

Computer and Communication Networks

NADER F. MIR

COMPUTER AND COMMUNICATION NETWORKS

This page intentionally left blank

COMPUTER AND COMMUNICATION NETWORKS

Nader F. Mir



PRENTICE
HALL

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco • New York
Toronto • Montreal • London • Munich • Paris • Madrid • Capetown
Sydney • Tokyo • Singapore • Mexico City

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The author and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

The publisher offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact:

U.S. Corporate and Government Sales
(800) 382-3419
corpsales@pearsontechgroup.com

For sales outside the United States, please contact:

International Sales
international@pearsoned.com



This Book Is Safari Enabled

The Safari® Enabled icon on the cover of your favorite technology book means the book is available through Safari Bookshelf. When you buy this book, you get free access to the online edition for 45 days.

Safari Bookshelf is an electronic reference library that lets you easily search thousands of technical books, find code samples, download chapters, and access technical information whenever and wherever you need it.

To gain 45-day Safari Enabled access to this book:

- Go to <http://www.prenhallprofessional.com/safarienabled>
- Complete the brief registration form
- Enter the coupon code VZGY-SBEH-4Q1D-K8IN-BX58

If you have difficulty registering on Safari Bookshelf or accessing the online edition, please e-mail customer-service@safaribooksonline.com.

Visit us on the Web: www.prenhallprofessional.com

Library of Congress Cataloging-in-Publication Data

Mir, Nader F.

Computer and communication networks / Nader F. Mir.

p. cm.

Includes bibliographical references and index.

ISBN 0-13-174799-1 (hardcover : alk. paper)

1. Computer networks. 2. Data transmission systems. I. Title.

TK5105.5M567 2006

004.6—dc22

2006025914

Copyright © 2007 Pearson Education, Inc.

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. For information regarding permissions, write to:

Pearson Education, Inc.
Rights and Contracts Department
One Lake Street
Upper Saddle River, NJ 07458
Fax: (201) 236-3290

ISBN 0-13-174799-1

Text printed in the United States on recycled paper at R.R. Donnelley in Crawfordsville, Indiana.

First printing, November 2006

To Sherry and Navid

This page intentionally left blank

Contents

Preface *xxi*

About the Author *xxxi*

PART I: Fundamental Concepts 1

1 Packet-Switched Networks 3

- 1.1 Basic Definitions in Data Networks 4
 - 1.1.1 Packet Switching versus Circuit Switching 4
 - 1.1.2 Messages, Packets, and Frames 5
 - 1.1.3 The Internet 6
 - 1.1.4 ISPs and Internetwork Components 9
- 1.2 Types of Packet-Switched Networks 10
 - 1.2.1 Connectionless Networks 11
 - 1.2.2 Connection-Oriented Networks 13
- 1.3 Packet Size and Optimizations 14
- 1.4 Summary 16
- 1.5 Exercises 17

2 Foundation of Networking Protocols 19

- 2.1 5-Layer TCP/IP Model 20
- 2.2 7-Layer OSI Model 22
- 2.3 Internet Protocols and Addressing 23
 - 2.3.1 IP Packet 23
 - 2.3.2 IP Addressing Scheme 24
 - 2.3.3 Subnet Addressing and Masking 25

2.3.4	Classless Interdomain Routing (CIDR)	27
2.3.5	Packet Fragmentation and Reassembly	28
2.3.6	Internet Control Message Protocol (ICMP)	29
2.3.7	IP Version 6 (IPv6)	30
2.4	Equal-Sized Packets Model: ATM	33
2.4.1	ATM Protocol Structure	34
2.4.2	ATM Cell Structure	36
2.5	Summary	39
2.6	Exercises	39

3 Networking Devices 43

3.1	Multiplexers	43
3.1.1	Frequency-Division Multiplexing (FDM)	44
3.1.2	Wavelength-Division Multiplexing (WDM)	44
3.1.3	Time-Division Multiplexing	45
3.2	Modems and Internet Access Devices	50
3.2.1	Line Coding Methods	50
3.2.2	Digital Modulation Techniques	52
3.2.3	Digital Subscriber Line (DSL) Modems	54
3.2.4	Cable Modems	56
3.3	Switching and Routing Devices	57
3.3.1	Repeaters, Hubs, and Bridges	57
3.3.2	Routers and Higher-Layer Switches	59
3.4	Router Structure	60
3.4.1	Input Port Processor (IPP)	60
3.4.2	Switch Fabric	63
3.4.3	Switch Controller	64
3.4.4	Output Port Processors (OPP)	65
3.5	Summary	67
3.6	Exercises	67

4 Data Links and Transmission 71

4.1	Data Links	72
4.2	Wired Links and Transmission	73
4.2.1	Twisted-Pair Links	73
4.2.2	Coaxial Cable	73
4.2.3	Optical Fiber	73
4.3	Wireless Links and Transmission	74
4.3.1	Choice of Antenna	75

