goyal e S. Kim. "Tutorial Paper on ABR Source Behavior." *ATM Forum/96-1270*, out. 1996. http://www.cis.ohio-state.edu/~jain/atmf/a96-1270.htm

[Jaiswal 2003] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose e D. Towsley. "Measurement and Classification of Out-of-Sequence Packets in a Tier-1 IP backbone." *Proceedings of 2003 INFOCOM*. ftp://gaia.cs.umass.edu/pub/Jaiswal03_oos.pdf.

[Jakobson 1993] G. Jacobson e M. Weissman. "Alarm Correlation." *IEEE Network Magazine*, 1993, p. 52-59.

[Ji 2003] P. Ji, Z. Ge, J. Kurose e D. Towsley. "A Comparison of Hard-State and Soft-State Signaling Protocols." *Proceedings of 2003 ACM SIGCOMM*. http://www.acm.org/sigs/sigcomm/sigcomm2003/papers/p251-ji.pdf

[Jiang 2001] W. Jiang, J. Lennox, H. Schulzrinne e K. Singh. "Towards Junking the PBX: Deploying IP Telephony." *NOSSDAV'01*. Port Jefferson, jun. 2001. http://www.cs.columbia.edu/~hgs/papers/Jian0106_Junking.pdf

[Jimenez 1997] D. Jimenez. "Outside Hackers Infiltrate MIT Network, Compromise Security." *The Tech*, vol. 117, n. 49, out. 1997, p. 1. http://www-tech.mit.edu/V117/N49/hackers.49n.html

[Jin 2004] C. Jin, D.X. We e S. Low. "FAST TCP: Motivation, Architecture, Algorithms, Performance." *Proceeding of IEEE Infocom*. Hong Kong, mar. 2004. http://netlab.caltech.edu/pub/papers/FAST-csreport2003.pdf.

[Kaaranen 2001] H. Kaaranen, S. Naghian, L. Laitinen, A. Ahtiainen e V. Niemi. *UMTS Networks, Architecture, Mobility and Services*. Nova York: John Wiley & Sons, 2001.

[Kahn 1967] D. Kahn. *The Codebreakers, the Story of Secret Writing*. The Macmillan Company, 1967.

[Kahn 1978] R.E. Kahn, S. Gronemeyer, J. Burchfiel e R. Kunzelman. "Advances in Packet Radio Technology." *Proceedings of the IEEE*, 66, 11, nov. 1978.

[Kangasharju 2000] J. Kangasharju, K.W. Ross e J.W. Roberts. "Performance Evaluation of Redirection Schemes in Content Distribution Networks." *Proceedings of 5th Web Caching and Content Distribution Workshop*. Lisboa, maio 2000. http://www.terena.nl/conf/wcw/Proceedings/S4/S4-2.ps

[Kapoor 1997] H. Kapoor. "CoreBuilder 5000 Switch Module Architecture." *3 Corporation*, white paper, n. 500645.

[Kar 2000] K. Kar, M. Kodialam e T.V. Lakshman. "Minimum Interference Routing of Bandwidth Guaranteed Tunnels with MPLS Traffic Engineering Applications." *IEEE J. Selected Areas in Communications*, dez. 2000. http://www.bell-labs.com/org/11347A/paper/minint_jsac.pdf

[Karol 1987] M. Karol, M. Hluchyj e A. Morgan. "Input Versus Output Queuing on a Space-Division Packet Switch." *IEEE Transactions on Communications*, vol. COM-35, n. 12, dez. 1987, p. 1347-1356.

[Katzela 1995] I. Katzela e M. Schwartz. "Schemes for Fault Identification in Communication Networks." *IEEE/ACM Transactions on Networking*, vol. 3, n. 6, dez. 1995, p. 753-764.

[Kaufman 1995] C. Kaufman, R. Perlman e M. Speciner. *Network Security, Private Communication in a Public World*. Englewood Cliffs: Prentice Hall, 1995.

[KaZaA 2004] KaZaA homepage. http://www.kazaa.com

[Kelly 2003] T. Kelly. *Scalable TCP: Improving Performance in Highspeed Wide Area Networks*. http://www-lce.eng.cam.ac.uk/~ctk21/papers/scalable_improve_hswan.pdf.

[Kende 2000] M. Kende. "The Digital Handshake: Connecting Internet Backbones." *FCC Report*, 2000. http://www.fcc.gov/Bureaus/OPP/working_papers/oppwp32.pdf

[Keshav 1998] S. Keshav e R. Sharma. "Issues and Trends in Router Design." *IEEE Communications Magazine*, vol. 36, n. 5, maio 1998, p. 144-151.

[Kilkki 1999] K. Kilkki. *Differentiated Services for the Internet*. Indianápolis: Macmillan Technical Publishing, 1999.

[Kleinrock 1961] L. Kleinrock. "Information Flow in Large Communication Networks." *RLE Quarterly Progress Report*, jul. 1961.

[Kleinrock 1964] L. Kleinrock. *1964 Communication Nets: Stochastic Message Flow and Delay*. Nova York: McGraw-Hill, 1964.

[Kleinrock 1975] L. Kleinrock. *Queuing Systems, vol. 1*. Nova York: John Wiley, 1975.

[Kleinrock 1975b] L. Kleinrock e F.A. Tobagi. "Packet Switching in Radio Channels: Part I – Carrier Sense Multiple-Access Modes and Their Throughput-Delay Characteristics." *IEEE Transactions on Communications*, vol. COM-23, n. 12, dez. 1975, p. 1400-1416.

[Kleinrock 1976] L. Kleinrock. *Queuing Systems, vol. 2*. Nova York: John Wiley, 1976.

[Kleinrock 2004] L. Kleinrock. "The Birth of the Internet." http://www.lk.cs.ucla.edu/LK/Inet/birth.html

[Kohler 2004] E. Kohler, M. Handley, S. Floyd e J. Padhye. DCCP homepage. http://www.icir.org/kohler/dccp/

[Korhonen 2003] J. Korhonen. *Introduction to 3G Mobile Communications*, 2 ed. Artech House, 2003.

[Krishnamurthy 2001] B. Krishnamurthy e J. Rexford. *Web Protocols and Practice: HTTP/1.1, Networking Protocols, and Traffic Measurement*. Boston: Addison-Wesley, 2001.

[Kurose 1996] J.F. Kurose. "Unix Network Programming." http://manic.cs.umass.edu/~amldemo/courseware/intro.html

[Labovitz 1997] C. Labovitz, G.R. Malan e F. Jahanian. "Internet Routing Instability." *Proceedings of ACM SIGCOMM '97*. Cannes, 1997, p. 115-126. http://www.acm.org/sigcomm/sigcomm97/papers/p109.html

[Labrador 1999] M. Labrador e S. Banerjee. "Packet Dropping Policies for ATM and IP Networks." *IEEE Communications Surveys*, vol. 2, n. 3, Third Quarter 1999, p. 2-14. http://www.comsoc.org/livepubs/surveys/public/3q99issue/banerjee.html

[Lakshman 1997] T.V. Lakshman e U. Madhow. "The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss." *IEEE/ACM Transactions on Networking*, vol. 5, n. 3, 1997, p. 336-350. http://citeseer.ist.psu.edu/lakshman96performance.html

[Lam 1980] S. Lam. "A Carrier Sense Multiple Access Protocol for Local Networks." *Computer Networks*, vol. 4, 1980, p. 21-32, 1980.

[Lamport 1981] L. Lamport. "Password Authentication with Insecure Communication." *Communications of the ACM*, vol. 24, n. 11, nov. 1981, p. 770-772.

[Larmouth 1996] J. Larmouth. *Understanding OSI*. International Thomson Computer Press, 1996. O capítulo 8 fala sobre a ASN.1 e está disponível on-line em http://www.salford.ac.uk/iti/books/osi/all.html#head8

[Larsen 1997] A. Larsen. "Guaranteed Service: Monitoring Tools." *Data Communications*, jun. 1997, p. 85-94.

[Lawton 2001] G. Lawton. "Is IPv6 Finally Gaining Ground?" *IEEE Computer Magazine*, ago. 2001, p. 11-15.

[Leiner 1998] B. Leiner, V. Cerf, D. Clark, R. Kahn, L. Kleinrock, D. Lynch, J. Postel, L. Roberts e S. Woolf. "A Brief History of the Internet." http://www.isoc.org/internet/history/brief.html

[Liang 2004] J. Liang, R. Kumar e K.W. Ross. "Understanding KaZaA." http://cis.poly.edu/~ross/papers/

[Lin 2001] Y. Lin e I. Chlamtac. *Wireless and Mobile Network Architectures*. Nova York: John Wiley & Sons, 2001.

[Liu 2002] B. Liu, D. Goeckel e D. Towsley. "TCP-Cognizant Adaptive Forward Error Correction in Wireless Networks." *Proceedings of Globe Internet 2002*. ftp://gaia.cs.umass.edu/pub/wirelessTCPtech.pdf

[Luotonen 1998] A. Luotonen. *Web Proxy Servers*. Englewood Cliffs: Prentice Hall, 1998.

[Lynch 1993] D. Lynch e M. Rose. *Internet System Handbook*. Reading: Addison Wesley, 1993.

[Macedonia 1994] M.R. Macedonia e D.P. Brutzman. "MBone Provides Audio and Video Across the Internet." *IEEE Computer Magazine*, vol. 27, n. 4, abr. 1994, p. 30-36. ftp://taurus.cs.nps.navy.mil/pub/mbmg/mbone.html

[Maconachy 2001] W.V. Maconachy, C. Schou, D. Ragsdale e D. Welch. "A Model for Information Assurance: an Integrated Approach." *Proceedings of the 2001 IEEE Workshop on Information Assurance and Security*. West Point, 2001. http://www.itoc.usma.edu/Documents/Workshop2001/paperW2C3(55).pdf

[Maennel 2002] O. Maennel e A. Feldmann. "Realistic BGP Traffic for Test Labs." *Proceedings of 2002 ACM Sigcomm*. http://www.acm.org/sigs/sigcomm/sigcomm2002/papers/bgplab.pdf.

[Mahdavi 1997] J. Mahdavi e S. Floyd. "TCP-Friendly Unicast Rate-Based Flow Control", nota não publicada, jan. 1997. http://www.psc.edu/networking/papers/tcp_friendly.html

[Mainwaring 2002] A. Mainwaring, R. Szewczyk, D. Culler e J. Anderson "Wireless Sensor Networks for Habitat Monitoring." *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, 2002. http://citeseer.ist.psu.edu/mainwaring02wireless.html

[Manelli 2001] T. Manelli. "What Happened to Internet Appliances?" *PC World*, abr. 2001. http://www.pcworld.com/news/article/0,aid,47184,00.asp

[manet 2004] IETF Mobile Ad-Hoc Networks (manet) Working Group. http://www.ietf.org/html.charters/manet-charter.html

[Maymounkov 2002] P. Maymounkov e D. Mazières. "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric." *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, mar. 2002, p. 53-65.

[McAuley 1994] A. McAuley. "Weighted Sum Codes for Error Detection and Their Comparison with Existing Codes." *IEEE/ACM Transactions on Networking*, vol. 2, n. 1, fev. 1994, p. 16-22.

[McCumber 1991] J. McCumber. "Information Systems Security: A Comprehensive Model." *Proceedings of the 14th National Computer Security Conference*. Baltimore, 1991.

[MCI 2004] MCI. "Terms and Conditions: Service Level Agreement." http://global.mci.com/terms/sla/

[McKeown 1997a] N. McKeown, M. Izzard, A. Mekkittikul,W. Ellersick e M. Horowitz. "The Tiny Tera: A Packet Switch Core." *IEEE Micro Magazine*, jan./fev. 1997. http://tiny-tera.stanford.edu/~nickm/papers/HOTI_96.ps

[McKeown 1997b] N. McKeown. "A Fast Switched Backplane for a Gigabit Switched Router." *Business Communications Review*, vol. 27, n. 12. http://www.bcr.com/bcrmag/12/mckeown.htm

[McKusik 1996] M.K. McKusick, K. Bostic, M. Karels e J. Quarterman. *The Design and Implementation of the 4.4BSD Operating System*. Reading: Addison Wesley, 1996.

[McQuillan 1980] J. McQuillan, I. Richer e E. Rosen. "The New Routing Algorithm for the Arpanet." *IEEE Transactions on Communications*, COM-28(5), maio 1980, p. 711-719.

[Medhi 1997] D. Medhi e D. Tipper (eds.). "Special Issue: Fault Management in Communication Networks." *Journal of Network and Systems Management*, vol. 5, n. 2, jun. 1997.

[Metcalfe 1976] R.M. Metcalfe e D.R. Boggs. "Ethernet: Distributed Packet Switching for Local Computer Networks." *Communications of the Association for Computing Machinery*, vol. 19, n. 7, jul. 1976, p. 395-404. http://www.acm.org/classics/apr96/

[Microsoft Player Media 2004] Microsoft Windows Media homepage. http://www.microsoft.com/windows/windowsmedia/

[Miller 1997] M.A. Miller. *Managing Internetworks with SNMP*, 2 ed. Nova York: M & T Books, 1997.

[Mockapetris 1988] P.V. Mockapetris e K.J. Dunlap. "Development of the Domain Name System." *Proceedings of SIGCOMM '88*. Stanford, 1988. http://citeseer.nj.nec.com/mockapetris88development.html

[Molinero-Fernandez 2002] P. Molinero-Fernandez, N. McKeown e H. Zhang. "Is IP Going to Take Over the World (of Communications)?" *Proceedings of the 2002 ACM Hotnets*. http://www.acm.org/sigcomm/HotNets-I/papers/fernandez.pdf

[Molle 1987] M.L. Molle, K. Sohraby e A.N. Venetsanopoulos. "Space-Time Models of Asynchronous CSMA Protocols for Local Area Networks." *IEEE Journal on Selected Areas in Communications*, vol. 5, n. 6, 1987, p. 956-968.

[Molva 1999] R. Molva. "Internet Security Architecture." *Computer Networks and ISDN Systems*, vol. 31, n. 8, 1999, p. 787-804.

[Moore 2003] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford e N. Weaver. "Inside the Slammer Worm." http://www.caida.org/outreach/papers/2003/sapphire2/

[Mouly 1992] M. Mouly e M. Pautet. *The GSM System for Mobile Communications*. Palaiseau: Cell and Sys, 1992.

[Moy 1998] J. Moy. *OSPF: Anatomy of An Internet Routing Protocol*. Reading: Addison Wesley, 1998.

[mrouted 1996] "mrouted", versão 3.8 do software de roteamento DVMRP para várias plataformas de roteamento de estação de trabalho. ftp://parcftp.xerox.com/pub/net-research/ipmulti

[Mukherjee 1997] B. Mukherjee. *Optical Communication Networks*. McGraw-Hill, 1997.

[Murphy 2003] S. Murphy. "BGP Security Vulnerabilities Analysis." draft-ietf-idr-bgp-vuln-00.txt, jun. 2003. ftp://ftp.rfc-editor.org/in-notes/internet-drafts/draft-ietf-idr-bgp-vuln-00.txt

[Nahum 2002] E. Nahum, T. Barzilai e D. Kandlur. "Performance Issues in WWW Servers." *IEEE/ACM Transactions on Networking*, vol. 10, n. 1, fev. 2002. http://www.research.ibm.com/people/n/nahum/publications/ton02-www-camera.pdf.

[Nesbitt 2002] S. Nesbitt. "Network Appliances." *About.com*, jan. 2002. http://netappliances.about.com/cs/settopboxes/

[Net2Phone 2004] http://www.net2phone.com/

[Netcraft 2004] The Netcraft Web Server Survey, Netcraft Web Site. http://www.netcraft.com/survey/

[Netscape Cookie 1999] Netscape Communications Corp. "Persistent Client State http Cookies." http://home.netscape.com/newsref/std/cookie_spec.html

[Netscape SSL 1998] Netscape Communications Corps. "Introduction to SSL." http://developer.netscape.com/docs/manuals/security/sslin/

[Neuman 1994] B. Neuman e T. Tso. "Kerberos: An Authentication Service for Computer Networks." *IEEE Communication Magazine*, vol. 32, n. 9, set. 1994, p. 33-38.