4.3.2	Wireless Channels	76
4.3.3	Capacity Limits of Wireless Channels	79
4.3.4	Channel Coding	79
4.3.5	Flat-Fading Countermeasures	79
4.3.6	Intersymbol Interference Countermeasures	80
4.3.7	Orthogonal Frequency Division Multiplexing (OFDM)	81
4.4	Methods of Channel Access on Links	82
4.4.1	Frequency-Division Multiple Access	83
4.4.2	Time-Division Multiple Access	83
4.4.3	Code-Division Multiple Access	85
4.4.4	Space-Division Multiple Access	87
4.4.5	Hybrid Multiple-Access Techniques	87
4.5	Error Detection and Correction	87
4.5.1	Error Detection Methods	89
4.5.2	Cyclic Redundancy Check (CRC) Algorithm	89
4.6	Link-Level Flow Control	94
4.6.1	Stop-and-Wait Flow Control	95
4.6.2	Sliding-Window Flow Control	96
4.7	Summary	98
4.8	Exercises	99

5 Local Area Networks and Networks of LANs 101

5.1	LANs and Basic Topologies	102
5.2	LAN Protocols	103
5.2.1	Logical-Link Layer (LLC)	103
5.2.2	Medium Access Control (MAC)	104
5.3	MAC and IP Addresses	104
5.3.1	Address Resolution Protocol (ARP)	105
5.3.2	Reverse Address Resolution Protocol (RARP)	106
5.4	Classification of MAC Protocols	106
5.5	Contention-Access MAC	107
5.5.1	Carrier Sense Multiple Access (CSMA)	108
5.5.2	Ethernet LAN: IEEE 802.3 Standard	113
5.6	Round-Robin-Access MAC	114
5.6.1	Token-Ring Access Protocol	114
5.6.2	Token-Ring: IEEE 802.5 Standard	116
5.7	Network of LANs	116
5.7.1	Using Repeaters, Hubs, and Bridges	117
5.7.2	Layers 2 and 3 Switches	124

5.8	Summary	125
5.9	Exercises	126
6	Wireless Networks and Mobile IP	129
6.1	Infrastructure of Wireless Networks	130
6.2	Wireless LAN Technologies	131
6.2.1	Infrared LANs	132
6.2.2	Spread-Spectrum LANs	133
6.2.3	Narrowband RF LANs	133
6.2.4	Home RF and Bluetooth	134
6.3	IEEE 802.11 Wireless Standard	134
6.3.1	802.11 Physical Layer	136
6.3.2	802.11 MAC Layer	137
6.3.3	WiFi Technology and 802.11	141
6.4	Cellular Networks	142
6.4.1	Connectivity	143
6.4.2	Frequency Reuse	146
6.4.3	Local and Regional Handoffs	149
6.4.4	Mobility Management	150
6.4.5	Generations of Cellular Systems	154
6.4.6	CDMA-Based Mobile Wireless	154
6.5	Mobile IP	155
6.5.1	Addresses and Agents	156
6.5.2	Agent Discovery Phase	157
6.5.3	Registration	158
6.5.4	Mobile IP Routing	159
6.5.5	Security	163
6.6	Wireless Mesh Networks (WMNs)	163
6.6.1	WiMAX Technology and IEEE 802.16	163
6.6.2	Applications of Mesh Networks	164
6.6.3	Physical and MAC Layers of WMNs	167
6.7	Summary	168
6.8	Exercises	168
7	Routing and Internetworking	171
7.1	Network-Layer Routing	172
7.1.1	Assigning Addresses to Hosts and Routers, and DHCP	173
7.1.2	Network Address Translation (NAT)	174
7.1.3	Route Cost	176
7.1.4	Classification of Routing Algorithms	176

7.2	Least-Cost-Path Algorithms	177
7.2.1	Dijkstra's Algorithm	177
7.2.2	Bellman-Ford Algorithm	178
7.3	Non-Least-Cost-Path Routing	180
7.3.1	Flood Routing	180
7.3.2	Deflection Routing	181
7.4	Intradomain Routing Protocols	182
7.4.1	Routing Information Protocol (RIP)	183
7.4.2	Open Shortest Path First (OSPF)	187
7.5	Interdomain Routing Protocols	190
7.5.1	Border Gateway Protocol (BGP)	190
7.6	Congestion Control at Network Layer	194
7.6.1	Unidirectional Congestion Control	196
7.6.2	Bidirectional Congestion Control	197
7.6.3	Random Early Detection (RED)	198
7.6.4	A Quick Estimation of Link Blocking	200
7.7	Summary	202
7.8	Exercises	203

8 Transport and End-to-End Protocols 207

8.1	Transport Layer	208
8.1.1	Interaction of Transport and Network Layers	209
8.2	Transmission Control Protocol (TCP)	209
8.2.1	TCP Segment	210
8.2.2	Connection Setup	212
8.3	User Datagram Protocol (UDP)	213
8.3.1	UDP Segment	213
8.3.2	Applications of TCP and UDP	214
8.4	Mobile Transport Protocols	215
8.4.1	TCP for Mobility	215
8.4.2	UDP for Mobility	216
8.5	TCP Congestion Control	217
8.5.1	Additive Increase, Multiplicative Decrease Control	217
8.5.2	Slow Start Method	219
8.5.3	Fast Retransmit Method	220
8.5.4	TCP Congestion Avoidance Methods	221
8.6	Summary	222
8.7	Exercises	223