[Neumann 1997] R. Neumann. "Internet Routing Black Hole." *The Risks Digest: Forum on Risks to the Public in Computers and Related Systems*, vol. 19, n. 12, maio 1997. http://catless.ncl.ac.uk/Risks/19.12.html#subj1.1

[Nielsen 1997] H.F. Nielsen, J. Gettys, A. Baird-Smith, E. Prud'hommeaux, H.W. Lie e C. Lilley. "Network Performance Effects of HTTP/1.1, CSS1, and PNG." *W3C Document*, 1997 (pode ser encontrado também em *Proceedings of ACM SIGCOMM '97*. Cannes, p. 155-166). http://www.acm.org/sigcomm/sigcomm97/papers/p102.html

[NIST 1993] National Institute of Standards and Technology. "Federal Information. Data Encryption Standard." *Processing Standards Publication* 46-2, 1993. http://www.itl.nist.gov/fipspubs/fip46-2.htm

[NIST 1999] National Institute of Standards and Technology. "Data Encryption Standard Fact Sheet." http://csrc.nist.gov/cryptval/des/des.txt

[NIST 1999b] National Institute of Standards and Technology. "Draft Federal Information Processing Standard (FIPS) 46-3, Data Encryption Standard (DES), and Request for Comments." http://csrc.nist.gov/cryptval/des/fr990115.htm

[NIST 2001] National Institute of Standards and Technology. "Advanced Encryption Standard (AES)." *Federal Information Processing Standards* 197, nov. 2001. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[Nmap 2004] Nmap homepage. http://www.insecure.com/nmap

[Nonnenmacher 1998] J. Nonnenmacher, E. Biersak e D. Towsley. "Parity-Based Loss Recovery for Reliable Multicast Transmission." *IEEE/ACM Transactions on Networking*, vol. 6, n. 4, ago. 1998, p. 349-361. ftp://gaia.cs.umass.edu/pub/NBT97:fec.ps.gz

[Nortel 2004] Nortel Networks, Optivity Portfolio. http://www.nortelnetworks.com/products/01/optivity

[NTIA 1998] National Telecommunications and Information Administration (NTIA), U.S. Department of Commerce. "Management of Internet Names and Addresses." Docket Number: 980212036-8146-02. http://www.ntia.doc.gov/ntiahome/domainname/6_5_98dns.htm

[Odlyzko 2003] A. Odlyzko. "Internet Traffic Growth: Sources and Implications." *A. M. Optical Transmission Systems and Equipment for WDM Networking II, Proc. SPIE*, 5247, 2003, p. 1-15. http://www.dtc.umn.edu/~odlyzko/doc/itcom.internet.growth.pdf.

[OpenView2004] HP OpenView homepage. http://www.openview.hp.com/

[OSS 2004] OSS Nokalva. "ASN.1 Resources." http://www.oss.com/asn1/

[Overpeer 2004] Overpeer Inc. http://www.overpeer.com

[Padhye 2000] J. Padhye, V. Firoiu, D. Towsley e J. Kurose. "Modeling TCP Reno Performance: A Simple Model and its Empirical Validation." *IEEE/ACM Transactions on Networking*, vol. 8, n. 2, abr. 2000, p. 133-145.

[Padhye 2001] J. Padhye e S. Floyd. "On Inferring TCP Behavior." *Proceedings of ACM SIGCOMM*. San Diego, 2001v. http://www.aciri.org/floyd/papers/tbit.pdf

[Pan 1997] P. Pan e H. Schulzrinne. "Staged Refresh Timers for RSVP." *2nd Global Internet Conference*. Phoenix, 1997. http://www.cs.columbia.edu/~pingpan/papers/timergi.pdf

[Parekh 1993] A. Parekh e R. Gallager. "A generalized Processor Sharing Approach to Flow Control in Iintegrated Services Networks: the Single-Node Case." *IEEE/ACM Transactions on Networking*, vol. 1, n. 3, jun. 1993, p. 344-357.

[Partridge 1992] C. Partridge e S. Pink. "An Implementation of the Revised Internet Stream Protocol (ST-2)." *Journal of Internetworking: Research and Experience*, vol. 3, n. 1, mar. 1992. http://www.sics.se/cna/publications/ST-2.ps

[Partridge 1998] C. Partridge *et al.* "A Fifty Gigabit per second IP Router." *IEEE/ACM Transactions on Networking*, vol. 6, n. 3, jun. 1998, p. 237-248.

[Paxson 1997] V. Paxson. "End-to-End Internet Packet Dynamics." *Proceedings of ACM SIGCOMM '97*. Cannes, set. 1997. http://www.acm.org/sigcomm/sigcomm97/papers/p086.html

[Perkins 1994] A. Perkins. "Networking with Bob Metcalfe." *The Red Herring Magazine*, nov. 1994. http://www.herring.com/mag/issue15/bob.html

[Perkins 1998a] C. Perkins, O. Hodson e V. Hardman. "A Survey of Packet Loss Recovery Techniques for Streaming Audio." *IEEE Network Magazine*, set./out. 1998, p. 40-47.

[Perkins 1998b] C. Perkins. *Mobile IP: Design Principles and Practice*. Reading: Addison Wesley, 1998.

[Perkins 2000] C. Perkins. *Ad-Hoc Networking*. Reading: Addison-Wesley, 2000.

[Perlman 1999] R. Perlman. *Interconnections: Bridges, Routers, Switches, and Internetworking Protocols*, 2 ed. Reading: Addison Wesley Professional Computing Series, 1999.

[PGPI 2004] The International PGP Home Page. http://www.pgpi.org

[Phifer 2000] L. Phifer. "The Trouble with NAT." *The Internet Protocol Journal*, vol. 3, n. 4, dez. 2000. http://www.cisco.com/warp/public/759/ipj_3-4/ipj_3-4_nat.html

[Pickholtz 1982] R. Pickholtz, D. Schilling e L. Milstein. "Theory of Spread Spectrum Communication – a Tutorial." *IEEE Transactions on Communications*, vol. COM-30, n. 5, maio 1982, p. 855-884.

[Piscatello 1993] D. Piscatello e A. Lyman Chapin. "Open Systems Networking." Reading: Addison Wesley, 1993.

[Point Topic 2003] Point Topic Ltd. *World Broadband Statistics*, set. 2003. http://www.point-topic.com

[QuickTime 2004] QuickTime homepage. http://www.apple.com/quicktime

[Quittner 1998] J. Quittner, M. Slatalla. *Speeding the Net: The Inside Story of Netscape and How it Challenged Microsoft*. Atlantic Monthly Press, 1998.

[Ramakrishnan 1990] K.K. Ramakrishnan e Raj Jain. "A Binary Feedback Scheme for Congestion Avoidance in Computer Networks." *ACM Transactions on Computer Systems*, vol. 8, n. 2, maio 1990, p. 158-181.

[Raman 1999] S. Raman e S. McCanne. "A Model, Analysis, and Protocol Framework for Soft State-based Communication." *Proceedings of ACM SIGCOMM '99*. Boston, ago. 1999. http://www.acm.org/sigs/sigcomm/sigcomm99/papers/session1-2.html

[Ramaswami 1998] R. Ramaswami e K. Sivarajan. *Optical Networks: A Practical Perspective*. Morgan Kaufman Publishers, 1998.

[Ramjee 1994] R. Ramjee, J. Kurose, D. Towsley e H. Schulzrinne. "Adaptive Playout Mechanisms for Packetized Audio Applications in Wide-Area Networks." *Proceeding IEEE Infocom 94*. ftp://gaia.cs.umass.edu/pib/ Ramj94:Adaptive.ps.Z

[Ratnasamy 2001] S. Ratnasamy, P. Francis, M. Handley, R. Karp e S. Shenker. "A Scalable Content-Addressable Network." *Proceedings of ACM SIGCOMM*. San Diego, 2001. http://www.acm.org/sigcomm/sigcomm2001/p13.html

[Rao 1996] K.R. Rao e J.J. Hwang. *Techniques and Standards for Image, Video and Audio Coding*. Englewood Cliffs: Prentice Hall, 1996.

[RAT 2004] Robust Audio Tool. http://www-mice.cs.ucl.ac.uk/multimedia/software/rat/

[RealNetworks 2004] RealNetworks homepage. http://www.realnetworks.com

[Reid 2003] N. Reid e R. Seide. *802.11 (Wi-Fi) Networking Handbook*. Nova York: McGraw-Hill/Osborne, 2003.

[RFC 001] S. Crocker. "Host Software", RFC 001 (o primeiríssimo RFC!). http://www.rfc-editor.org/rfc/rfc1.txt

[RFC 741] D. Cohen. "Specifications for the Network Voice Protocol NVP", RFC 741, nov. 1977. ftp://ftp.rfc-editor.org/in-notes/rfc741.txt

[RFC 768] J. Postel. "User Datagram Protocol", RFC 768, ago. 1980. http://www.rfc-editor.org/rfc/rfc768.txt

[RFC 789] E. Rosen. "Vulnerabilities of Network Control Protocols", RFC 789. http://www.rfc-editor.org/rfc/rfc789.txt

[RFC 791] J. Postel. "Internet Protocol: DARPA Internet Program Protocol Specification", RFC 791, set. 1981. http://www.rfc-editor.org/rfc/rfc791.txt

[RFC 792] J. Postel. "Internet Control Message Protocol", RFC 792, set. 1981. http://www.rfc-editor.org/rfc/rfc792.txt

[RFC 793] J. Postel. "Transmission Control Protocol", RFC 793, set. 1981. http://www.rfc-editor.org/rfc/rfc793.txt

[RFC 801] J. Postel. "NCP/TCP Transition Plan", RFC 801, nov. 1981. http://www.rfc-editor.org/rfc/rfc801.txt

[RFC 821] J. Postel. "Simple Mail Transfer Protocol", RFC 821, ago. 1982. http://www.rfc-editor.org/rfc/rfc211.txt substituído pelo RFC 2821.

[RFC 822] D.H. Crocker. "Standard for the Format of ARPA Internet Text Messages", RFC 822, ago. 1982. http://www.rfc-editor.org/rfc/rfc822.txt

[RFC 826] D.C. Plummer. "An Ethernet Address Resolution Protocol – or – Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware", RFC 826, nov. 1982. http://www.rfc-editor.org/rfc/rfc826.txt

[RFC 829] V. Cerf. "Packet Satellite Technology Reference Sources", RFC 829, nov. 1982. http://www.rfc-editor.org/rfc/rfc829.txt

[RFC 854] J. Postel e J. Reynolds. "TELNET Protocol Specification", RFC 854, maio 1993. http://www.rfc-editor.org/rfc/rfc854.txt

[RFC 904] D. Mills. "Exterior Gateway Protocol Formal Specification", RFC 904, abr. 1984. http://www.rfc-editor.org/rfc/rfc904.txt

[RFC 950] J. Mogul e J. Postel. "Internet Standard Subnetting Procedure", RFC 950, ago. 1985. http://www.rfc-editor.org/rfc/rfc904.txt.

[RFC 959] J. Postel e J. Reynolds. "File Transfer Protocol (FTP)", RFC 959, out 1985. http://www.rfc-editor.org/rfc/rfc959.txt

[RFC 977] B. Kantor e P. Lapsley. "Network News Transfer Protocol", RFC 977, fev. 1986. http://www.rfc-editor.org/rfc/rfc977.txt

[RFC 1028] J. Davin, J.D. Case, M. Fedor e M. Schoffstall. "A Simple Gateway Monitoring Protocol", RFC 1028, nov. 1987. http://www.rfc-editor.org/rfc/rfc1028.txt.

[RFC 1034] P.V. Mockapetris. "Domain Names – Concepts and Facilities", RFC 1034, nov. 1987. http://www.rfc-editor.org/rfc/rfc1034.txt

[RFC 1035] P. Mockapetris. "Domain Names – Implementation and Specification", RFC 1035, nov. 1987. http://www.rfc-editor.org/rfc/rfc1035.txt

[RFC 1058] C.L. Hendrick. "Routing Information Protocol", RFC 1058, jun. 1988. http://www.rfc-editor.org/rfc/rfc1058.txt

[RFC 1071] R. Braden, D. Borman e C. Partridge. "Computing The Internet Checksum", RFC 1071, set. 1988. http://www.rfc-editor.org/rfc/rfc1071.txt

[RFC 1075] D. Waitzman, C. Partridge e S. Deering. "Distance Vector Multicast Routing Protocol", RFC 1075, nov. 1988. http://www.rfc-editor.org/rfc/rfc1071.txt.

[RFC 1112] S. Deering. "Host Extension for IP Multicasting", RFC 1112, ago. 1989. http://www.rfc-editor.org/rfc/rfc1112.txt

[RFC 1122] R. Braden. "Requirements for Internet Hosts – Communication Layers", RFC 1122, out. 1989. http://www.rfc-editor.org/rfc/rfc1122.txt

[RFC 1123] R. Braden (ed.) "Requirements for Internet Hosts – Application and Support", RFC-1123, out. 1989. ftp://ftp.rfc-editor.org/in-notes/rfc1123.txt.

[RFC 1142] D. Oran. "OSI IS-IS Intra-Domain Routing Protocol", RFC 1142, fev. 1990. ftp://ftp.rfc-editor.org/in-notes/rfc1142.txt

[RFC 1180] T. Socolofsky e C. Kale. "A TCP/IP Tutorial", RFC 1180, jan. 1991. http://www.rfc-editor.org/rfc/rfc1180.txt

[RFC 1190] C. Topolcic. "Experimental Internet Stream Protocol: Version 2 (ST-II)", RFC 1190, out. 1990. ftp://ftp.rfc-editor.org/in-notes/rfc1190.txt

[RFC 1191] J. Mogul e S. Deering. "Path MTU Discovery", RFC 1191, nov. 1990. ftp://ftp.rfc-editor.org/in-notes/rfc1191.txt

[RFC 1213] K. McCloghrie e M.T. Rose. "Management Information Base for Network Management of TCP/IP-based internets: MIB-II", RFC 1213, mar. 1991. http://www.rfc-editor.org/rfc/rfc1213.txt

[RFC 1256] S. Deering. "ICMP Router Discovery Messages", RFC 1256, set. 1991. http://www.rfc-editor.org/rfc/rfc1256.txt

[RFC 1320] R. Rivest. "The MD4 Message-Digest Algorithm." RFC 1320, abr. 1992. http://www.rfc-editor.org/rfc/rfc1320.txt

[RFC 1321] R. Rivest. "The MD5 Message-Digest Algorithm." RFC 1321, abr. 1992. http://www.rfc-editor.org/rfc/rfc1321.txt

[RFC 1323] V. Jacobson, S. Braden e D. Borman. "TCP Extensions for High Performance", RFC 1323, maio 1992. http://www.rfc-editor.org/rfc/rfc1323.txt

[RFC 1332] G. McGregor. "The PPP Internet Protocol Control Protocol (IPCP)", RFC 1332, maio 1992. http://www.rfc-editor.org/rfc/rfc1332.txt

[RFC 1378] B. Parker. "The PPP AppleTalk Control Protocol (ATCP)", RFC 1378, nov. 1992. http://www.rfc-editor.org/rfc/rfc1378.txt

[RFC 1422] S. Kent. "Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management", RFC 1422, fev. 1993. http://www.rfc-editor.org/rfc/rfc1422.txt

[RFC 1510] J. Kohl e C. Neuman. "The Kerberos Network Authentication Service (V5)", RFC 1510, set. 1993. http://www.rfc-editor.org/rfc/rfc1510.txt

[RFC 1519] V. Fuller, T. Li, J. Yu e K. Varadhan. "Classless Inter-Domain Routing (CIDR)", RFC 1519, set. 1993. http://www.rfc-editor.org/rfc/rfc1519.txt

[RFC 1542] W. Wimer. "Clarifications and Extensions for the Bootstrap Protocol", RFC 1542, out. 1993. http://www.rfc-editor.org/rfc/rfc1542.txt

[RFC 1547] D. Perkins. "Requirements for an Internet Standard Point-to-Point Protocol", RFC 1547, dez. 1993. http://www.rfc-editor.org/rfc/rfc1547.txt

[RFC 1584] J. Moy. "Multicast Extensions to OSPF", RFC 1584, mar. 1994. http://www.rfc-editor. org/rfc/rfc1584.txt

[RFC 1631] K. Egevang e P. Francis. "The IP Network Address Translator (NAT)", RFC 1631, maio 1994. http://www.rfc-editor.org/rfc/rfc1631.txt

[RFC 1633] R. Braden, D. Clark e S. Shenker. "Integrated Services in the Internet Architecture: an Overview", RFC 1633, jun. 1994. http://www.rfc-editor.org/rfc/rfc1633.txt

[RFC 1636] R. Braden, D. Clark, S. Crocker e C. Huitema. "Report of IAB Workshop on Security in the Internet Architecture", RFC 1636, nov. 1994. http://www.rfc-editor.org/rfc/rfc1636.txt

[RFC 1661] W. Simpson (ed.). "The Point-to-Point Protocol (PPP)", RFC 1661, jul. 1994. http://www.rfc-editor.org/rfc/rfc1661.txt

[RFC 1662] W. Simpson (ed.) "PPP in HDLC-Like Framing", RFC 1662, jul. 1994. http://www.rfc-editor.org/rfc/rfc1662.txt

[RFC 1700] J. Reynolds e J. Postel. "Assigned Numbers", RFC 1700, out. 1994. http://www.rfc-editor.org/rfc/rfc1700.txt

[RFC 1730] M. Crispin. "Internet Message Access Protocol – Version 4", RFC 1730, dez. 1994. http://info.internet.isi.edu/in-notes/rfc/files/rfc1730.txt. Substituído pelo RFC 2060.