9 Applications and Network Management	225
9.1 Application-Layer Overview	226
9.1.1 Client and Server Model	226
9.2 Domain Name System (DNS)	227
9.2.1 Domain Name Space	228
9.2.2 Name/Address Mapping	230
9.2.3 DNS Message Format	231
9.3 Remote Login Protocols	232
9.3.1 TELNET Protocol	233
9.3.2 Secure Shell (SSH) Protocol	234
9.4 Electronic Mail (E-mail)	235
9.4.1 Simple Mail Transfer Protocol (SMTP) and E-mail	235
9.5 File Transfer and FTP	237
9.5.1 File Transfer Protocol (FTP)	237
9.5.2 Secure Copy Protocol (SCP)	237
9.6 World Wide Web (WWW) and HTTP	237
9.6.1 Web Caching (Proxy Server)	238
9.7 Network Management	239
9.7.1 Elements of Network Management	241
9.7.2 Structure of Management Information (SMI)	241
9.7.3 Management Information Base (MIB)	242
9.7.4 Simple Network Management Protocol (SNMP)	243
9.8 Summary	245
9.9 Exercises	246
10 Network Security	249
10.1 Overview of Network Security	250
10.1.1 Elements of Network Security	250
10.1.2 Threats to Network Security	251
10.2 Overview of Security Methods	255
10.2.1 Cryptographic Techniques	255
10.2.2 Authentication Techniques	256
10.3 Secret-Key Encryption Protocols	257
10.3.1 Data Encryption Standard (DES)	257
10.3.2 Advanced Encryption Standard (AES)	259
10.4 Public-Key Encryption Protocols	260
10.4.1 RSA Algorithm	260
10.4.2 Diffie-Hellman Key-Exchange Protocol	262

10.5	Authentication	263
10.5.1	Secure Hash Algorithm (SHA)	264
10.6	Authentication and Digital Signature	265
10.7	Security of IP and Wireless Networks	266
10.7.1	IP Security and IPsec	266
10.7.2	Security of Wireless Networks and IEEE 802.11	267
10.8	Firewalls	269
10.9	Summary	270
10.10	Exercises	271

PART II: Advanced Concepts 273

11 Packet Queues and Delay Analysis 275

11.1	Little's Theorem	276
11.2	Birth-and-Death Process	278
11.3	Queueing Disciplines	279
11.4	Markovian FIFO Queueing Systems	281
11.4.1	$M/M/I$ Queueing Systems	281
11.4.2	Systems with Limited Queueing Space: $M/M/I/b$	286
11.4.3	$M/M/a$ Queueing Systems	287
11.4.4	Models for Delay-Sensitive Traffic: $M/M/a/a$	292
11.4.5	$M/M/\infty$ Queueing Systems	293
11.5	Non-Markovian and Self-Similar Models	295
11.5.1	Pollaczek-Khinchin Formula and $M/G/1$	295
11.5.2	$M/D/I$ Models	298
11.5.3	Self-Similarity and Batch-Arrival Models	298
11.6	Networks of Queues	299
11.6.1	Burke's Theorem	299
11.6.2	Jackson's Theorem	304
11.7	Summary	308
11.8	Exercises	309

12 Quality of Service and Resource Allocation 315

12.1	Overview of QoS	316
12.2	Integrated Services QoS	316
12.2.1	Traffic Shaping	317
12.2.2	Admission Control	324
12.2.3	Resource Reservation Protocol (RSVP)	324

12.2.4	Packet Scheduling	325
12.3	Differentiated Services QoS	335
12.3.1	Per-Hop Behavior (PHB)	336
12.4	Resource Allocation	337
12.4.1	Management of Resources	338
12.4.2	Classification of Resource-Allocation Schemes	338
12.4.3	Fairness in Resource Allocation	340
12.4.4	ATM Resource Allocation	340
12.4.5	Cell Scheduling and QoS	343
12.5	Summary	344
12.6	Exercises	344

13 Networks in Switch Fabrics 349

13.1	Characteristics and Features of Switch Fabrics	350
13.1.1	Blocking and Nonblocking Networks	350
13.1.2	Features of Switch Fabrics	351
13.1.3	Complexity of Switching Networks	351
13.1.4	Definitions and Symbols	351
13.2	Crossbar Switch Fabrics	352
13.3	Blocking Switch Fabrics	353
13.3.1	Omega Network	353
13.3.2	Banyan Network	355
13.3.3	Delta Networks	355
13.3.4	Beneš Networks	356
13.4	Nonblocking Switch Fabrics: Clos Networks	357
13.4.1	Estimation of Blocking Probabilities	360
13.4.2	Five-Stage Clos Networks	361
13.5	Concentration and Expansion Switches	361
13.5.1	Knockout Switching Network	363
13.5.2	Expansion Network	364
13.6	Shared-Memory Switch Fabrics	365
13.7	Techniques for Improving Performance	366
13.7.1	Parallel-Plane Switching Networks	367
13.8	Case Study: Multipath Buffered Crossbar	368
13.8.1	Queueing Model	370
13.8.2	Markov Chain Model	371
13.8.3	Throughput and Delay	374