[RFC 1752] S. Bradner e A. Mankin. "The Recommendations for the IP Next Generation Protocol", RFC 1752, jan. 1995. http://www.rfc-editor.org/rfc/rfc1752.txt

[RFC 1760] N. Haller. "The S/KEY One-Time Password System", RFC 1760, fev. 1995. http://www.rfc-editor.org/rfc/rfc1760.txt

[RFC 1762] S. Senum. "The PPP DECnet Phase IV Control Protocol (DNCP)", RFC 1762, mar. 1995. http://www.rfc-editor.org/rfc/rfc1762.txt

[RFC 1771] Y. Rekhter e T. Li. "A Border Gateway Protocol 4 (BGP-4)", RFC 1771, mar. 1995. http://www.rfc-editor.org/rfc/rfc1771.txt

[RFC 1772] Y. Rekhter e P. Gross. "Application of the Border Gateway Protocol in the Internet", RFC 1772, mar. 1995. http://www.rfc-editor.org/rfc/rfc1772.txt

[RFC 1773] P. Traina. "Experience with the BGP-4 Protocol", RFC 1773, mar. 1995. http://www.rfc-editor.org/rfc/rfc1773.txt.

[RFC 1779] S. Kille. "A String Representation of Distinguished Names", RFC 1779, mar. 1995. http://www.rfc-editor.org/rfc/rfc1779.txt. Substituído pelo RFC 2253

[RFC 1810] J. Touch. "Report on MD5 Performance", RFC 1810, jun. 1995. http://www.rfc-editor.org/rfc/rfc1810.txt

[RFC 1812] F. Baker (ed.) "Requirements for IP Version 4 Routers", RFC-1812, jun. 1995. ftp://ftp.rfc-editor.org/in-notes/rfc1812.txt.

[RFC 1884] R. Hinden e S. Deering. "IP Version 6: Addressing Architecture." RFC 1884, dez. 1995. http://www.rfc-editor.org/rfc/rfc1884.txt. Substituído pelo RFC 2373.

[RFC 1906] J. Case, K. McCloghrie, M. Rose e S. Waldbusser. "Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1906, jan. 1996. http://www.rfc-editor.org/rfc/rfc1906.txt

[RFC 1907] J. Case, K. McCloghrie, M. Rose e S. Waldbusser. "Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1907, jan. 1996. http://www.rfc-editor.org/rfc/rfc1907.txt

[RFC 1911] G. Vaudreuil. "Voice Profile for Internet Mail", RFC 1911, fev. 1996. http://www.rfc-editor.org/rfc/rfc1911.txt. Substituído pelo RFC 2421.

[RFC 1918] Y. Rekhter, B. Moskowitz, D. Karrenberg, G.J. de Groot e E. Lear. "Address Allocation for Private Internets", RFC 1918, fev. 1996. ftp://ftp.rfc-editor.org/in-notes/rfc1918.txt

[RFC 1930] J. Hawkinson e T. Bates. "Guidelines for Creation, Selection, and Registration of an Autonomous System (AS)", RFC 1930, mar. 1996. ftp://ftp.rfc-editor.org/in-notes/rfc1930.txt

[RFC 1938] N. Haller e C. Metz. "A One-Time Password System", RFC 1938, maio 1996. ftp://ftp.rfc-editor.org/in-notes/rfc1938.txt

[RFC 1939] J. Myers e M. Rose. "Post Office Protocol – Version 3", RFC 1939, maio 1996. http://www.rfc-editor.org/rfc/rfc1939.txt

[RFC 1945] T. Berners-Lee, R. Fielding e H. Frystyk. "Hypertext Transfer Protocol – HTTP/1.0", RFC 1945, maio 1996. http://www.rfc-editor.org/rfc/rfc1945.txt

[RFC 1994] W. Simpson. "PPP Challenge Handshake Authentication Protocol (CHAP)", RFC 1994, ago. 1996. ftp://ftp.rfc-editor.org/in-notes/rfc1994.txt

[RFC 2001] W. Stevens. "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms", RFC 2001, jan. 1997. http://www.rfc-editor.org/rfc/rfc2001.txt. Substituído pelo RFC 2581.

[RFC 2003] C. Perkins. "IP Encapsulation within IP", RFC 2003, out. 1996. http://www.rfc-editor.org/rfc/rfc2003.txt

[RFC 2004] C. Perkins. "Minimal Encapsulation within IP", RFC 2004, out. 1996. http://www.rfc-editor.org/rfc/rfc2004.txt

[RFC 2011] K. McCloghrie. "SNMPv2 Management Information Base for the Internet Protocol using SMIv2", RFC 2011, nov. 1996. http://www.rfc-editor.org/rfc/rfc2011.txt

[RFC 2012] K. McCloghrie. "SNMPv2 Management Information Base for the Transmission Control Protocol using SMIv2", RFC 2012, nov. 1996. http://www.rfc-editor.org/rfc/rfc2012.txt

[RFC 2013] K. McCloghrie. "SNMPv2 Management Information Base for the User Datagram Protocol using SMIv2", RFC 2013, nov. 1996. http://www.rfc-editor.org/rfc/rfc2013.txt

[RFC 2018] M. Mathis, J. Mahdavi, S. Floyd e A. Romanow. "TCP Selective Acknowledgment Options", RFC 2018, out. 1996. http://www.rfc-editor.org/rfc/rfc2018.txt

[RFC 2021] S. Waldbusser. "Remote Network Monitoring Management Information Base Version 2 using SMIv2", RFC 2021, jan. 1997. http://www.rfc-editor.org/rfc/rfc2021.txt

[RFC 2045] N. Freed e N. Borenstein. "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, nov. 1996. http://www.rfc-editor.org/rfc/rfc2045.txt

[RFC 2046] N. Freed e N. Borenstein. "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, nov. 1996. http://www.rfc-editor.org/rfc/rfc2046.txt

[RFC 2048] N. Freed, J. Klensin e J. Postel "Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures", RFC 2048, nov. 1996. http://www.rfc-editor.org/rfc/rfc2048.txt

[RFC 2050] K. Hubbard, M. Kosters, D. Conrad, D. Karrenberg e J. Postel. "Internet Registry IP Allocation Guidelines", RFC 2050, nov. 1996. http://www.rfc-editor.org/rfc/rfc2050.txt

[RFC 2060] R. Crispin. "Internet Message Access Protocol – Version 4rev1", RFC 2060, dez. 1996. http://www.rfc-editor.org/rfc/rfc2060.txt

[RFC 2068] R. Fielding, J. Gettys, J. Mogul, H. Frystyk e T. Berners-Lee. "Hypertext Transfer Protocol – HTTP/1.1", RFC 2068, jan. 1997. http://www.rfc-editor.org/rfc/rfc2068.txt. Substituído pelo RFC 2616.

[RFC 2104] H. Krawczyk, M. Bellare e R. Canetti. "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, fev. 1997. http://www.rfc-editor.org/rfc/rfc2104.txt

[RFC 2109] D. Kristol e L. Montulli. "HTTP State Management Mechanism", RFC 2109, fev. 1997. http://www.rfc-editor.org/rfc/rfc2109.txt

[RFC 2131] R. Droms. "Dynamic Host Configuration Protocol", RFC 2131, mar. 1997. http://www.rfc-editor.org/rfc/rfc2109.txt

[RFC 2136] P. Vixie, S. Thomson,Y. Rekhter e J. Bound. "Dynamic Updates in the Domain Name System", RFC 2136, abr. 1997. http://www.rfc-editor.org/rfc/rfc2136.txt

[RFC 2153] W. Simpson. "PPP Vendor Extensions", RFC 2153, maio 1997. http://www.rfc-editor.org/rfc/rfc2153.txt

[RFC 2186] K. Claffy e D. Wessels. "Internet Caching Protocol (ICP), Version 2", RFC 2186, set. 1997. http://www.rfc-editor.org/rfc/rfc2186.txt

[RFC 2189] A. Ballardie. "Core Based Trees (CBT Version 2) Multicast Routing: Protocol Specification", RFC 2189, set. 1997. http://www.rfc-editor.org/rfc/rfc2189.txt

[RFC 2201] A. Ballardie. "Core Based Trees (CBT) Multicast Routing Architecture", RFC 2201, set. 1997. http://www.rfc-editor.org/rfc/rfc2201.txt

[RFC 2205] R. Braden (ed.), L. Zhang, S. Berson, S. Herzog e S. Jamin. "Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification", RFC 2205, set. 1997. http://www.rfc-editor.org/rfc/rfc2205.txt

[RFC 2210] J. Wroclawski. "The Use of RSVP with IETF Integrated Services", RFC 2210, set. 1997. http://www.rfc-editor.org/rfc/rfc2210.txt

[RFC 2211] J. Wroclawski. "Specification of the Controlled-Load Network Element Service", RFC 2211, set. 1997. http://www.rfc-editor.org/rfc/rfc2211.txt

[RFC 2212] S. Shenker, C. Partridge e R. Guerin. "Specification of Guaranteed Quality of Service", RFC 2212, set. 1997. http://www.rfc-editor.org/rfc/rfc2212.txt

[RFC 2215] S. Shenker e J. Wroclawski. "General Characterization Parameters for Integrated Service Network Elements", RFC 2215, set. 1997. http://www.rfc-editor.org/rfc/rfc2215.txt

[RFC 2225] M Laubach e J. Halpern. "Classical UP and ARP over ATM", RFC 2225, abr. 1998. http://www.rfc-editor.org/rfc/rfc2225.txt

[RFC 2246] T. Dierks e C. Allen. "The TLS Protocol", RFC 2246, jan. 1998. http://www.rfc-editor.org/rfc/rfc2246.txt

[RFC 2253] M. Wahl, S. Kille e T. Howes. "Lightweight Directory Access Protocol (v3)", RFC 2253, dez. 1997. http://www.rfc-editor.org/rfc/rfc2253.txt

[RFC 2284] L. Blunk e J. Vollbrecht. "PPP Extensible Authentication Protocol (EAP)", RFC 2284, mar. 1998. ftp://ftp.rfc-editor.org/in-notes/rfc2284.txt

[RFC 2326] H. Schulzrinne, A. Rao e R. Lanphier. "Real Time Streaming Protocol (RTSP)", RFC 2326, abr. 1998. http://www.rfc-editor.org/rfc/rfc2326.txt

[RFC 2328] J. Moy. "OSPF Version 2", RFC 2328, abr. 1998. http://www.rfc-editor.org/rfc/rfc2328.txt

[RFC 2362] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma e L. Wei. "Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification", RFC 2362, jun. 1998. http://www.rfc-editor.org/rfc/rfc2362.txt

[RFC 2373] R. Hinden e S. Deering. "IP Version 6 Addressing Architecture", RFC 2373, jul. 1998. http://www.rfc-editor.org/rfc/rfc2373.txt

[RFC 2400] J. Postel e J. Reynolds. "Internet Official Protocol Standards", RFC 2400, set. 1998. http://www.rfc-editor.org/rfc/rfc2400.txt. Substituído pelo RFC 2500.

[RFC 2401] S. Kent e R. Atkinson. "Security Architecture for the Internet Protocol." RFC 2401, nov. 1998. http://www.rfc-editor.org/rfc/rfc2401.txt

[RFC 2402] S. Kent e R. Atkinson. "IP Authentication Header", RFC 2402, nov. 1998. http://www.rfc-editor.org/rfc/rfc2402.txt

[RFC 2405] C. Madson e N. Doraswamy. "The ESP DES-CBC Cipher Algorithm with Explicit IV", RFC 2405, nov. 1998. http://www.rfc-editor.org/rfc/rfc2405.txt

[RFC 2406] S. Kent e R. Atkinson. "IP Encapsulating Security Payload (ESP)", RFC 2406, nov. 1998. http://www.rfc-editor.org/rfc/rfc2406.txt

[RFC 2407] D. Piper. "The Internet IP Security Domain of Interpretation for ISAKMP", RFC 2407, nov. 1998. http://www.rfc-editor.org/rfc/rfc2407.txt

[RFC 2408] D. Maughan, M. Schertler, M. Schneider e J. Turner. "Internet Security Association and Key Management Protocol (ISAKMP)", RFC 2408, nov. 1998. http://www.rfc-editor. org/rfc/rfc2408.txt

[RFC 2409] D. Harkins e D. Carrel. "The Internet Key Exchange (IKE)", RFC 2409, nov. 1998. http://www.rfc-editor.org/rfc/rfc2409.txt

[RFC 2411] R. Thayer, N. Doraswamy e R. Glenn. "IP Security Document Road Map", RFC 2411, nov. 1998. http://www.rfc-editor.org/rfc/rfc2411.txt

[RFC 2420] H. Kummert. "The PPP Triple-DES Encryption Protocol (3DESE)", RFC 2420, set. 1998. http://www.rfc-editor.org/rfc/rfc2420.txt

[RFC 2421] G. Vaudreuil e G. Parsons. "Voice Profile for Internet Mail – version 2", RFC 2421, set. 1998. http://www.rfc-editor.org/rfc/rfc2421.txt

[RFC 2427] C. Brown e A. Malis. "Multiprotocol Interconnect over Frame Relay", RFC 2427, set. 1998. http://www.rfc-editor.org/rfc/rfc2427.txt

[RFC 2437] B. Kaliski e J. Staddon. "PKCS #1: RSA Cryptography Specifications, Version 2", RFC 2437, out. 1998. http://www.rfc-editor.org/rfc/rfc2437.txt

[RFC 2453] G. Malkin. "RIP Version 2", RFC 2453, nov. 1998. http://www.rfc-editor.org/rfc/rfc2453.txt.

[RFC 2460] S. Deering e R. Hinden. "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, dez. 1998. http://www.rfc-editor.org/rfc/rfc2460.txt

[RFC 2463] A. Conta e S. Deering. "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6)", RFC 2463, dez. 1998. http://www.rfc-editor.org/rfc/rfc2463.txt

[RFC 2474] K. Nicols, S. Blake, F. Baker e D. Black. "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, dez. 1998. http://www.rfc-editor.org/rfc/rfc2473.txt

[RFC 2475] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang e W. Weiss. "An Architecture for Differentiated Services." RFC 2475, dez. 1998. http://www.rfc-editor.org/rfc/rfc2475.txt

[RFC 2481] K.K. Ramakrishnan e S. Floyd. "A Proposal to Add Explicit Congestion Notification (ECN) to IP", RFC 2481, jan. 1999. http://www.rfc-editor.org/rfc/rfc2481.txt

[RFC 2500] J. Reynolds e R. Braden. "Internet Official Protocol Standards", RFC 2500, jun. 1999. http://www.rfc-editor.org/rfc/rfc2500.txt.

[RFC 2535] D. Eastlake. "Domain Name System Security Extensions", RFC 2535, mar. 1999. ftp://ftp.rfc-editor.org/in-notes/rfc2535.txt

[RFC 2578] K. McCloghrie, D. Perkins e J. Schoenwaelder. "Structure of Management Information Version 2 (SMIv2)", RFC 2578, abr. 1999. http://www.rfc-editor.org/rfc/rfc2578.txt

[RFC 2579] K. McCloghrie, D. Perkins e J. Schoenwaelder. "Textual Conventions for SMIv2", RFC 2579, abr. 1999. http://www.rfc-editor.org/rfc/rfc2579.txt

[RFC 2580] K. McCloghrie, D. Perkins e J. Schoenwaelder. "Conformance Statements for SMIv2", RFC 2580, abr. 1999. http://www.rfc-editor.org/rfc/rfc2580.txt

[RFC 2581] M. Allman, V. Paxson e W. Stevens. "TCP Congestion Control", RFC 2581, abr. 1999. http://www.rfc-editor.org/rfc/rfc2581.txt

[RFC 2582] S. Floyd e T. Henderson. "The NewReno Modification to TCP's Fast Recovery Algorithm", RFC 2582, abr. 1999. ftp://ftp.isi.edu/in-notes/rfc2582.txt

[RFC 2597] J. Heinanen, F. Baker, W. Weiss e J. Wroclawski. "Assured Forwarding PHB Group", RFC 2597, jun. 1999. http://www.rfc-editor.org/rfc/rfc2597.txt

[RFC 2598] V. Jacobson, K. Nichols e K. Poduri. "An Expedited Forwarding PHB", RFC 2598, jun. 1999. http://www.rfc-editor.org/rfc/rfc2598.txt

[RFC 2616] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee e R. Feilding. "Hypertext Transfer Protocol – HTTP/1.1", RFC 2616, jun. 1999. http://www.rfc-editor.org/rfc/rfc2616.txt

[RFC 2638] K. Nichols, V. Jacobson e L. Zhang. "A Two-Bit Differentiated Services Architecture for the Internet", RFC 2638, jul. 1999. http://www.rfc-editor.org/rfc/rfc2638.txt

[RFC 2644] D. Senie. "Changing the Default for Directed Broadcasts in Router", RFC 2644, ago. 1999. http://www.rfc-editor.org/rfc/rfc2644.txt

[RFC 2663] P. Srisuresh e M. Holdrege. "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663.