13.9 Summary 375

13.10 Exercises 376

14 Optical Networks and WDM Systems 379

14.1 Overview of Optical Networks 380

 14.1.1 Protocol Models and Standards 380

14.2 Basic Optical Networking Devices 382

 14.2.1 Tunable Lasers 382

 14.2.2 Optical Buffers or Delay Elements 382

 14.2.3 Optical Amplifiers 382

 14.2.4 Optical Filters 382

 14.2.5 Wavelength-Division Multiplexer (WDM) 383

 14.2.6 Optical Switches 384

14.3 Large-Scale Optical Switches 386

 14.3.1 Crossbar Switching Network 387

 14.3.2 Spanke-Beneš Switching Network 387

14.4 Optical Routers 388

 14.4.1 Structure of Wavelength Routing Nodes 389

14.5 Wavelength Allocation in Networks 391

 14.5.1 Classification of Optical Networks 392

 14.5.2 Wavelength Allocation 393

14.6 Case Study: An All-Optical Switch 395

 14.6.1 Self-Routing in SSN 397

 14.6.2 Transmission in SSN 397

14.7 Summary 398

14.8 Exercises 399

15 Multicasting Techniques and Protocols 401

15.1 Basic Definitions and Techniques 402

 15.1.1 IP Multicast Address 403

 15.1.2 Basic Multicast Tree Algorithms 404

 15.1.3 Classification of Multicast Protocols 406

15.2 Intradomain Multicast Protocols 406

 15.2.1 Distance Vector Multicast Routing Protocol (DVMRP) 407

 15.2.2 Internet Group Management Protocol (IGMP) 407

 15.2.3 Multicast OSPF (MOSPF) Protocol 409

 15.2.4 Protocol-Independent Multicast (PIM) 410

 15.2.5 Core-Based Trees (CBT) Protocol 413

15.2.6	Multicast Backbone (MBone)	413
15.3	Interdomain Multicast Protocols	414
15.3.1	Multiprotocol BGP (MBGP)	414
15.3.2	Multicast Source Discovery Protocol (MSDP)	415
15.3.3	Border Gateway Multicast Protocol (BGMP)	417
15.4	Node-Level Multicast Algorithms	417
15.4.1	Tree-Based Multicast Algorithm	418
15.4.2	Boolean Splitting Multicast Algorithm	420
15.4.3	Packet Recirculation Multicast Algorithm	423
15.4.4	Multicasting in Three-Dimensional Switches	424
15.5	Summary	426
15.6	Exercises	427

16 VPNs, Tunneling, and Overlay Networks 431

16.1	Virtual Private Networks (VPNs)	432
16.1.1	Remote-Access VPN	433
16.1.2	Site-to-Site VPN	433
16.1.3	Tunneling and Point-to-Point Protocol (PPP)	434
16.1.4	Security in VPNs	436
16.2	Multiprotocol Label Switching (MPLS)	437
16.2.1	MPLS Operation	438
16.2.2	Routing in MPLS Domains	439
16.2.3	Tunneling and Use of FEC	441
16.2.4	Traffic Engineering	442
16.2.5	MPLS-Based VPNs	443
16.3	Overlay Networks	444
16.3.1	Peer-to-Peer (P2P) Connection	445
16.4	Summary	446
16.5	Exercises	447

17 Compression of Digital Voice and Video 449

17.1	Overview of Data Compression	450
17.2	Digital Voice and Compression	451
17.2.1	Signal Sampling	451
17.2.2	Quantization and Distortion	452
17.3	Still Images and JPEG Compression	455
17.3.1	Raw-Image Sampling and DCT	456
17.3.2	Quantization	459
17.3.3	Encoding	460

17.4	Moving Images and MPEG Compression	461
17.4.1	MP3 and Streaming Audio	462
17.5	Limits of Compression with Loss	463
17.5.1	Basics of Information Theory	463
17.5.2	Entropy of Information	464
17.5.3	Shannon's Coding Theorem	465
17.5.4	Compression Ratio and Code Efficiency	467
17.6	Compression Methods Without Loss	467
17.6.1	Run-Length Encoding	468
17.6.2	Huffman Encoding	468
17.6.3	Lempel-Ziv Encoding	469
17.7	Case Study: FAX Compression for Transmission	470
17.8	Summary	472
17.9	Exercises	472
18	VoIP and Multimedia Networking	479
18.1	Overview of IP Telephony	480
18.1.1	VoIP Quality-of-Service	481
18.2	VoIP Signaling Protocols	482
18.2.1	Session Initiation Protocol (SIP)	483
18.2.2	H.323 Protocols	486
18.3	Real-Time Media Transport Protocols	490
18.3.1	Real-Time Transport Protocol (RTP)	491
18.3.2	Real-Time Control Protocol (RTCP)	493
18.3.3	Estimation of Jitter in Real-Time Traffic	496
18.4	Distributed Multimedia Networking	497
18.4.1	Content Distribution Networks (CDNs)	498
18.4.2	CDN Interactions with DNS	498
18.4.3	Providing QoS to Streaming	499
18.5	Stream Control Transmission Protocol (SCTP)	500
18.5.1	SCTP Packet Structure	501
18.6	Self-Similarity and Non-Markovian Streaming Analysis	503
18.6.1	Self-Similarity with Batch Arrival Models	503
18.7	Summary	506
18.8	Exercises	507
19	Mobile Ad-Hoc Networks	511
19.1	Overview of Wireless Ad-Hoc Networks	512

19.2	Routing in Ad-Hoc Networks	513
19.2.1	Classification of Routing Protocols	514
19.3	Routing Protocols for Ad-Hoc Networks	515
19.3.1	Destination-Sequenced Distance Vector (DSDV) Protocol	515
19.3.2	Cluster-Head Gateway Switch Routing Protocol	517
19.3.3	Wireless Routing Protocol (WRP)	517
19.3.4	Dynamic Source Routing (DSR) Protocol	519
19.3.5	Temporally Ordered Routing Algorithm (TORA)	520
19.3.6	Associative-Based Routing (ABR) Protocol	521
19.3.7	Ad-Hoc On-Demand Distance Vector (AODV) Protocol	522
19.4	Security of Ad-Hoc Networks	528
19.4.1	Types of Attacks	529
19.4.2	Criteria for a Secure Routing Protocol	530
19.5	Summary	531
19.6	Exercises	531