[RFC 2715] D. Thaler. "Interoperability Rules for Multicast Routing Protocols", RFC 2715, out. 1999. http://www.rfc-editor.org/rfc/rfc2715.txt

[RFC 2716] B. Aboba e D. Simon. "PPP EAP TLS Authentication Protocol", RFC 2716, out. 1999. ftp://ftp.rfc-editor.org/in-notes/rfc2716.txt

[RFC 2733] J. Rosenberg e H. Schulzrinne. "An RTP Payload Format for Generic Forward Error Correction", RFC 2733, dez. 1999. http://www.rfc-editor.org/rfc2733.txt

[RFC 2821] J. Klensin (ed.) "Simple Mail Transfer Protocol", RFC 2821, abr. 2001. http://www.rfc-editor.org/rfc/rfc2821.txt

[RFC 2827] P. Ferguson e D. Senie. "Network Ingress Filtering: Defeating Denial of Service Attacks which Employ IP Source Address Spoofing", RFC 2827, maio 2000. http://www.rfc-editor.org/rfc/rfc2827.txt

[RFC 2893] R. Gilligan e E. Nordmark. "Transition Mechanisms for IPv6 Hosts and Routers", RFC 2893, ago. 2000. http://www.rfc-editor.org/rfc/rfc2893.txt

[RFC 2961] L. Berger, D. Gan, G. Swallow, P. Pan, F. Tommasi e S. Molendini. "RSVP Refresh Overhead Reduction Extensions", RFC 2961, abr. 2001. ftp://ftp.rfc-editor.org/in-notes/rfc3260.txt

[RFC 2988] V. Paxson e M. Allman. "Computing TCP's Retransmission Timer." RFC 2988, nov. 2000. ftp://ftp.isi.edu/in-notes/rfc2988.txt

[RFC 3022] P. Srisuresh e K. Egevang. "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, jan. 2001. http://www.rfc-editor.org/rfc/rfc3022.txt

[RFC 3031] E. Rosen, A. Viswanathan e R. Callon. "Multiprotocol Label Switching Architecture", RFC 3031, jan. 2001. ftp://ftp.rfc-editor.org/in-notes/rfc3031.txt.

[RFC 3032] E. Rosen, D. Tappan, G. Fedorkow, Y. Rekhter, D. Farinacci, T. Li e A. Conta. "MPLS Label Stack Encoding", RFC 3032, jan. 2001. ftp://ftp.rfc-editor.org/in-notes/rfc3032.txt

[RFC 3052] M. Eder e S. Nag. "Service Management Architectures Issues and Review", RFC 3052, jan. 2001, http://www.rfc-editor.org/rfc/rfc3139.txt

[RFC 3139] L. Sanchez, K. McCloghrie e J. Saperia. "Requirements for Configuration Management of IP-Based Networks", RFC 3139, jun. 2001. http://www.rfc-editor.org/rfc/rfc3139.txt

[RFC 3209] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan e G. Swallow. "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, dez. 2001. ftp://ftp.rfc-editor.org/in-notes/rfc3209.txt

[RFC 3221] G. Huston. "Commentary on Inter-Domain Routing in the Internet", RFC 3221, dez. 2001. ftp://ftp.rfc-editor.org/in-notes/rfc3221.txt

[RFC 3232] J. Reynolds. "Assigned Numbers: RFC 1700 is Replaced by an On-line Database", RFC 3232, jan. 2002. http://www.rfc-editor.org/rfc/rfc3232.txt

[RFC 3260] D. Grossman. "New Terminology and Clarifications for Diffserv", RFC 3260, abr. 2002. ftp://ftp.rfc-editor.org/in-notes/rfc3260.txt

[RFC 3261] J. Rosenberg, H. Schulzrinne, G. Carmarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley e E. Schooler. "SIP: Session Initiation Protocol", RFC 3261, jul. 2002. http://www.rfc-editor.org/rfc/rfc3261.txt

[RFC 3344] C. Perkins (ed.) "IP Mobility Support for IPv4", RFC 3344, out. 2002. ftp://ftp.rfc-editor.org/in-notes/rfc3344.txt

[RFC 3346] J. Boyle, V. Gill, A. Hannan, D. Cooper, D. Awduche, B. Christian e W.S. Lai. "Applicability Statement for Traffic Engineering with MPLS", RFC 3346, ago. 2002. ftp://ftp.rfc-editor.org/in-notes/rfc3346.txt

[RFC 3376] B. Cain, S. Deering, I. Kouvelas, B. Fenner e A. Thyagarajan. "Internet Group Management Protocol, Version 3", RFC 3376, out. 2002. ftp://ftp.rfc-editor.org/in-notes/rfc3376.txt

[RFC 3390] M. Allman, S. Floyd e C. Partridge. "Increasing TCP's Initial Window", RFC 3390, out. 2002. ftp://ftp.rfc-editor.org/in-notes/rfc3390.txt.

[RFC 3410] J. Case, R. Mundy, D. Partain e D. Partain. "Introduction and Applicability Statements for Internet Standard Management Framework", RFC 3410, dez. 2002. ftp://ftp.rfc-editor.org/in-notes/rfc3410.txt

[RFC 3411] D. Harrington, R. Presuhn e B. Wijnen. "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", RFC 3411, dez. 2002. ftp://ftp.rfc-editor.org/in-notes/rfc3411.txt

[RFC 3414] U. Blumenthal e U. Blumenthal. "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", RFC 3414, dez. 2002.

[RFC 3415] B. Wijnen, R. Presuhn e K. McCloghrie. "View-Based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)", RFC 3415, dez. 2002. ftp://ftp.rfc-editor.org/in-notes/rfc3415.txt

[RFC 3416] R. Presuhn, J. Case, K. McCloghrie, M. Rose e S. Waldbusser. "Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)", dez. 2002. ftp://ftp.rfc-editor.org/in-notes/rfc3416.txt

[RFC 3468] L. Andersson e G. Swallow. "The Multiprotocol Label Switching (MPLS) Working Group Decision on MPLS Signaling Protocols", RFC 3468, fev. 2003. ftp://ftp.rfc-editor.org/in-notes/rfc3468.txt

[RFC 3469] V. Sharma e F. Hellstrand (eds.) "Framework for Multi-Protocol Label Switching (MPLS)-based Recovery", RFC 3469, fev. 2003. ftp://ftp.rfc-editor.org/in-notes/rfc3469.txt

[RFC 3550] H. Schulzrinne, S. Casner, R. Frederick e V. Jacobson. "RTP: A Transport Protocol for Real-Time Applications", RFC 3550, jul. 2003. ftp://ftp.rfc-editor.org/in-notes/rfc3550.txt

[RFC 3569] S. Bhattacharyya (ed.) "An Overview of Source-Specific Multicast (SSM)", RFC 3569, jul. 2003. ftp://ftp.rfc-editor.org/in-notes/rfc3569.txt

[RFC 3588] P. Calhoun, J. Loughney, E. Guttman, G. Zorn e J. Arkko. "Diameter Base Protocol", set. 2003. ftp://ftp.rfc-editor.org/in-notes/rfc2716.txt

[RFC 3600] J. Reynolds e 6 S. Ginoza, 6. "Internet Official Protocol Standards", RFC 3600, nov. 2003. ftp://ftp.rfc-editor.org/in-notes/rfc3600.txt

[RFC 3649] S. Floyd. "HighSpeed TCP for Large Congestion Windows", RFC 3649, dez. 2003. ftp://ftp.rfc-editor.org/in-notes/rfc3649.txt.

[Rhee 1998] I. Rhee. "Error Control Techniques for Interactive Low-Bit Rate Video Transmission over the Internet." *Proceedings ACM SIGCOMM'98*. Vancouver, 31 ago.-4 set. 1998. http://www.acm.org/sigcomm/sigcomm98/tp/abs_24.html

[Roberts 1967] L. Roberts e T. Merril. "Toward a Cooperative Network of Time-Shared Computers." *AFIPS Fall Conference*, out. 1966.

[Rom 1990] R. Rom e M. Sidi. *Multiple Access Protocols: Performance and Analysis*. Nova York: Springer-Verlag, 1990.

[Root Servers 2004] http://www.root-servers.org/

[Rose 1996] M. Rose. *The Simple Book: An Introduction to Internet Management*, 2 ed. rev. Englewood Cliffs: Prentice Hall, 1996.

[Rosenberg 2000] J. Rosenberg, L. Qiu e H. Schulzrinne. "Integrating Packet FEC into Adaptive Playout Buffer Algorithms on the Internet." *IEEE INFOCOM 2000*. Tel Aviv, 2000.

[Ross 1995] K.W. Ross. *Multiservice Loss Models for Broadband Telecommunication Networks*. Berlim: Springer, 1995.

[Ross 2003] K.W. Ross. "Asynchronous Voice: A Personal Account." *IEEE Multimedia*, abr./jun. 2003, p. 70-74.

[Rowston 2001] A. Rowston e P. Druschel. "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems." *Proceedings of IFIP/ACM Middleware 2001*. Heidelberg, 2001.

[RSA 1978] R.L. Rivest, A. Shamir e L.M. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems." *Communications of the ACM*, vol. 21, n. 2, fev. 1978, p. 120-126.

[RSA Challenge 2002] RSA Data Security Inc. "What Is the RSA Secret Key Challenge?" http://www.rsasecurity.com/rsalabs/faq/2-4-4.html

[RSA FAQ 2004] RSA Inc. "RSA Laboratories' Frequently Asked Questions About Today's Cryptography, Version 4.1." http://www.rsasecurity.com/rsalabs/faq

[RSA Fast 2004] RSA Laboratories. "How Fast Is RSA?" http://www.rsasecurity.com/rsalabs/faq/3-1-2.html

[RSA Key 2004] RSA Laboratories. "How Large a Key Should Be Used in the RSA Crypto System?" http://www.rsasecurity.com/rsalabs/faq/3-1-5.html

[Rubenstein 1998] D. Rubenstein, J. Kurose e D. Towsley "Real-Time Reliable Multicast Using Proactive Forward Error Correction." *Proceedings of NOSSDAV '98*. Cambridge, jul. 1998. http://gaia.cs.umass.edu/pub/Rubenst98:proact.ps.gz

[Rubin 2001] A. Rubin. *White-Hat Security Arsenal: Tackling the Threats*. Addison Wesley, 2001.

[Saltzer 1984] J. Saltzer, D. Reed e D. Clark. "End-to-End Arguments in System Design." *ACM Transactions on Computer Systems (TOCS)*, vol. 2, n. 4, nov. 1984.

[Saroiu 2002] S. Saroiu, K. Gummadi, R. Dunn, S. Gribble e H. Levy. "An Analysis of Internet Content Delivery Systems." *Proceedings of Usenix OSDI 2002*, p. 315-328. http://www.usenix.org/events/osdi02/tech/saroiu/saroiu_html/index.html

[Savage 1999] S. Savage, A. Collins, E. Hoffman, J. Snell e T. Anderson. "The End-to-End Effects of Internet Path Selection." *Proceedings of 1999 ACM SIGCOMM*. Boston, set. 1999.

[Savage 2000] S. Savage, D. Wetherall, A. Karlin e T. Anderson. "Practical Network Support for IP Traceback." *Proceedings of the 2000 ACM SIGCOMM Conference*, Estocolmo, ago. 2000, p. 295-306. http://www.cs.washington.edu/homes/savage/papers/Sigcomm00.pdf

[Saydam 1996] T. Saydam e T. Magedanz. "From Networks and Network Management into Service and Service Management." *Journal of Networks and System Management*, vol. 4, n. 4, dez. 1996, p. 345-348.

[Schneier 1995] B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, 1995.

[Schulzrinne 1997] H. Schulzrinne. "A Comprehensive Multimedia Control Architecture for the Internet." *NOSSDAV'97 (Network and Operating System Support for Digital Audio and Video)*. St. Louis, 19 maio 1997. http://www.cs.columbia.edu/~hgs/papers/ Schu9705_Comprehensive.ps.gz

[Schulzrinne-RTP 2004] Henning Schulzrinne's RTP site. http://www.cs.columbia.edu/~hgs/rtp

[Schulzrinne-RTSP 2004] Henning Schulzrinne's RTSP site. http://www.cs.columbia.edu/~hgs/rtsp

[Schulzrinne-SIP 2004] Henning Schulzrinne's SIP site. http://www.cs.columbia.edu/~hgs/sip

[Schurmann 1996] G. Schurmann. "Multimedia Mail." *ACM Multimedia Systems*, out. 1996, p. 281-295.

[Schwartz 1977] M. Schwartz. *Computer-Communication Network Design and Analysis*. Englewood Cliffs: Prentice-Hall, 1997.

[Schwartz 1980] M. Schwartz. *Information, Transmission, Modulation, and Noise*. Nova York: McGraw Hill, 1980.

[Schwartz 1982] M. Schwartz. "Performance Analysis of the SNA Virtual Route Pacing Control." *IEEE Transactions on Communications*, vol. COM-30, n. 1, jan. 1982, p. 172-184.

[Schwiebert 2001] L. Schwiebert, S. Gupta e J. Weinmann. "Research Challenges in Wireless Networks of Biomedical Sensors." *ACM Mobicom 2001*, 2001, p. 151-165. http://citeseer.ist.psu.edu/schwiebert01 research.html

[Scourias 2001] J. Scourias e T. Farley. "Overview of the Global System for Mobile Communications: GSM." http://www.privateline.com/PCS/GSM0.html

[Segaller 1998] S. Segaller. *Nerds 2.0.1, A Brief History of the Internet*. TV Books, 1998.

[Semeria 1996] C. Semeria. "Understanding IP Addressing: Everything You Ever Wanted to Know." http://www.3com.com/nsc/501302s.html

[Shacham 1990] N. Shacham e P. McKenney. "Packet Recovery in High-Speed Networks Using Coding and Buffer Management." *Proceedings of IEEE Infocom Conference*. São Francisco, 1990, p. 124-131.

[Sharma 1997] Puneet Sharma, Deborah Estrin, Sally Floyd e Van Jacobson. "Scalable Timers for Soft State Protocols." *Proceedings of IEEE Infocom '97 Conference*, Kobe, abr. 1997.

[Shipley 2001] P. Shipley. "Open WLANS: The Early Results of War Driving." http://www.dis.org/filez/openlans.pdf

[Sidor 1998] D. Sidor. "TMN Standards: Satisfying Today's Needs While Preparing for Tomorrow." *IEEE Communications Magazine*, vol. 36, n. 3, mar. 1998, p. 54-64.

[Singh 1999] S. Singh. *The Code Book: The Evolution of Secrecy from Mary, Queen of Scotsto Quantum Cryptography*. Doubleday Press, 1999.

[SIP Software 2004] H. Schulzrinne. Software Package site. http://www.cs.columbia.edu/IRT/software

[SMIL 2004] W3C Synchronized Multimedia homepage. http://www.w3.org/AudioVideo

[Snoeren 2001] A. Snoeren, C. Partridge, L. Sanchez, C. Jones, F. Tchàkountio, S. Kent e W.T. Strayer. "Hash-Based IP Traceback." *Proceedings of the 2001 ACM Sigcomm*. http://www.acm.org/sigcomm/ sigcomm2001/p1-snoeren.pdf

[Solari 1997] S.J. Solari. *Digital Video and Audio Compression*. Nova York: McGraw Hill, 1997.

[Solensky 1996] F. Solensky. "IPv4 Address Lifetime Expectations." In: S. Bradner e A. Mankin (eds.) *IPng: Internet Protocol Next Generation*. Reading: Addison Wesley, 1996.

[Spragins 1991] J.D. Spragins. *Telecommunications Protocols and Design*. Reading: Addison Wesley 1991.

[Sprint 2004] Sprint. "Network Overview: SLAs." http://www.sprintbiz.com/about/network_slas.html

[Spurgeon 2002] C. Spurgeon. "Charles Spurgeon's Ethernet Web Site." http://wwwhost.ots.utexas.edu/ ethernet/ethernet-home.html

[Srinivasan 1999] V. Srinivasan e G. Varghese. "Fast Address Lookupp Using Controlled Prefix Expansion." *ACM Transactions Computer Sys.*, vol 17, n. 1, fev. 1999, p. 1-40.

[Stallings 1993] W. Stallings. *SNMP, SNMP v2, and CMIP: The Practical Guide to Network Management Standards*. Reading: Addison Wesley, 1993.

[Stallings 1999] W. Stallings. *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*. Reading: Addison Wesley, 1999.

[Steinder 2002] M. Steinder e A. Sethi. "Increasing Robustness of Fault Localization through Analysis of Lost, Spurious, and Positive Symptoms." *Proceedings of IEEE INFOCOM*, 2002. http://www.ieee-infocom.org/2002/papers/665.pdf

[Stevens 1990] W.R. Stevens. *Unix Network Programming*. Englewood Cliffs: Prentice-Hall, 1990.

[Stevens 1994] W.R. Stevens. *TCP/IP Illustrated, Vol. 1: The Protocols*. Reading: Addison Wesley, 1994.

[Stevens 1997] W.R. Stevens. *Unix Network Programming, volume 1: Networking APIs-Sockets and XTI*, 2ed. Englewood Cliffs: Prentice-Hall, 1997.