20 Wireless Sensor Networks 535

20.1	Sensor Networks and Protocol Structures	536
20.1.1	Clustering in Sensor Networks	536
20.1.2	Protocol Stack	537
20.1.3	Sensor Node Structure	538
20.2	Communication Energy Model	540
20.3	Clustering Protocols	545
20.3.1	Classification of Clustering Protocols	546
20.3.2	LEACH Clustering Protocol	546
20.3.3	DEEP Clustering Protocol	547
20.3.4	Reclustering	551
20.4	Routing Protocols	551
20.4.1	Intracluster Routing Protocols	552
20.4.2	Intercluster Routing Protocols	554
20.5	Case Study: Simulation of a Sensor Network	557
20.5.1	Cluster-Head Constellation and Distribution of Load	557
20.5.2	Optimum Percentage of Cluster Heads	558
20.6	Other Related Technologies	559
20.6.1	Zigbee Technology and IEEE 802.15.4	559
20.7	Summary	560
20.8	Exercises	561

Appendix A: Glossary of Acronyms	563
Appendix B: RFCs	569
Appendix C: Probabilities and Stochastic Processes	573
C.1 Probability Theory	573
C.1.1 Bernulli and Binomial Sequential Laws	574
C.1.2 Counting and Sampling Methods	574
C.2 Random Variables	574
C.2.1 Basic Functions	575
C.2.2 Conditional Functions	575
C.2.3 Popular Random Variables	576
C.2.4 Expected Value and Variance	577
C.2.5 A Function of Random Variable	578
C.3 Multiple Random Variables	578
C.3.1 Basic Functions of Two Random Variables	578
C.3.2 Two Independent Random Variables	579
C.4 Stochastic (Random) Processes	579
C.4.1 IID Random Process	580
C.4.2 Brownian Motion Random Process	580
C.5 Theory of Markov Chains	580
C.5.1 Continuous-Time Markov Chains	581
Index	583

This page intentionally left blank

Preface

This textbook represents more than a decade of work. During this time, some material became obsolete and had to be deleted. In my days as a telecommunication engineer and a university professor, much has changed in the fields of data communications and computer networks. Nonetheless, this text covers both the foundations and the latest advanced topics of computer networking.

The Internet is a revolutionary communication vehicle by which we all conveniently communicate every day and do business with one another. Because of its complexities at both hardware and software levels, the Internet is a challenge to those who want to study this field. The growing number and variety of communication services offer obvious challenges for computer network experts in designing cost-effective networks to meet the requirements of emerging communication systems. This book fills the gaps in current available texts.

Objectives

This textbook offers a mix of theory, architecture, and applications. The lack of computer communications books presenting moderate analysis with detailed drawing figures covering both wireline and wireless communication technologies led me to write this book. The main objective of this book is to help readers learn the fundamentals and certain advance concepts of computer and communication networks, using a unified set of symbols throughout a single textbook. The preparation of this book responds to the explosive demand for learning computer communication science and engineering.

This book targets two groups of people. For people in academia, at both the undergraduate and graduate levels, the book provides a thorough design and performance

evaluation of communication networks. The book can also give researchers the ability to analyze and simulate complex communication networks. For engineers who want to work in the communication and networking industry and need a reference covering various angles of computer networks, this book provides a variety of learning techniques: exercises, case studies, and computer simulation projects. The book makes it easy and fun for an engineer to review and learn from a reliable networking reference covering all the necessary concepts and performance models.

Organization of This Book

It would be impossible to cover all networking subjects in one textbook. The range of topics presented in this text, however, allows instructors to choose the topics best suited for their classes. Besides the explanations provided for each chapter, readers will learn how to model a communication network and how to mathematically analyze them. Readers of this text will benefit from the combination of theory and applications presented in each chapter, with the more theoretical portions of each chapter challenging those readers who are more ambitious. This book is organized into 20 chapters in two main parts as follows:

The ten chapters of Part I cover the fundamental topics in computer networking, with each chapter serving as a base for the following chapter. Part I of the book begins with an overview of networking, focusing on TCP/IP schemes, describing wireless networking, and ending with a discussion of the World Wide Web (WWW) and network security. Part I is most appropriate for readers with no experience in computer communications. The ten chapters in Part II cover detailed analytical aspects and a closer perspective of advanced networking protocols: switches, routers, multiplexers, delay and congestion analysis, multimedia networking, multicasting, data compression, voice over IP, optical networks, and sensor networks.

Chapter 1, Packet-Switched Networks, introduces computer networks, touching on the need for networks, explaining relevant packet-switched networks, and giving an overview of today's Internet. Fundamental concepts, such as *messages*, *packets*, and *frames* and *packet switching* versus *circuit switching*, are defined. Various types of packet-switched networks are defined, and how a message can be handled by either *connection-oriented networks* or *connectionless networks* is explained. Finally, this chapter presents a detailed analysis of packet size and optimizations.

Chapter 2, Foundation of Networking Protocols, presents the basics of the five-layer Internet Protocol reference model, as well as other protocols: the seven-layer OSI model and the *equal-size packet protocol* model.

Chapter 3, Networking Devices, introduces the overall architectures of networking devices, such as multiplexers, modems, and switching devices. *Multiplexers* are used in all layers of network. Networking modems are used for access to the Internet from remote and residential areas. Finally, switching devices, such as hubs, bridges, switches, and routers, are used to switch packets from one path to another.

Chapter 4, Data Links and Transmission, focuses on the links and transmission interfaces, the two basic components that networking starts with. This chapter presents both wired and wireless links and describes their characteristics, advantages, and channel access methods. This chapter also presents various *error-detection and correction* techniques at the link level and discusses the integrity of transmitted data. The chapter ends by presenting link-layer *stop-and-wait* and *sliding-window* flow control.

Chapter 5, Local Area Networks and Networks of LANs, explores the implementation of small networks, using the functional aspects of the fundamental knowledge gained in Chapters 2, 3, and Chapter 4 on basic protocols, devices, and links, respectively. The chapter provides some pointers for constructing a network with those devices and making connections, gives several examples of local area networks (LANs), and explains how such LANs are internetworked.

Chapter 6, Wireless Networks and Mobile IP, presents the basics of wireless networking. The chapter discusses challenges in designing a wireless network: *management of mobility*, *network reliability*, and *frequency reuse*. Next, the chapter presents an overview of wireless communication systems at all levels, from satellite to local-area networks and discusses wireless LANs and such standards as IEEE 802.11. The chapter then shifts to cellular networks, one of the main backbones of our wireless networking infrastructure. *Mobile IP* and *Wireless mesh networks* (WMNs), including WiFi and WiMAX technologies, are introduced at the end of this chapter.

Chapter 7, Routing and Internetworking, focuses on routing in wide area networks (WANs) and introduces related routing algorithms and protocols. Our networking infrastructure is clearly classified into those networks that use optimal routes and those that use nonoptimal routes. These two classes of algorithms are described in detail. Routing protocols are also classified as those that are applied within a domain and those that are applied beyond a domain. This chapter also presents congestion-control algorithms: *network-congestion control* and *link-flow control*. The chapter also looks at *random early detection* for congestion control and describes a useful technique to estimate the link-blocking probability.

Chapter 8, Transport and End-to-End Protocols, first looks at the basics of the *transport layer* and demonstrates how a simple file is transferred. This layer handles the details of data transmission. Several techniques for transmission control and protocol

(TCP) congestion control are discussed. Next, *congestion-avoidance* methods, which are methods of using precautionary algorithms to avoid a possible congestion in a TCP session, are presented. The chapter ends with a discussion of methods of ATM congestion control.

Chapter 9, Applications and Network Management, presents the fundamentals of the *application layer*, which determines how a specific user application should use a network. Among the applications are the *Domain Name System* (DNS); *e-mail protocols*, such as SMTP, and the *World Wide Web* (WWW).

Chapter 10, Network Security, focuses on security aspects of networks. After introducing network threats, hackers, and attacks, this chapter discusses encryption techniques: public- and private-key protocols, encryption standards, key-exchange algorithms, authentication methods, digital signature and secure connections, firewalls, IPsec, and security methods for virtual private networks.

Chapter 11, Packet Queues and Delay Analysis, begins Part II, discussing Little's theorem, Markov chain theorem, and birth and death processes. Queueing-node models are presented with several scenarios: finite versus infinite queueing capacity, one server versus several servers, and Markovian versus non-Markovian systems. Non-Markovian models are essential for many network applications, as multimedia traffic cannot be modeled by Markovian patterns. In addition, delay analysis, based on networks of queues, is discussed. *Burke's theorem* is applied in both serial and parallel queueing nodes. *Jackson's theorem* is presented for situations in which a packet visits a particular queue more than once, resulting in *loops* or *feedback*.

Chapter 12, Quality of Service and Resource Allocation, covers quality-of-service issues in networking. The two broad categories of QoS discussed are the *integrated services approach*, for providing service quality to networks that require maintaining certain features in switching nodes; and the *differentiated services approach* (DiffServ), which is based on providing quality-of-service support to a broad class of applications. These two categories include a number of QoS protocols and architectures, such as *traffic shaping*, *admission control*, *packet scheduling*, *reservation methods*, the *Resource Reservation Protocol* (RSVP), and traffic conditioner and bandwidth broker methods. This chapter also explains fundamentals of resource allocation in data networks.

Chapter 13, Networks in Switch Fabrics, looks inside switch fabrics of such Internet devices as routers. The chapter begins by classifying characteristics of switching networks and presenting features and basic definitions of switch fabrics. As the building blocks of switching fabrics, *crossbar switches* are emphasized. In particular, a case study at the end of chapter combines a number of buffered crosspoints to form a buffered crossbar. A number of other switch architectures—both blocking and nonblocking, as

well as, shared-memory, *concentration-based*, and *expansion-based* switching networks are presented.

Chapter 14, Optical Networks and WDM Systems, presents principles of fiber-optic communications and networking. The optical communication technology uses principles of light emission in the glass medium, which can carry more information over longer distances than electrical signals can carry in a copper or coaxial medium. The discussion on optical networks starts with basic optical devices, such as *optical filters*, *wavelength-division multiplexers* (WDMs), *optical switches*, and *optical buffers* and *optical delay lines*. After detailing optical networks using routing devices, the chapter discusses *wavelength re-use and allocation* as a link in all-optical networks. The chapter ends with a case study on an optical switching network, presenting a new topology: the *spherical switching network* (SSN).