[Stewart 1999] J. Stewart. *BGP4: Interdomain Routing in the Internet*. Addison Wesley, 1999.

[Stoica 2001] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek e H. Balakrishnan. "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications." *Proceedings of ACM SIGCOMM 2001*. San Diego, 2001. http://www.acm.org/sigcomm/sigcomm2001/p12.html

[Stoll 1995] C. Stoll, The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage, Pocket Books, 1995.

[Stone 1998] J. Stone, M. Greenwald, C. Partridge e J. Hughes. "Performance of Checksums and CRC's Over Real Data." *IEEE/ACM Transactions on Networking*, vol. 6, n. 5, out. 1998, p. 529-543.

[Stone 2000] J. Stone e C. Partridge. "When Reality and the Checksum Disagree." *Proceedings of ACM SIGCOMM '00*. Estocolmo, ago. 2000.

[Strayer 1992] W.T. Strayer, B. Dempsey e A. Weaver. *XTP: The Xpress Transfer Protocol*. Reading: Addison Wesley, 1992.

[Stubblefield 2002] A. Stubblefield, J. Ioannidis e A. Rubin. "Using the Fluhrer, Mantin, and Shamir Attack to Break WEP." *Proceedings of the 2002 Network and Distributed Systems Security Symposium*, 2002, p. 17-22. http://www.cs.rice.edu/~astubble/wep/wep_attack.pdf

[Subramanian 2000] M. Subramanian. *Network Management: Principles and Practice*. Reading: Addison Wesley, 2000.

[Subramanian 2002] L. Subramanian, S. Agarwal, J. Rexford e R. Katz. "Characterizing the Internet Hierarchy from Multiple Vantage Points." *Proceedings of the 2002 IEEE Infocom*.

[Sun 2004] Sun Microsystems. "System and Network Management." http://www.sun.com/products-n-solutions/software/management/

[Sunshine 1978] C. Sunshine e Y.K. Dalal. "Connection Management in Transport Protocols." *Computer Networks*. Amesterdã, 1978.

[T-Mobile 2004] T-Mobile HotSpot US Location Map. http://locations.hotspot.t-mobile.com

[Tangmunarunkit 2001] H. Tangmunarunkit, R. Govindan, D. Estrin e S. Shenker. "The Impact of Routing Policy on Internet Paths." *Proceedings of the 2001 IEEE INFOCOM*. Alasca, abr. 2001. http://www.isi.edu/~hongsuda/publication/info2001.ps

[TechnOnLine 2004] TechOnLine. "Protected Wireless Networks." Online Webcast Tutorial. http://www.techonline.com/community/tech_topic/internet/21752

[Teleography 2002] Teleography – a Research Division of Pirmetrica. "WorldCom Controls the Most Internet Bandwidth, Connections, and Revenue." http://www.telegeography.com/press/releases/2002/10-jul-2002.html.

[Thaler 1997] D. Thaler e C. Ravishankar. "Distributed Center-Location Algorithms." *IEEE Journal on Selected Areas in Communications*, vol. 15, n. 3, abr. 1997, p. 291-303.

[Think 2002] Technical History of Network Protocols. "Cyclades." http://www.cs.utexas.edu/users/jeffo/cs370/CYCLADES/jeffo/cyclades/index.htm

[Thinplanet 2002] Thinplanet homepage. http://www.thinplanet.com/

[Thottan 1998] M. Thottan e C. Ji. "Proactive Anomaly Detection Using Distributed Intelligent Agents." *IEEE Network Magazine*, vol. 12, n. 5, set./out. 1998, p. 21-28.

[Tobagi 1990] F. Tobagi. "Fast Packet Switch Architectures for Broadband Integrated Networks." *Proceedings of the IEEE*, vol. 78, n. 1, jan. 1990, p. 133-167.

[Turner 1986] J. Turner. "New Directions in Communications (or Which Way to the Information Age?)." *Proceedings of the Zürich Seminar on Digital Communication*. Zurique, mar. 1986, p. 25-32.

[Turner 1988] J.S. Turner "Design of a Broadcast Packet Switching Network." *IEEE Transactions on Communications*, vol. 36, n. 6, jun. 1988, p. 734-743.

[Utah 2004] Utah Division of Corporations and Commercial Codes. "Digital Signature Licensing Information." http://www.commerce.state.ut.us/corporat/dsmain.htm

[Varghese 1997] G. Varghese e A. Lauck. "Hashed and Hierarchical Timing Wheels: Efficient Data Structures for Implementing a Timer Facility." *IEEE/ACM Transactions on Networking*, vol. 5, n. 6, dez. 1997, p. 824-834.

[Verisign 2004] http://www.verisign.com

[Verizon 2004] Verizon Communication. Verizon Broadband Anytime. http://www.verizon.net/wifi/

[Verma 2001] D.C. Verma. *Content Distribution Networks: An Engineering Approach*. John Wiley, 2001.

[Viterbi 1995] A. Viterbi. *CDMA: Principles of Spread Spectrum Communication*. Reading: Addison Wesley, 1995.

[VON 2004] Voice on the Net. http://www.von.com

[von Lohmann 2003] F. von Lohmann. "Peer-to-Peer File Sharing and Copyright Law: A Primer for Developers." *2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*. Berkeley, 2003. http://iptps03.cs.berkeley.edu/final-papers/copyright.pdf

[Voydock 1983] V.L. Voydock e S.T. Kent. "Security Mechanisms in High-Level Network Protocols." *ACM Computing Surveys*, vol. 15, n. 2, jun. 1983, p. 135-171.

[W3C 1995] The World Wide Web Consortium. "A Little History of the World Wide Web." 1995. http://www.w3.org/History.html

[WAP 2004] WAP Forum. "WAP 2.0 Technical White Paper." http://www.wapforum.org/what/whitepapers.htm.

[Wakeman 1992] Ian Wakeman, Jon Crowcroft, Zheng Wang e Dejan Sirovica. "Layering Considered Harmful." *IEEE Network*, jan. 1992, p. 20-24.

[Waldvogel 1997] M. Waldvogel *et al.* "Scalable High Speed IP Routing Lookup." *Proceedings of ACM SIGCOMM '97*. Cannes, set. 1997. http://www.acm.org/sigs/sigcomm/sigcomm97/papers/p182.html

[Walker 2000] J. Walker. "IEEE P802.11 Wireless LANs, Unsafe at Any Key Size; An Analysis of the WEP Encapsulation", out. 2000. http://www.drizzle.com/~aboba/IEEE/0-362.zip

[Weatherspoon 2000] S. Weatherspoon. "Overview of IEEE 802.11b Security." *Intel Technology Journal*, 2nd Quarter 2000. http://developer.intel.com/technology/itj/q22000/articles/art_5.htm

[Web ProForum 1999] Web ProForum. "Tutorial on H.323", 1999. http://www.webproforum.com/h323/index.html

[Wei 2004] W. Wei, B. Wang, J. Kurose e D. Towsley. "Detecting and Distinguishing Wired and Wireless Packet Losses in an End-End Connection." *Technical Report*, Uviversidade de Massachusetts (Departamento de Ciências da Computação), 2004.

[Weinstein 2002] S. Weinstein. "The Mobile Internet:Wireless LAN vs. 3G Cellular Mobile." *IEEE Communications Magazine*, fev. 2002. p. 26-28.

[Weiser 1991] M. Weiser. "The Computer for the Twenty-First Century." *Scientific American*, set. 1991: 94-10. http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html

[Wessels 2001] D. Wessels. *Web Caching*. Sebastopol: O'Reilly, 2001.

[Wimba 2004] Wimba homepage. http://www.wimba.com

[Woo 1994] T. Woo, R. Bindignavle, S. Su e S. Lam. "SNP: an Interface for Secure Network Programming." *Proceedings of 1994 Summer USENIX*. Boston, jun. 1994, p. 45-58. http://www.cs.utexas.edu/users/lam/Vita/Cpapers/WBSL94.pdf

[Wood 2004] L. Wood. "Lloyds Satellites Constellations." http://www.ee.surrey.ac.uk/Personal/L.Wood/constellations/iridium.html

[Xiao 2000] X. Xiao, A. Hannan, B. Bailey e L. Ni. "Traffic Engineering with MPLS in the Internet." *IEEE Network*, mar./abr. 2000. http://www.cse.msu.edu/~xiaoxipe/papers/mplsTE/mpls.te.pdf

[Yahoo-MIME 1999] Yahoo MIME WWW page. http://dir.yahoo.com/Computers_and_Internet/Multimedia/MIME/

[Yeager 1996] N.J. Yeager e R.E. McGrath. *Web Server Technology*. São Francisco: Morgan Kaufmann Publishers, 1996.

[Zegura 1997] E. Zegura, K. Calvert e M. Donahoo. "A Quantitative Comparison of Graph-based Models for Internet Topology." *IEEE/ACM Transactions on Networking*, vol. 5, n. 6, dez. 1997. http://www.cc.gatech.edu/fac/Ellen.Zegura/papers/ton-model.ps.gz. Veja também http://www.cc.gatech.edu/ projects/gtim, para um pacote de software que gera redes com estrutura realística.

[Zhang 1991] L. Zhang, S. Shenker e D.D. Clark. "Observations on the Dynamics of a Congestion Control Algorithm: The Effects of Two Way Traffic." *Proceedings of ACM SIGCOMM '91*. Zurique, 1991. http://www1.acm.org/pubs/citations/proceedings/comm/115992/p133-zhang/

[Zhang 1993] L. Zhang, S. Deering, D. Estrin, S. Shenker e D. Zappala. "RSVP: A New Resource Reservation Protocol." *IEEE Network Magazine*, vol. 7, n. 9, set. 1993, p. 8-18.

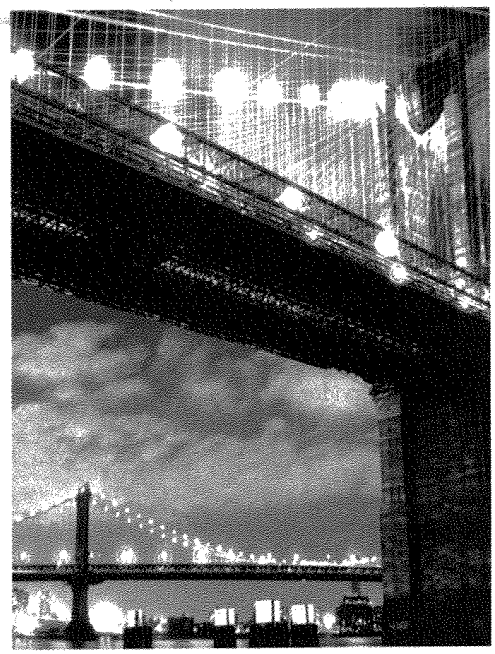
[Zhang 1998] L. Zhang, R. Yavatkar, F. Baker, P. Ford, K. Nichols, M. Speer e Y. Bernet. "A Framework for Use of RSVP with Diff-serv Networks", <draft-ietf-diffservrsvp-01.txt>, 20 nov. 1998. Trabalho em desenvolvimento.

[Zhao 2004] B.Y. Zhao, L. Huang, J. Stribling, S.C. Rhea, A.D. Joseph e J. Kubiatowicz. "Tapestry: A Resilient Global-scale Overlay for Service Deployment." *IEEE Journal on Selected Areas in Communications*, vol. 22, n. 1, jan. 2004. http://www.cs.berkeley.edu/~adj/publications/paper-files/tapestry_jsac.pdf

[Ziff-Davis 1998] Ziff-Davis Publishing. "Ted Nelson: Hypertext pioneer", 1998. http://www.zdnet.com/zdtv/screensavers_story/0,3656,2127396-2102293,00.html

[Zimmermann 2004] P. Zimmermann. "Why do you need PGP?" http://www.pgpi.org/doc/whypgp/en/

Índice



A

Abordagem
  de nó central na construção de spanning trees, 305
  TLV, 591
Abramson, Norman, 42
Abstração de corrente de bytes, 10
Acesso
  aleatório, protocolos de, 338, 341-346
  corporativo, 20, 22
  dedicado, 369
  enlaces, 328
  firewalls, 541-546
  residencial, 20-21
  sem fio, 20, 22-24
  múltiplo por divisão de código (CDMA), 340, 398-401, 415
Acesso celular à Internet, 413-418
  gerenciamento, 429-433
Acesso múltiplo
  problemas do, 338
  protocolos de, 337-348
Acesso múltiplo com deteção de portadora (carrier sense multiple access — CSMA), 344-346
Acesso múltiplo por divisão de código (*code division multiple access* — CDMA), 340, 398-401, 415
ACKs duplicados, 134-191
Ações do SR, 175-176
Acordo de segurança (AS), 558
Acordos de Nível de Serviços (Service Level Agreements (SLAs), 573
Ad hoc, redes, 396
Adaptadores e camada de enlace, 331-332
Adição
  de caches, 80
  de registros, 105
  de reversão envenenada, 286
Agentes
  anúncios de, 427
  correspondentes, 424
  de usuário, 85
  descoberta de, 427, 428
  de gerenciamento de rede, 576
  solicitação de, 427
Agentes externos, 420
  âncoras, 425
Agentes nativos, 420
  encapsulamento, 423
Agregação
  de endereços, 264-265
  de rotas, 264-265
AIMD (algoritmo de aumento aditivo, diminuição multiplicativa — *additive-increase, multiplicative-decrease*), 209
Algoritmos
  de aumento aditivo, diminuição multiplicativa (*additive-increase, multiplicative-decrease* — AIMD), 209
  de *broadcast*, 306
  de comutação de circuitos, 287
  de controle de congestionamento TCP, 208
  de criptografia, 516
  de decriptação, 516
  de Dijkstra, 278
  de estado de enlace (*link-state* — LS), 277, 278-281
  de função de hash, 534-536
  de roteamento global, 277
  de roteamento multicast, 310-311
  de roteamento, 236, 275-290
  de Troca de Chaves da Internet (Internet Key Exchange — IKE), 560
  de vetor de distâncias (*distance-vector* — DV), 281-287
  RSA, 522
  sensíveis à carga, 278
Algoritmos de roteamento, 275-291
  de comutação de circuitos, 287
  de estado de enlace (LS), 278-281
  de vetor de distâncias (DV), 281-287
  descentralizado, 277
  dinâmicos, 277
  estáticos, 277
  mensagens, 286-287
  multicast, 310-311
  robustez, 287
  roteamento hierárquico, 287-290
  velocidade de convergência, 287
ALOHAnet, 41
Analisador de pacotes, 53, 547
Andreesen, Marc, 44
Anúncios, 291-293
  de agentes, 427
  protocolo de informação de roteamento (*routing information protocol* — RIP), 291-292
Anycast, endereços, 271
Apelidos
  de hospedeiros, 97
  de servidores de correio, 97
Aplicação
  arquitetura da, 57-67
  princípios de, 74–87
  processos, 61-63
Aplicações
  auxiliares, 450
  cliente, 8
  cliente/servidor em Java , 116-122
  de multimídia sensíveis a atraso, 442
  de multimídia tolerantes a perda, 442
  de redes multimídia, 442-449
  de redes, 65
  distribuídas, 4, 8
  elásticas, 65
  gateways de, 544-546
  HTTP (HyperText Transfer Protocol — Protocolo de Transferência de Hipertexto), 70-71
  protocolos de transporte, 154
  sensíveis à largura de banda, 64

serviços, 64-65
servidor, 8
SNMP, 586-589
telefone por Internet, 457-465
tolerantes a perda, 64
Web, 63-64
Apresentação
mensagens de, 9
de três vias, 179, 196
ARPAnet, 43
Arquitetura
cliente-servidor, 59
da aplicação, 58-61
da Internet, 1-4. *Veja também Internet*
de camadas, 34-38
de roteadores, 247
do acesso celular à Internet, 413-415
LANs sem fio IEEE 802.11, 402-404
P2P, 60
redes móveis, 539
redes multimídia, 614–620
Arquiteturas híbridas, 60
Arquivos
base HTML, 68
busca de, 109
divisão de dados dos, 182
meta, 451
Protocolo de Transferência de Arquivos (File Transfer Protocol — FTP), 82-83
transferência de, 109
Árvores *unicast* de caminho de menor custo, 322
ASCII, extensões MIME que não seguem o, 90-91
ASN.1, 589-593
Assinaturas digitais, 525-530
Associação
LANs sem fio IEEE 802.11, 402-404
Ataque
ao texto cifrado, 518
com texto aberto conhecido, 518
do homem do meio, 530
Ataques, 518, 546-549. *Veja também Segurança*
com texto escolhido, 518
da brigada do balde, 530
de inundação SYN, 548
de negação de serviço (*denial-of-service* — DoS), 548-549
de negação de serviço (DoS) distribuídos, 548
de reprodução, 527
smurf, 548
Atenuação de percurso, 396
ATM (modo de transferência assíncrono — *assynchronous transfer mode*), 17, 199, 378-382
serviços de rede de taxa de bits, 240-241
Atraso de fila, 15, 28, 29
e perda de pacote, 31-33
Atraso de transmissão, 28, 29
comparado com atraso de propagação, 30-31
Atributos BGP, 298-299
Atraso
aplicações que não são de tempo real, 65
de empacotamento, 389
de processamento, 28, 29
de processamento nodal, 28
de propagação fim-a-fim de canal, 345
de reprodução adaptativo, 431-462
de reprodução e variação de atraso de pacote, 459
de reprodução fixo, 460-461
e rotas, 33-34
fim-a-fim, 33, 459
nodal total, 28
redes de comutação de pacotes, 28-34
taxas de envio do hospedeiro, 200
TCP, 215-216
tempo de viagem de ida e volta (*round-time trip* — RTT), 71
tipos de, 28-31
variação de, 459
Áudio
acesso a, 450-452
compressão de, 448-449
de fluxo contínuo ao vivo, 444
de fluxo contínuo armazenado, 443, 449-457
H.323, 476-477
interativo em tempo real, 444-445
reparação no receptor, 464-465
RTP, 467
RTSP (protocolo de fluxo contínuo em tempo real), 454-457
Áudio e vídeo de fluxo contínuo
armazenados, 443
redes multimídia, 449-457
Áudio e vídeo interativos em tempo real, 444-445
Autenticação, 525-530
de fonte, 558
Autodidatas, comutadores, 368
Auto-regulação, 208
Autoridade certificadora (*certification authority* — CA), 536
Auxiliares, aplicações, 450