Chapter 15, Multicasting Techniques and Protocols, covers the multicast extension of routing protocols in the Internet. First, the chapter defines basic terms and algorithms: multicast group, multicast addresses, and multicast tree algorithms, which form the next set of foundations for understanding packet multicast in the Internet. Two main classes of protocols are discussed: *intradomain* multicast routing protocols, by which packets are multicast within a domain; and *interdomain* routing protocol, by which packet multicast among domains is managed. In addition, techniques and algorithms used within the hardware of routers are introduced.

Chapter 16, VPNs, Tunneling, and Overlay Networks, introduces some useful Internet applications. The chapter explains how networks can be *overlaid* or *tunneled* and describes *virtual private networks* (VPNs), by which a private-sector entity tunnels over the public networking infrastructure, maintaining private connections. Other, related topics in this chapter are *multiprotocol label switching* (MPLS) networks and *overlay networks*.

Chapter 17, Compression of Digital Voice and Video, focuses on data-compression techniques for voice and video to prepare digital voice and video for multimedia networking. The chapter starts with the analysis of information-source fundamentals, source coding, and limits of data compression and explains all the steps of the conversion from raw voice to compressed binary form, such as sampling, quantization, and encoding. The chapter also summarizes the limits of compression and explains typical processes of still-image and video-compression techniques, such as JPEG, MPEG, and MP3. An end-of-chapter case study covers most of the chapter content, looking at FAX compression.

Chapter 18, VoIP and Multimedia Networking, presents the transportation of real-time signals along with the signaling protocols used in voice over IP (VoIP)

telephony and multimedia networking. The chapter presents protocols designed to provide real-time service requirements to the Internet. After discussing the *Session Initiation Protocol* (SIP) and the *H.323 series of protocols*, which are responsible for session signaling and numbering, real-time transport protocols, such as *Real-Time Transport protocol* (RTP) and the *Real-Time Control Protocol* (RTCP) are presented. The next topic is streaming video in a single server, using *content distribution networks* (CDNs). Also discussed is the *Stream Control Transmission Protocol* (SCTP), which provides a general-purpose transport protocol for transporting stream traffic. The chapter ends with detailed streaming source modeling and analysis.

Chapter 19, Mobile Ad-Hoc Networks, presents a special type of wireless networks, known as the *mobile ad-hoc network* (MANET). Ad-hoc networks do not need any fixed infrastructure to operate and support dynamic topology scenarios where no wired infrastructure exists. The chapter explains how a mobile user can act as a routing node and how a packet is routed from a source to its destination without having any static router in the network. The chapter also discusses *table-driven routing protocols* such as DSDV, CGSR, and WRP, and also *source-initiated routing protocols*, as well as DSR, ABR, TORA, and AODV. At the end of the chapter, we will discuss the security of ad-hoc networks.

Chapter 20, Wireless Sensor Networks, presents an overview of such sensor networks and describes intelligent sensor nodes, as well as an overview of a protocol stack for sensor networks. The chapter explains how the “power” factor distinguishes the routing protocols of sensor networks from those of computer networks and describes *clustering protocols* in sensor networks. These protocols specify the topology of the hierarchical network partitioned into nonoverlapping *clusters* of sensor nodes. The chapter also presents a typical routing protocol for sensor networks, leading to a detailed numerical case study on the implementation of a clustering protocol. This chapter ends with *ZigBee technology*, based on IEEE standard 802.15.4. This technology uses low-power nodes and is a well-known low-power standard.

Exercises and Computer Simulation Projects

A number of exercises are given at the end of each chapter. The exercises normally challenge readers to find the directions to solutions in that chapter. The answers to the exercises are not always simple and may be more elusive, but this is typical of real and applied problems in networking. These problems encourage the reader to go back through the text and pick out what the instructor believes is significant. Besides typical exercises problems, there are numerous occasions for those who wish to incorporate

projects into their courses. The computer simulation projects are normally meant to be a programming miniproject. Projects listed in the exercises range from simulations to partial hardware design.

Throughout the text are case studies that show how and where computer communication integration is used with the materials studied in the associated chapter. A case study is basically a practical example for better understanding the essence of the corresponding chapter.

Appendices

The book's appendixes make it essentially self-sufficient. **Appendix A, Glossary of Acronyms**, defines acronyms. **Appendix B, RFCs**, encourages readers to delve more deeply into each and every protocol presented in the book by consulting the many references provided. **Appendix C, Probabilities and Stochastic Processes**, reviews probabilities, random variables, and random processes.

Instructions and Instructor Supplements

The text can be used in a variety of ways. An instructor can use Part I of the book for the first graduate or a senior undergraduate course. In this case, one or two additional chapters from Part II, such as Chapters 13, 15, or 16, can also be included.

Part II of the text is aimed at the second graduate course in computer communication networks. An instructor can also choose the desired chapters, depending on the need and the content of her/his course. For example, if the graduate course is geared more toward architectures, Chapters 12, 13, 14, and 15 would be appropriate. Nevertheless, an instructor can include a couple of chapters from Part I, such as Chapters 3 and 6, in her/his graduate course.