B

Backbones, 26-28, 295
hubs de, 364
Bancos de dados
consultas a, 101
inserção de registros em, 105
Bancos de dados centralizados distantes, 98
Bancos de dados distribuídos hierárquicos DNS, 98-101
Baran, Paul, 40
Barramentos, comutação por, 251
Base de Informações de Gerenciamento (Management Information Base — MIB), 576, 582-584
módulos, 579
Bellovin, Steven M., 569-570
Berners-Lee, Tim, 44, 141-142
BITNET, 43
Bits
ACK, 182
paridade, 333
de prioridade de perda de célula (CLP), 379
de prioridade de perda de células (CLP), 379
erros de, 160-166
FIN, 182
não detectados, erros de, 332
NI, 206
PSH, 182
RST, 182
SYN, 182
URG, 182
Blocos de endereços, obtenção de, 264-265
Bloqueio de cabeça de fila (*head-of-the-line* — HOL *blocking*), 254
Bluetooth, 412
Borda, roteadores de, 288
Broadcast
algoritmos de, 306
de estado de enlace, 278
e roteamento, 301-306
endereços de, 350
enlaces, 337
número de seqüência de, 303
por *spanning-tree*, 304-305
tempestades de, 302
Browsers, 44. *Veja também Interfaces*
HTTP (HyperText Transfer Protocol — Protocolo de Transferência de Hipertexto), 68
Buffers
clientes, 453
de envio, 179
de saída, 15
e trajetos com múltiplos roteadores, 202-204
finitos, 201
infinitos, 199-200
Buffers finitos
e trajetos com múltiplos roteadores, 202-204
roteadores com, 201-202
Busca de arquivos, 109
Byte stuffing, 375
Bytes de controle de erro no cabeçalho (HEC), 380

C

Cabeçalhos
campos de comprimento de, 181
de mensagens DNS, 103-104
linhas de, 73, 74
RTP, 466
somas de verificação dos, 257
Cabo coaxial, 25
Cabo
cabo híbrido coaxial/fibra (*hybrid fiber-coaxial cable* — HFC), 20
coaxial, 25
modens a, 21
Cache
DNS, 101-102
Web, 78-79
Cadeias, 115
Caixas postais, 85
Camada de adaptação ATM (*ATM adaptation layer* — AAL), 379
Camada de aplicação, 37
compartilhamento de arquivos P2P, 106-113
construindo servidores Web, 128-132
correio eletrônico, 84-95
DNS, 95-105
FTP, 82-84
HTTP (HyperText Transfer Protocol — Protocolo de Transferência de Hipertexto), 68-82
programação de aplicações com TCP, 113-122
programação de sockets com UDP, 122-128
protocolos de, 63-64
Camada de enlace, 37-38, 328-332
adaptadores, 331-332

Comutação de Rótulos Multiprotocolo (MPLS), 382-384
comutadores de, 364-372
detecção de erros, 332-337
endereçamento na, 348-355
Ethernets, 356-364
hubs, 364-372
protocolo ponto a ponto (PPP), 372-377
protocolos de acesso múltiplo, 337-348
quadros de, 39
reconhecimento (ACK) na, 405
redes ATM, 378-382
redes como, 377-384
serviços, 328-331
Camada de rede, 37, 235-239
algoritmos de roteamento, 275-290
circuitos virtuais (CVs), 241-243, 246
confidencialidade na, 558
datagramas, 39, 244-246
modelos de serviço, 239-241
Protocolo da Internet (IP), 255-275
relação entre camada de transporte e, 145-146
roteadores, 247-254
roteamento *broadcast*, 301-306
roteamento *multicast*, 306-313
Camada de sockets de segurança (SSL), 555-558
Camada de transporte, 37, 143-145
congestionamento, 199, 222
demultiplexação, 147-153
multiplexação, 147-153
segmentos de, 38
serviços de, 143-147
TCP, 178-198
transferência confiável de dados, 157-158
UDP, 153-157
Camada física, 38
Camadas
arquitetura de, 34-38
camada de adaptação ATM (*ATM adaptation layer* — AAL), 379
de aplicação, 37
de enlace, 37-38, 328-332
de protocolo, 34-39
de rede, 37, 235
física, 38
segurança, 550-564
transporte, 37
Camadas horizontais, 35
Caminhos, 3, 277
mais curtos, 277
propagação multivias, 396
RSVP, 500
repasse pelo caminho inverso (RPF), 303
Campos
de comprimento de cabeçalho, 181
de endereço, 409-410
de endereços administrados (COA), 427
de flag, 182
de número de porta da fonte, 149
de número de reconhecimento, 181
de números de seqüência, 181
de opções, 181
de ponteiro de urgência, 182
de soma de verificação, 181
de taxa explícita (ER), 206
ER, 206
Canais
atraso de propagação fim-a-fim, 345
com perda, 164-166
de rádio por satélite, 26
LANs sem fio IEEE 802.11, 402-404
protocolos de divisão de, 338, 339-341
transferência confiável de dados, 159-166
virtuais (CVs), 380
Canais com perda, transferência confiável de dados por, 164-166
Cargas oferecidas, 201
Cargas úteis, 257
LANs sem fio IEEE 802.11, 409
Carona para números de reconhecimento, 184
Case, Jeff, 596-597
CDMA-2000, 417
Phase 1, 417
Células de gerenciamento de recursos (células RM), 205
Central de comutação de unidade móvel (*mobile switching center* — MSC), 414
Central de Comutação para Portal de Serviços Móveis de Borda (Gateway Mobile Services Switching Center — GMSC), 430
Central de distribuição de chaves (KDC), 536-538
Centralizados, diretórios, 107-108
Cerf, Vinton, 42, 180, 325-326
CERN, 44
Certificação, 536-541
Certificados, 540
Chamadas
processos de aceitação de, 485
roteamento de, para usuários móveis, 430-431
Chaves, 516, 521
de sessão única, 538
de sessões, 523
distribuição de, 536-541
privadas, 521
Chaves públicas, 521
certificação de, 538-541
criptografia de, 521-525
sistemas de, 516
Cifras
de César, 517
monoalfabéticas, 517
Circuitos, 12
comutação de, 11-17
circuito virtual (CV), 17
dedicados, 3
Circuitos virtuais (CVs), 17
camada de rede, 241-243, 246
e redes de datagramas, 246
Cisco, 248
Clark, Jim, 44
Clientes, 8, 27, 59. *Veja também Browsers*
aplicações, em Java, 116-122
buffers, 453
Ping, 140
processos, 61-62
requisições de objeto, 78
TCP, 196
Clientes/servidores Ping, 140
Cliente-servidor, arquitetura, 59
CMISE/CMIP, 577
Códigos polinomiais, 335
Colisão
de quadros, 338
detecção de, 344
domínios de, 365
Comandos FTP, 83-84
Compartimentos de tempo, 339
Comportamento por salto, 494, 496-497
Compressão em redes multimídia, 448-449
Comprimento de datagramas, 256
Computação nomádica, 56
Comunicação sem fio, enlaces de, 395
Comunicação
entre processos, 61-63
lógica, 144
segura, 513
Comutação, 11-17
acelerada, 370-371
central de comutação de unidade móvel (MSC), 414
de Rótulos Multiprotocolo (MPLS), 382-384
Comutação de rótulos, roteadores de, 382
Comutadores
autodidatas, 368
e camada de enlace, 364-372
de camada de enlace, 3
de Ethernet, 356
filtragem por, 366-367
repasse por, 366-367
Conexões. *Veja também Acesso*
cabo híbrido coaxial/fibra (*hybrid fiber-coaxial cable* — HFC), 20
de controle, 83
de dados, 83
em sub-redes, 262-263
estabelecimento de, 239
fim-a-fim, 12
linha digital de assinante (*digital subscriber line* — DSL), 20
modem discado, 20
não persistentes, 70-72
par de fios trançado sem blindagem (*unshielded twisted pair* — UTP), 24
persistentes, 70-72
persistentes com/sem paralelismo, 72
ponto–a-ponto, 179
reversão de, 268
sockets de, 115
TCP (Transmission Control Protocol — Protocolo de Controle de Transmissão), 65-66
Conexões TCP paralelas, justiça e, 333-334
Configuracão
de endereços IP, 266
manual do endereço IP, 266
Congestionamento
camada de transporte, 199-283
controle de, 147, 199-206
janelas de congestionamento dinâmicas, 219-222
janelas de congestionamento estáticas, 216-219
janelas de, 207
prevenção de, 209
serviço de controle de fluxo, 193
TCP, 66, 206-222
Congestionamento, pacotes de, 204
Conjunto básico de serviços (*basic service set* — BSS), 402
Construindo servidores Web, 158-159
Consultas, 100
inundação de, 100-110
iterativas, 101
recursivas, 101

Contagem regressiva, temporizadores de, 165
Conteúdo, 78
Contramedidas para ataques, 546-549
Controle
de congestionamento
de congestionamento assistido pela rede, 204
de fluxo, 10, 193-195, 330
Controle, conexões de, 83
Cookies, 76-77
Corpos de entidades, 74
Correção
de erros de repasse (FEC), 334, 462-463
e deteção de erros, 332-337
Correio eletrônico
camada de aplicação, 84-95
formatos de mensagem de, 89-91
pela Web, 95
protocolos de acesso ao, 92-95
segurança, 550-554
SMTP (Simple Mail Transfer Protocol), 64, 85, 86-89
Corrente de bytes, abstração de, 10
Correspondentes, 420
Criptografia, 512, 513, 515-525
algoritmos de, 516
de chaves públicas, 521-525
polialfabética, 518
CSMA com detecção de colisão (CSMA/CD), 344
CSNET (*computer science network* — rede da ciência de computadores), 43
CTS, 406-407
Custos
do congestionamento, 199-204
dos enlaces, 285-286
Cyclades, 42

D

Dados duplicados, pacotes de, 165
Dados, conexões de, 83
DARPA (Defense Advanced Research Projects Agency — Agência de Projetos de Pesquisa Avançada), 42
Datagramas, 38-39
envio, 352-353
formato de, 255-260
IPv6, 271-273
Davies, Donald, 40
de sistema de nome de domínio (DNS), 98
Decriptação, 512, 513
algoritmos de, 516
Demultiplexação, 147-153
na camada de transporte, 147-153
DES triplo (3DES), 520
Descarte
de pacotes, 32
do final da fila, 253
Descoberta de agentes, 427
Destinatários, 164
repetição seletiva (*selective repeat* — SR), 174
Desvanecimento, 398
Deteção. *Veja também Segurança*
aleatória rápida (*random early detection* — RED), 253
de colisão, 344
Deteção de portadora, 344
Dijkstra, algoritmos de, 278
Diretórios centralizados, 107-108
Disciplina
de enfileiramento por varredura cíclica, 487
de escalonamento do enlace, 485
de varredura cíclica de conservação de trabalho, 487
Dispositivos
gerenciados, 575
plug and play, 369
Distribuição
de carga, 97
de chaves, 536-541
DNS, 37, 95-105, 140
bancos de dados distribuídos hierárquicos, 98-101
cache, 101-102
mensagens, 102-105
registros, 102-105
serviços do, 95-98
visão geral do, 98-102
Domínios, 267
de colisão, 364
Roteamento Interdomínio sem Classe (Classless Interdomain Routing — CIDR), 262
DVD, vídeo, 449

E

EAP por LAN (EaPoL), 564
eBGP, 298
Eficiência
da Ethernet, 362
de intervalos, 342
de protocolo, 342
Elásticas, aplicações, 65
Elementos de comutação, 247, 250-251
taxas dos, 252
E-mail pela Web, 95
Empacotamento, atraso de, 389
Encadeamento de blocos de cifras, 519
Encapsulamento, 38
agentes nativos, 423
Encerramento do CV, 243
Endereçamento
de classes cheias, 263
de processos, 63
hierárquico, 264
na camada de enlace, 348-355
Endereço aos cuidados de (*care-of-address* — COA), 421, 427
Endereços
agregação de, 264-265
anycast, 271
campos de, 409-410
de *broadcast*, 350
de LANs, 349
endereçamento de classes cheias, 263
externos, 421
físicos, 349
hospedeiros, 266
indiretos, 307
interfaces, 260-262
IP, 96, 260-268
IP temporários, 266
IP de destino, 257
IP de fonte, 257
MAC, 348-350
obtenção de blocos de, 264-265
permanentes, 421
prefixos, 263
redes móveis, 421-425
Roteamento Interdomínio sem Classes (Classless Interdomain Routing — CIDR), 262-263
SIP, 473
sub-redes, 262-263
tempo de aluguel de, 354
tradução de endereços na rede (NAT), 266-268
Enfileiramento
justo ponderado (WFQ), 487-489
prioritário, 486-487
Enhanced Data Rates for Global Evolution (Melhores Taxas de Dados para Evolução Global — EDGE), 417
Enlaces
acesso aos, 329-330
broadcast, 337
de comunicação, 3
de comunicação sem fio, 395
e redes sem fio, 396-401
OSPF, 296
ponto-a-ponto, 337, 372
por demanda, 16
resolução de problemas, 285-286
Enlaces ponto-a-ponto, 337, 372
LANs sem fio IEEE 802.11 como, 408
Enquadramento
camada de enlace, 329-330
de dados PPP, 374-375
Entidade
gerenciadora, 575
registradora, 105
Entrada, 116
cadeias de, 115
padrão, 116
portas de, 247, 248-250
Entrada, filtragem de, 548
Entrega, 239-240
de melhor esfoço, 146
Envio assegurado, 497
Envio, buffers de, 179
Erros
de bits não detectados, 332
deteção de, 160, 332-337
de repasse, correção de (FEC), 334
na camada de enlace, 330
Escalabilidade da largura de banda, 470-471
Escalonadores de pacotes, 253
Escalonamento em redes multimídia, 485-489
Esgotamento de temporização
duplicação do tempo de, 191
tempo de viagem de ida e volta e, 184-187
Espaçamento Curto Interquadros (Short Inter-frame Spacing — SFIS), 405
Espaçamento Interquadros Distribuído (Distributed Inter-Frame Space — DIFS), 405
Esquemas
de paridade bidimensional, 334
de recuperação de perdas, 462
Estabelecimento
circuitos virtuais (CVs), 243
de conexões, 239
Estações-base, 395-408
Estado de enlace, algoritmo de, 277, 278-281
Estados
do TCP, 196
soft, 502

Estimativa de tempo de viagem de ida e volta, 184-187
Estrutura de Informações de Gerenciamento (Structure of Management Information — SMI), 580-582
Estruturas de gerenciamento padrão da Internet, 578-589
Ethernets, 22
  camada de enlace, 356-364
  conceito de Metcalfe, 42-43
  CSMA/CD, 360-362
  eficiência de, 362
  Gigabit, 363-364
  tecnologias, 362-364
Eventos
  Go-Back-N (GBN), 170-171
  SR, 175
Explorando a heterogeneidade, 110-113
Extensões MIME não-ASCII, 90-91

F

Falsificação, 548
Fases de backoff exponencial, 361
FDDI (interface de dados distribuída de fibra), 347
Ferramentas
  analisadores de pacotes, 53
Fibras óticas, 25
Fila
  atrasos de, 15, 28, 29
  de mensagens, 85
  de saída, 15
Fila, formação de, 252-254
  de requisições, 111
  disciplina de enfileiramento por varredura cíclica, 487
  enfileiramento prioritário, 486-487
  gerenciamento ativo de fila (*active queue management* — AQM), 253
  WFQ (enfileiramento justo ponderado), 487-489
Filtragem
  de entrada, 548
  de pacotes, 542-544
  por comutadores, 366-367
Fim-a-fim, atraso, 33, 459
Fim-a-fim, roteamento, 18
Firewalls, 541-546
  de filtragem de pacotes, 542
Flags
  campos de, 182
  deslocamento, 256
Floyd, Sally, 233-234
Fluxo, 271
  de pacotes, 239-240
Fluxo contínuo, servidores de, 452-454
Fora da banda
  informação de controle, 83
  protocolos, 455
Formato
  de datagramas, 255-260
  de mensagens, 72-76
  de mensagens de e-mail, 89-91
  de mensagens DNS, 104
Fragmentação
  deslocamento de, 256
  do datagrama IP, 258-260
Frequency-hopping spread spectrum (FHSS), 412
FTP, 82-84
  comandos, 83-84
  respostas, 83-84
Função de hash, algoritmos de, 534-536
Funções de medição, 496

G

G.729 (compressão), 449
Garantias de qualidade de serviço, 253
Gargalos em compartilhamento de arquivos P2P, 108
Gateways
  de aplicação, 544-546
  de camada de aplicação, 542
GE Information Services, 42
General Packet Radio Service (Serviço Geral de Pacote por Rádio — GPRS), 417
Geradores, 336
Gerenciados, dispositivos, 575
Gerenciamento
  ASN.1, 589-593
  ativo de fila (*active queue management* — AQM), 253
  células de gerenciamento de recursos (células RM), 205
  da conexão TCP, 195-198
  de desempenho, 573
  de mobilidade, 418-425, 429-433
  estrutura do gerenciamento padrão da Internet, 578-589
  infra-estrutura do, 575-578
  Internet Group Management Protocol (IGMP), 308-310
  visão geral do, 571-575
Gerenciamento ativo de fila (*active queue management* — AQM), 253
Gerenciamento de rede, agentes de, 576
GET condicional, 81-82
Gigabit Ethernet, 363-364
Global System for Mobile Communications (Sistema Global para Comunicações Móveis — GSM), 416
Go-Back-N (GBN), 169-173
  eventos, 170-171
Grafos, 276
Grupos
  Internet Group Management Protocol (IGMP), 308-310
  multicast, 307
GSM, 449

H

H.261, 449
H.323, 476-477
Heterogeneidade, 110-113
HFC (*hybrid fiber-coaxial cable* — cabo híbrido coaxial/fibra), 20
História das redes de computadores, 40-45
Hospedeiros sem fio, 395
Hotmail, 86
HTML, arquivos-base, 68
HTML, arquivos-base, 68
HTTP (Protocolo de Transferência de Hipertexto), 63, 140
  cache Web, 78-80
  camada de aplicação, 68-82
  conexões não persistentes, 70-72
  conexões persistentes, 70-72
  conteúdo, 78
  cookies, 76-77
  GET condicional, 81-82
  SMTP, 89
  TCP, 88
Hubs
  de backbones, 364
  camada de enlace, 364-365
  Ethernets, 356, 363
Hyperlinks, 71

I

iBGP, 298
Identificador
  de canal virtual (VCI), 379
  Identificador de circuito virtual (ID CV), 17
Identificador de Conjunto de Serviços (Service Set Identifier — SSID), 403
Identificadores e deslocamento, 256
IEEE 802.11
  LANs sem fio, 401-412
  segurança em, 560-564
IEEE 802.11i, 563-564
IEEE 802.15 WPAN, 412
IMAP (Internet Mail Access Protocol), 92, 94-95
Implantação de túnel, 274
Importação, políticas de, 299
Inanição, 453
Indicação explícita de congestionamento à frente, 206
Informação
  de estado, 17
  de estado de conexão, 243
Informações de controle na banda, 83
Infra-estrutura
  do gerenciamento, 575-578
  modo de, 396
  LANs sem fio de, 402
Integridade, 530-536
Interação
  de servidores DNS, 100
  no sistema final, 8
  usuário-servidor: cookies, 76-77
Intercalação, 462, 464
Interface de dados distribuída de fibra (FDDI), 347
Interfaces, 44
  de programação de aplicação (*application programming interface* — API), 62
  endereçamento, 260-262
Internet
  acesso celular à, 413-418
  atraso, 28-34
  backbones da, 26-28
  camada de transporte, 146-147
  camadas de protocolo, 34-39
  compartilhamento de arquivos P2P, 106-113
  compressão na, 449
  construindo servidores Web, 128-132
  correio eletrônico, 84-95
  década de 90, 43-45
  desenvolvimentos recentes, 45
  DNS, 95-105
  história das redes de computadores, 40-45
  meios físicos, 24-26
  MIME, 89-91
  padrões da, 4
  perda, 28-34
  programação de aplicações com TCP, 113-123

programação de sockets com UDP, 122-128
protocolos, 5-7
redes 1-4
redes de acesso, 19-24
redes de comutação virtuais, 17-18
redes de datagramas, 17-18
roteamento multicast, 311-313
roteamento, 290-301
serviços de protocolo de transporte, 65-67
serviços não orientados para conexão/orientados para conexão, 8-10
serviços orientados para conexão, 8-10
serviços, 4-5
somas de verificação da, 334-335
suporte para rede multimídia, 446-447
Internet Group Management Protocol (IGMP), 308-310
Intervalos bem-sucedidos, 342
Intranets, 4
Inundação, 302-303
controlada, 303
controlada com número de seqüência, 303
de consultas, 108-110
de consultas de escopo limitado, 109
não controlada, 302
SYN, ataques de, 548
IP. *Ver* Protocolo da Internet
IPv4, 260-268
IPv6, 270-274
IS-136 (Interim Standard — Padrão Provisório), 416
IS-95, 416

J

Janelas
de congestionamento dinâmicas, 219-222
de congestionamento estáticas, 216-219
de recepção, 181, 193
Java
aplicações cliente/servidor em, 116-122
construindo servidores Web, 128-132
programação de sockets com UDP, 122-128
Jitter, 240
Justiça
e TCP, 213-215
e UDP, 215

K

Kahn, Robert, 40, 42, 180
KaZaA, 45, 110-113
Kerberos, 537
Kleinrock, Leonard, 40, 55-56

L

Lam, Simon S., 391-392
LANs sem fio IEEE 802.11, 408
LANs sem fio, 22
Largura de banda
da aplicação, 64-65
do RTPC, 470-471
mínima garantida, 240
Leaky buckets, 489-490
Leitores de correio, 85
Licklider, J.C.R., 40
Linguagem de Markup WAP (WAP Markup Language — WML), 23
Linha digital de assinante (*digital subscriber line* — DSL), 20
Linhas
de estado, 74
de requisição, 73
Loops de roteamento, 285

M

MAC
endereços, 348-350
LANs sem fio IEEE 802.11, 404-411
Mapeamento, 546
e protocolo SNMP, 584-586
Máquina de estado finito (*finite-state machine* — FSM), 159, 170
Máscaras e sub-redes, 261-262
Média móvel exponencial ponderada, 185
Meios
compartilhados, 25
físicos para Internet, 24-26
guiados, 24
não guiados, 24
Memória, comutação por, 250
Memórias de conteúdo endereçável (*content addressable memories* — CAMs), 249
Mensagem de camada de aplicação, 38
Mensagens, 38-39
algoritmos de roteamento, 286-287
analisadores de pacotes, 53
camada de aplicação, 61
comutação de pacotes, 14-15
de apresentação, 9
de descoberta DHCP, 353
de requisição, 72-74
de resposta, 74-76
de sinalização, 243
DNS, 102-105
fila de, 85
HTTP, 72-76
orientadas para receptor, 500
Protocolo de Configuração Dinâmica de Hospedeiros (Dynamic Host Configuration Protocol — DHCP), 353
Protocolo de Mensagesn de Controle da Internet (ICMP), 269-270
recebidas, 91
resposta RIP, 291
resumos de, 532-534
RSVP, 499-500
segurança em, 550-554
SIP, 471-475
trap, 584
Metarquivos, 451
Metcalfe, Bob, 359
Microsoft, browser da, 44
MIME, 89-91
Minitel, servidores, 43
Mobilidade em sub-redes, 411-412
Modelos de serviço camada de rede, 239-241
Modens. *Veja também Acesso; Conexões*
a cabo, 21
discados, 20
Modo
comando-resposta, 584
denso, 312
esparso, 312
promíscuo, 547
Modulação por codificação de pulso (*pulse code modulation* — PCM), 448
Monitoração, 571-573. *Veja também Gerenciamento*
Mosaic, 44
MP3, 449
MPEG 1 de camada 3, 449
MPEG 1, 449
MPEG 2, 449
MPEG 4, 449
MPLS (Comutação de Rótulos Multiprotocolo), 382-384
MSCs âncora, 433
Multicast, grupos, 307
Multiplexação, 147-153
acesso residencial, 21
camada de transporte, 147-153
estatística, 15-17
Multiplexação por divisão de freqüência (*frequency division multiplexing* — FDM), 12, 415
Multiplexação por divisão de tempo (*time-division multiplexing* — TDM), 12, 339-340, 415
Multivias, propagação, 396

N

Não persistentes, conexões, 70-72
Napster, 60
Navegação, 44. *Veja também Interfaces*
Nelson, Ted, 44
Netscape, 44
NI, bits, 206
Níveis de ISPs, 26
NOC, 575
Nomes
canônicos de hospedeiros, 97
de hospedeiros, 95
Nonces, 528
Nós
envio de datagramas, 352-353
roteamento de redes móveis, 422-425
vizinhos, 276
Notação decimal separada por pontos, 260
NSFNET, 43-44
Núcleo, 305
Número
de porta bem-conhecidos, 149
de porta de destino, 149, 152, 181
de porta de fonte, 181
de porta, 63
de roaming, 430
de sistema autônomo (ASN), 298
roaming da estação móvel (MSRN), 430
Números de seqüência, 162
campo de, 181
estudo de caso, 183-184
no protocolo Go-Back-N, 169-170
pacotes, 175-176
para um segmento, 182
protocolos SR, 174
variação de atraso, 459

O

Objetos, 68
conexões persistentes, 72
gerenciados, 576
requisição de, 78
Opções para endereços IP, 257
Opções, campo de, 181
Operações SR, 175-176

OSPF Multicast (MOSPF), 295
OSPF, 291
roteamento intra-AS, 293-296

P
P2P (*peer-to-peer*)
arquitetura, 60
compartilhamento de arquivos, 57, 60, 106-113
trabalho em rede, 45
Pacotes
análise de, 53, 547
ARP, 352-353
comutadores de, 3
de congestionamento, 204
de dados duplicados, 165
descarte de, 32
duplicados, 162
escalonadores de, 253
filtragem de, 542-544
fluxo de, 239-240
números de seqüência, 175
perda de, 15, 252, 458, 462-465
perdidos, 32
prefixos, 245
RTCP, 469-471
RTP, 465
tempo de viagem de ida e volta (*round-trip time* — RTT) de, 71
variação de atraso de, 445, 459
Padrão Avançado de Criptografia (AES), 519-520
Padrão para Criptografia de Dados (DES), 519-520
Padrões para o acesso celular à Internet, 416-418
Página Web, 78-80
Par de fios trançado sem blindagem (*unshielded twisted pair* — UTP), 24
Pares, 27
BGP, 298
Partida lenta, 209
Percurso, atenuação de, 396
Perda
atraso de fila e, 31-33
de pacotes, 15, 252, 462-465
em redes de comutação de pacotes, 28-34
Perfis de tráfego, 495
Períodos de silêncio, 13
Perkins, Charlie, 439-441
Permissões (tokens), 347
PGP, 552, 553-554
Picorredes (piconets), 412
Pilhas
de protocolos, 36
duplas, 273-274
Placas
(cartões) de interface de rede (NICs), 331
de linha, 247
Poda, 311
Política de descarte de pacotes, 486
Políticas de importação, 299
Ponteiro de urgência, campo de, 182
Ponto-a-ponto, conexões, 179
Pontos
de Acesso de Rede (Network Access Points — NAPs), 28
de acesso, 395, 402
de encontro, 305
de presença (*points of presence* — POPs), 28
Pontos de falha únicos, 98
compartilhamento de arquivos P2P, 108
POP3 (Post Office Protocol Version 3), 92, 93-94
Portas
de entrada, 247, 248-250
de saída, 248-249, 252
varredura de, 546
Pouzin, Louis, 42
PPP
camada de enlace, 372-377
enquadramento de dados, 374-375
Prefixos, 245, 263
em BGP, 298
Prevenção de congestionamento, 209
Primeiro a entrar/primeiro a sair (FIFO), 486
Princípios fim-a-fim, 157
Prioridades de incentivo, 112
Privacidade Equivalente Sem Fio (Wired Equivalent Privacy — WEP), 561-562
Problema do roteamento triangular, 424
Problema do terminal oculto, 398
LANs sem fio IEEE 802.11, 406-407
Problemas de auto-carregamento (*bootstrap*), 110
Processadores
de mensagem de interface (IMPs), 40
de roteamento, 248
Programação
baseada em eventos, 172
programação de aplicações com TCP, 113-122
protocolo de transporte confiável, 231-232
sockts, 138-140
Programas Nslookup, 104
Projeto de hub multinível, 364
Propagação
atraso de propagação fim-a-fim de canal, 345
atraso de, 28, 30
comparada com atraso de transmissão, 30-31
multivias, 396
tempo de viagem de ida e volta (*round-trip time* — RTT), 71
Protocolo
ALOHA, 42, 341-344
bit alternante, 165
de Autenticação de Cabeçalho (AH), 558-559
de camada de enlace, 328
de Configuração Dinâmica de Hospedeiros (Dynamic Host Configuration Protocol — DHCP), 266, 353-355
de Controle de Enlace (*link-control protocol* — LCP), 376
de controle de rede (*network-control protocol* — NCP), 40, 376
de Controle RTP (RTCP), 469-471
de gerenciamento de rede , 576
de janela deslizante, 170
de resolução de endereços (*address resolution protocol* — ARP), 350-353
de Roteador de Borda (Border Gateway Protocol — BGP), 297
de Roteamento Multicast de Vetor de Distâncias (Distance-Vector Multicast Routing Protocol — DVMRP), 311
de Roteamento PIM, 312
de Segurança de Encapsulamento de Carga Útil (Encapsulation Security Payload — ESP), 558, 560
de segurança IP (IPsec), 558-559
de transporte confiável, 231-232
DIAMETER, 564
Extensível de Autenticação (Extensible Authentication Protocol — EAP), 563
plug and play, 266
Protocolo da Internet (Internet Protocol — IP), 4
e Protocolo de Mensagem de Controle da Internet (Internet Control Message Protocol — ICMP), 269-270
endereçamento IPv4, 260-268
endereços, 63, 96
falsificação, 548
formatos de datagramas, 255-260
fragmentação do, 258-260
mobilidade de sub-rede, 411-412
móvel, 426-428
SIP, 471-475
sobre ATM, 380-382
tradução de endereços na rede (NAT), 266-268
Protocolo de Controle de Transmissão (Transmission Control Protocol — TCP), 4, 10, 42
camada de transporte, 178-198
e servidores Web, 152-153
encerramento, 196
estados do, 196
futuros, 212-213
gerenciamento de, 195-198
HTTP, 69
justiça, 213-215
programação de aplicações com, 113-122
segmentos, 180, 181-184
serviços, 65-66
Protocolo de Datagrama do Usuário (User Datagram Protocol — UDP), 10, 42
camada de transporte, 153-157
justiça e, 215
programação de sockets com, 122-128
segmentos, 155-156
serviços do, 66-67
somas de verificação, 156-157
Protocolo de Informação de Roteamento (Routing Information Protocol — RIP)
roteamento intra-AS, 291-293
Protocolo de Mensagens de Controle da Internet (Internet Control Message Protocol — ICMP), 269-270
e IP, 269
Protocolo de transferência confiável de dados, 158
com paralelismo, 166-169
construção de um, 159-166
Protocolos, 4
de acesso aleatório, 338, 341-346
de acesso ao correio, 92-95
de camada de aplicação, 63-64
de localização de usuários móveis, 425
de passagem de permissão, 347
de polling (seleção), 346
de revezamento, 338, 346
de roteamento inter-AS, 289
de transferência confiável de dados com paralelismo, 166-169
formatos de datagramas IPv4, 256
camadas de, 34-40
de acesso ao correio, 92-95

plug and play, 266
de sinalização, 243
para aplicações interativas em tempo real, 465-477
pare e espere, 161, 168
pilhas de, 36
sem estado, 70
Internet, 5-7Protocolos ARQ (Automatic Repeat reQuest — ARQ), 160
slotted ALOHA, 341-342
Provedores, 27
de Serviço de Internet (Internet Service Providers — ISPs), 3, 26-28
Proxy, servidores, 78

Q

Quadros, 38-39
camada de enlace, 329
colisões, 338
de sinalização, 403
Ethernets, 356-360
LANs sem fio IEEE 802.11, 408-411
temporais, 339
Quantização, 448
Quarta geração de acesso celular à Internet (4G), 418

R

Rádio por satélite, canais de, 26
RADIUS, 564
Realimentação do destinatário, 160-161
RealNetworks, 444
Recebimento de um ACK, 171
Reconhecimento
cumulativo, 171, 183
seletivo, 193
Reconhecimento (*acknowledgment* — ACK)
campos de número de, 181
estudo de caso, 183-184
na camada de enlace, 405
pacotes, 161-162, 182
recebimento de pacotes, 171
Reconhecimentos
negativos, 160
positivos, 160
Recuperação
de perda de pacotes, 462-465
rápida, 210
Rede
de área pessoal sem fio (*wireless personal area network* — WPAN), 412
local pública terrestre móvel nativa(PLMN nativa), 429
stub com múltiplas conexões, 300
Redes
1980-1990, 43
ad hoc, 396
aplicações de, 65
como camadas de enlace, 377-384
comutação, 11-17
de acesso da Internet, 19-24
de acesso, 22-23
de áreas metropolitana (*metropolitan area networks* — MANs), 348
de computadores, 1
de comutação virtuais, 17-18
de sobreposição, 108
de tempo compartilhado, 42
externas, 420
gerenciamento de, 571
Internet, 1-4
multimídia, 442
multicast de sobreposição, 447
nativas, 420, 429
OSPF, 295
proprietárias (1972-1980), 41-43
segurança, 512
sem fio, 393-394
sistemas finais, 9–11
stub com múltuplas conexões, 300
stub, 300
taxonomia de, 18
virtuais privadas (VPNs), 384
visitadas, 420, 430
Redes de comutação de pacotes, 17-18
atraso em, 28-34
desenvolvimento de (1961-1972), 40-41
perda em, 28-34
Redes de datagramas, 17-18
camada de rede, 244-246
circuitos virtuais (VCs), 246
Redes de distribuição de conteúdo (*content distribution networks* — CDNs), 447, 477-480, 571
Redes de interconexão, comutação por, 251
Redes locais (LANs), 22, 347-348
caches, 80
endereços de, 349
segmentos de, 364
sem fio, 22, 401-412
Redes móveis, 393-394. *Veja também Redes sem fio*
acesso celular à Internet, 413-418
endereçamento, 421
gerenciamento de, 418-425, 429-433
IP, 426-428
roteamento, 422-425
Redes multimídia, 442
aplicações de, 442-449
arquitetura de, 480-485
áudio e vídeo de fluxo contínuo armazenados, 449-457
compressão em, 448-449
escalonamento, 485-489
protocolos para aplicações interativas em tempo real, 465-477
redes de distribuição de conteúdo, 477-480
regulação, 489-490
resolução de problemas, 445-446
RSVP, 497-502
serviços de melhor esforço, 457-465
serviços diferenciados (Diffserv), 493-497
serviços integrados (Intserv), 491-493
suporte da Internet, 446-447
telefone por Internet, 457-465
Redes sem fio, 393-396
enlaces e, 396-401
LANs, 401-412
questões de mobilidade, 434-435
Registro
Americano para Números da Internet (American Registry for Internet Numbers — ARIN), 270
de localização de visitantes (*visitor location register* — VLR), 430
nativo de localização (*home location register* — HLR), 430
Registros
adição de, 105
de recursos (RR), 102
DNS, 102-105
Regra da concordância do prefixo mais longo, 245
Regras Básicas de Codificação (Basic Encoding Rules — BERs), 591
Regulação de redes multimídia, 489-490
Remetentes
e caminhos com múltiplos roteadores, 202-204
GBN, 170
SR, 174
Reparação de áudio no receptor, 464-465
Repasse, 235, 236-239, 247
acelerado, 497
pelo caminho inverso (*reverse path forwarding* — RPF), 303
por comutadores, 366-367
tabelas de, 237
Repetição seletiva, 169, 173-178
Reprodução contínua, 443
Requisições, 112
Resolução de problemas
camada de enlace, 330
correção de erros de repasse (FEC), 334
de DNS, 98
detecção de erro, 332-337
em compartilhamento de arquivos P2P, 107
enlaces, 285-286
GET condicional, 81-82
perda de pacotes, 462-465
problema do roteamento triangular, 424
redes multimídia, 445-446
redes sem fio, 434-435
reparação no receptor, 464-465
Respostas, 83-84
Resumo
de rotas, 264
de mensagem, 532-534
Retransmissões, 161
controle de congestionamento, 201-202
rápidas, 191
reconhecimento perdido, 189
Reversão de conexões, 268
Reversão envenenada
adição de, 286
RFCs (*requests for comments* — pedidos de comentários), 4
Robustez dos algoritmos de roteamento, 287
Rotas, 3
agregação de, 264-265
atraso e, 33-34
comutação de pacotes, 14
resumo de, 264-265
Roteadores, 3, 239, 276
arquitetura de, 247-248
camada de rede, 247-253
Cisco, 248
com buffers finitos, 201
com buffers infinitos, 199-204
da fonte, 276
de borda, 19, 288
de borda de área, 295
de comutação de rótulos, 382
de destino, 276
de primeiro salto, 276
default, 276
placas de linha de, 247
prefixos, 245
sub-redes, 262
versus comutadores, 371-372
Roteamento, 235, 236-239
algoritmos de, 236

da batata quente, 290
de chamadas móveis, 430-431
de comutação de circuitos, algoritmos de, 287
de vetor de distâncias, algoritmo de, 281-287
direto, 424-425
externo a sistemas autônomos: BGP, 297-301
fim-a-fim, 18
global, algoritmos de 277
indireto, 422-424
Interdomínio sem Classes (Classless Interdomain Routing — CIDR), 262-263
hierárquico, 287-290
loops de, 285
multicast, 311-313
multicast, algoritmos de 310-311
na Internet, 290-301
processadores de, 248
redes móveis, 422-425
Roteamento Interdomínio sem Classes (Classless Interdomain Routing — CIDR), 262
tabelas de, 283, 292
Roteamento intra-AS
OSPF, 293-296
RIP, 291-293
RSA, algoritmos, 522
RSVP, 497-502
RTP (*real-time protocol* — protocolo de tempo real), 450, 465-468
RTS, 406-407
RTSP (*real-time streaming protocol* — protocolo de fluxo contínuo em tempo real), 450, 454-457

S

Saída, 116
buffers de, 15
cadeia de, 115
de portas, 247, 252
filas de, 15
padrão, 116
Saltos
compartilhamento de, 199-200
trajetos com múltiplos roteadores, 202
Satélites
de baixa altitude, 26
geoestacionários, 26
Scantlebury, Roger, 40
Schulzrinne, Henning, 509-511
Segmentos, 37
de LAN, 364
números de seqüência para, 182
SYNACK, 195
tamanho máximo de, 179
TCP, 180, 181-184
UDP, 155-156
Segunda Geração de Acesso Celular à Internet (2G), 416
Segurança
acordo de (AS), 558
ataques, 546-549
autenticação, 525-530
baseada no usuário, 587
certificação, 536-541
criptografia de chaves públicas, 521-525
criptografia, 515-525
distribuição de chaves, 536-541
e administração, 586-589
em e-mail, 550-554
em IEEE 802.11, 560-564
firewalls, 541-546
integridade, 530-536
na camada de transporte (TLS), 555-558
nas camadas, 550-564
visão geral de, 513-515
Seqüestro, 549
Serviço(s)
camada de enlace, 328-331
camada de rede, 239-241
camada de transporte, 143-147
camadas de protocolo, 34-39
confiável de corrente de bytes, 115
confiável de transporte, 66
confiável orientado para conexão, 4
da Internet, 4-5
de apresentação, 591
de controle de congestionamento, 10
de controle de fluxo, 193
de melhor esforço de redes multimídia, 445, 457-465
de protocolo de transporte, 65-67
de qualidade garantida, 493
de rede de carga controlada, 493
de transferência confiável de dados, 187-193
diferenciados (Diffserv), 447, 493-497
do UDP, 66-67
dos protocolos de transporte, 65-67
full-duplex, 179
integrados (Intserv), 491-493
ISPs (provedores de serviço de Internet), 26-28
não confiáveis, 146
não orientados para conexão, 8-10, 149, 359-360
não orientados para conexão, não confiáveis, 359-360
não confiável não orientado para conexão, 4, 359-360
orientados para conexão, 8-10, 66
para aplicações, 64-65
sempre disponíveis, 22
serviços integrados (Intserv), 491-493
sistema de nome de domínio (*domain name system* — DNS), 95-98
TCP, 66
tipo de serviço (TOS), 256
Serviços diferenciados (Diffserv), 447
redes multimídia, 493-497
Servidores de correio, 85
apelidos, 97
Servidores, 7-8, 59
aplicações em Java, 116-122
construindo servidores Web, 128-132
de chaves públicas PGP, 554
de correio, 85
de domínio de alto nível (*top-level domain* — TLD), 99
de fluxo contínuo, 452-454
de nomes raiz, 99
DNS, 95-105
DNS com autoridade, 100
DNS locais, 100
interação usuário-servidor (cookies), 76-78
Minitel, 43
Ping, 140
processos, 61-62
proxy, 78
Web, 69
Servidores Web, 69
acesso a áudio e vídeo por meio de, 450-452
construindo, 128-132
e TCP, 152-153
transitórios, 106
Sessão única, chaves de, 538
Sessões
BGP, 298
chaves de, 523
chaves de sessão única, 338
RSVP, 498
RTP, 466
Sever farm, 59b
Shannon, Claude, 55
Sinalização, 243
quadros de, 403
SIP, 471-475
Sistemas
autônomos (*autonomous systems* — ASs), 288
autônomos interconectados, 288-289
de chaves simétricas, 516, 517-520
Sistemas finais, 3, 7-8
apelidos, 97
comutação, 12-13
endereços, 63, 266
multicast, 310
Protocolo de Configuração Dinâmica de Hospedeiros (Dynamic Host Configuration Protocol — DHCP), 266, 353-355
sem fio, 393-394
SMTP (Simple Mail Transfer Protocol), 64, 85, 86-89
SNMP (Simple Network Management Protocol — Protocolo Simples de Gerenciamento de Rede), 577, 578
e mapeamento, 584-586
segurança, 586-589
Sockets, 62
de conexão, 115
multiplexação, 148
programação de, 138-140
programação com TCP, 113-122
programação com UDP, 122-128
Soft, estado, 502
Solicitação de agente, 427
Somas de verificação
campos de, 181
da Internet, 334-335
de cabeçalhos, 257
UDP, 156-157
Spanning tree mínima, 304
Stanford Research Institute (SRI), 40
Sub-redes, 260-261
endereçamento de, 262-263
envio de datagramas, 352-353
máscaras, 261
mobilidade em, 411-412
roteadores, 262-263
SYNACK, segmentos, 195

T

Tabelas
ARP, 351
de comutação, 366
de repasse, 237
de roteamento, 283, 292
tradução de endereço de rede (*network address translation* — NAT), 267

Tamanho
de janela, 170
máximo de segmento (*maximum segment size* — MSS), 179
Taxa
constante de bits (*constant bit rate* — CBR), 240
de bits disponível (*available bit rate* — ABR), 199-241
do elemento de comutação, 252
mínima de transmissão de células (*minimum cell transmission rate* — MCR), 241
Taxas de transmissão, 3
Taxonomia de redes, 18
TCP Reno, 210
TCP Tahoe, 210
TCP/IP, 4, 179, 180
TCPs futuros, 212-213
Tecnologia
embutida, 56
Ethernet, 362-364
Telefone por Internet, 444, 457-465
Telenet, 42
Telnet, 183-184
Tempo
de envelhecimento, 369
de expiração, 191
de vida (*time-to-live* — TTL), 256-257
Tempo de viagem de ida e volta (*round-trip time* — RTT), 71
esgotamento de temporização, 184-187
estimativa de, 184-187
Temporais, quadros, 339
Temporização na camada de aplicação, 65
Temporizadores de contagem regressiva, 165
Terceira Geração (3G)
acesso celular à Internet, 416-417
do acesso sem fio, 23
Testes de aceitação, 501
Texto
aberto, 516
cifrado, 516
claro, 516
Tipo
de carga útil (PT), 379
de serviço (*type of service* — TOS), 256
Token rings, 347
Tolerantes a perda, aplicações, 64
Tomlinson, Ray, 41
Torres celulares, 395
Trabalho em rede (1972-1980), 41-43
Tradução
de endereços de rede (NAT), 266-268
identificador de circuito virtual (ID CV), 17
Tradução de endereços na Internet (NAT), 266-268
IP, 266-268
Tráfego
engenharia de, 384
intensidade de, 31
perfis de, 495
volume de, 98
Trajetos com múltiplos roteadores, 202-204
Transdutores (mídia players), 450
Transferência
bidirecional de dados, 159
de arquivos, 109
de dados, 243
paralela, 112
unidirecional de dados, 159
Transferência confiável de dados, 10, 64, 147
camada de transporte, 157-178
canais, 159-166
por canais com perda, 164-166
Transferências (Handoffs), 396, 431-433
Transmissões
armazena e reenvia, 14
full-duplex, 330
half-duplex, 330
totais, 14
Troca de Chaves da Internet (Internet Key Exchange — IKE), 560
Túneis, 274
Tymnet, 42

U

Unidade
de dados de protocolo (PDU AAL), 379
máxima de transmissão (*maximum transmission unit* — MTU), 179, 258
Universal Mobile Telecommunications (Service Serviço Universal de Telecomunicação Móvel — UMTS), 417
Utilização, 168

V

Variação de atraso de pacote, 459
Varredura de portas 546
Vazão
de uma conexão TCP, 212-214
por conexão, 200
taxas de envio de hospedeiros, 200
Velocidade
da linha, 249
de algoritmos de roteamento, 287
de chipping, 398
de convergência, 287
Verificação de redundância cíclica (CRC), 335-337, 358
Verificações de paridade, 333-334
Vídeo
acesso a, 450-452
compressão de, 448-449
de fluxo contínuo ao vivo, 444
de fluxo contínuo armazenado, 443, 449-457
DVD, 449
e áudio de fluxo contínuo ao vivo, 444
H.323, 476-477
interativo em tempo real, 444-445
RTSP (protocolo de fluxo contínuo em tempo real), 454-457
Violação de direitos autorais, 108
Volume de tráfego, 98

W

Web, aplicações, 63-64
World Wide Web (WWW), 44
e o HTTP, 68-82

CADASTRO PARA MALA DIRETA

Favor preencher todos os campos

★ Devolvendo-nos este cadastro preenchido, você passará a receber informações dos nossos lançamentos, nas áreas que determinar. **INVISTA EM SEU FUTURO PROFISSIONAL.**

Bairro: **Cidade:** **UF:** **Cep:** –

Telefone: **Celular:** **E-mail:** **Sexo:** F M

1. Escolaridade:

- ☐ 1° Grau
- ☐ 2° Grau
- ☐ 3° Grau
- ☐ Pós-Graduação
- ☐ MBA
- ☐ Mestrado
- ☐ Doutorado
- ☐ Outros (especificar): _____

Quantos livros técnicos compra por mês?: _____ por ano? _____

2. Área de Interesse:

- ☐ 1. Informática
- ☐ 2. Marketing
- ☐ 3. Vendas
- ☐ 4. Administração
- ☐ 5. Economia
- ☐ 6. Recursos Humanos
- ☐ 7. Qualidade/Produtividade
- ☐ 8. Psicologia
- ☐ 9. Eng. Elétrica/Eletrônica
- ☐ 10. Engenharia Civil
- ☐ 11. Engenharia Mecânica
- ☐ 12. Comércio Exterior
- ☐ 13. Engenharia Química
- ☐ 14. Ecologia
- ☐ 15. Telecomunicações
- ☐ 16. Publicidade/Propaganda
- ☐ 17. Turismo
- ☐ 18. Ensino/Educação
- ☐ 19. Contabilidade
- ☐ 20. Finanças
- ☐ 21. Matemática
- ☐ 22. Outros (especificar): _____

3. Profissão/Ocupação:

- ☐ 1. Presidente
- ☐ 2. Supervisor
- ☐ 3. Diretor
- ☐ 4. Gerente
- ☐ 5. Analista
- ☐ 6. Programador
- ☐ 7. Empresário
- ☐ 8. Consultor
- ☐ 9. Digitador
- ☐ 10. Estudante
- ☐ 11. Professor
- ☐ 12. Aposentado
- ☐ 13. Outros (especificar): _____

Obra: Rede de computadores e a Internet: uma abordagem top-down

Autoria: James F. Kurose / Keith W. Ross

PEARSON Addison Wesley
Av. Ermano Marchetti, 1435
05038-001 - São Paulo - SP - Brasil
Fone: (11) 3613-1222
e-mail: vendas@pearsoned.com

DOBRE AQUI E COLE

ISR - 40 - 1248/89
UP - AC - ITAIM BIBI
DR/São Paulo

CARTA-RESPOSTA

NÃO É NECESSÁRIO SELAR.

O selo será pago por

Pearson Education do Brasil Ltda.

04533-970 - São Paulo - SP

DOBRE AQUI

PEARSON EDUCATION
Prentice Hall / Addison Wesley / Makron Books
Promoção Universitária
CORTESIA
(Venda Proibida)
universitarios@pearsoned.com
(011) 3613-1212