An *instructor's solutions manual* is available to qualified instructors. Other instruction material, such as Power Point presentations will also be provided to instructors. Please contact the publisher for more information.

Acknowledgments

Writing a text is rarely an individual effort. Many experts from industry and academia graciously provided help. I would like to thank them all very warmly for their support. Many of them have given me invaluable ideas and support during this project. I should

acknowledge all those scientists, mathematicians, professors, engineers, authors, and publishers who helped me in this project.

I am honored to publish this book with Prentice Hall. I wish to express my deep gratitude to everyone who made a tremendous effort to make this project succeed. In particular, I would like to thank acquisitions editor Catherine Nolan for all her help. In the middle of the last phase of the review process, Catherine voluntarily set up a meeting while I was attending a conference in Boston and provided me with invaluable information on my remaining tasks in this project. I would also like to thank Managing Editor John Fuller, Marketing Manager Suzette Ciancio, Project Editor Lara Wysong, Development Editor Jinnifer Blackwell, Copy Editor Evelyn Pyle, and San Francisco Bay Sales Representative Ellen Wynn, who enthusiastically initiated my manuscript to the publisher.

I am grateful to all reviewers of this book for making constructive suggestions that helped me reshape the book to its present form. I would especially like to recognize the following people, who provided invaluable feedback during the writing phase of the manuscript. I took all their comments seriously and incorporated them into the manuscript. I greatly appreciate their time spent on this project.

Professor Nirwan Ansari (New Jersey Institute of Technology)

Professor Mohammed Atiquzzaman (University of Oklahoma)

Dr. Radu Balan (Siemens Corporate Research)

R. Bradley (About.Com, CompNetworking Guide)

Deepak Biala (On-Fiber Communications)

Dr. Robert Cane (VPP, United Kingdom)

M. Kevin Choy (Atmel, Colorado)

Professor Rod Fatohi (San Jose State University)

Professor Zongming Fei (University of Kentucky)

Dr. Carlos Ferari (JTN-Network Solutions)

Professor Jim Griffioen (University of Kentucky)

Dr. Jac Grolan (Alcatell)

Ajay Kalamber

Aurna Ketaraju (Intel)

Dr. Hardeep Maldia (Sermons Communications)

Will Morse (Texas Safe-Computing)

Professor Sarhan Musa (P. V. Texas A&M University)
Professor Achille Pattavina (Politecnico di Milano TNG)
Dr. Robert J. Paul (NsIM Communications)
Bala Peddireddi (Intel)
Christopher H. Pham (Cisco Systems)
Jasmin Sahara (University of Southern California)
Dipti Sathe (Altera Corporation)
Dr. Simon Sazeman (Siera Communications and Networks)
Professor Mukesh Singhal (University of Kentucky)
Professor Kazem Sohraby (University of Arkansas)
Dr. Richard Stevensson (BoRo Comm)
Kavitha Venkatesan (Cisco Systems)
Dr. Steve Willmar (SIM Technology)
Dr. Hemeret Zokhil (JPLab)

I am truly indebted to my graduate students, who helped me throughout the long phase of preparing the manuscript of this textbook. Over the years, more than 75 of my master's and PhD students read various portions of this book and made constructive comments. I wish them the best for their honest support and verbal comments on early versions of this book used in my class lectures. I am especially thankful to those who reviewed some sections of the book while taking my networking courses: Nasima Akhtar, Robert Bergman, Anushya Dharmarajan, Sudha Immaneni Howard Chan, Rekha Modipalle, Rinku Pal, Vishal Parikh, Preeti Pujni, Rashi Sharma, Maryam Tabesh, and Sitthapon Pumpichet.

Special thanks to Marzieh Veyseh, University of California—Santa Cruz, whose work on sensor networks resulted in our mutually published journal articles and for making all the information about sensor networks available for Chapter 20. Many thanks to Dr. Belle W. Wei, Dean of Engineering, SJSU, whose constructive comments resulted in great improvements in those journal articles and impacted Chapter 20.

I am also thankful to Dr. Eftekhari for a great technical help and support, and Professor Michael O'Flynn (not only a colleague but also a mentor), who graciously read through some sections of this book and made invaluable comments. Last but not least, thanks to Parviz Karandish, my best friend and a mentor who has helped me in many ways to make this project happen.

How to Contact the Author

Please feel free to send me any feedback at the Department of Electrical Engineering, San Jose State University, San Jose, CA 95192, U.S.A, or via e-mail at nmir@sjtu.edu. I would love to hear from you, especially if you have suggestions for improving this book. I will carefully read all review comments. You can find out more about me at www.engr.sjsu.edu/nmir. I hope that you enjoy the text and that you receive from it a little of my enthusiasm for computer communications.

Nader F. Mir
San Jose, California

About the Author

Nader F. Mir received the B.Sc. degree (with honors) in electrical and computer engineering in 1985 and the M.Sc. and PhD degrees, both in electrical engineering, from Washington University in St. Louis, Missouri, in 1990 and 1994, respectively.

He is currently a full professor and associate chairman of the Electrical Engineering Department at San Jose State University, California. He is also the director of MSE Program in Optical Sensors for Lockheed Martin Space Systems. Previously, he was an associate professor at this school and assistant professor at the University of Kentucky in Lexington. From 1994 to 1996, he was a research scientist at the Advanced Telecommunications Institute, Stevens Institute of Technology, in New Jersey, working on the design of advanced telecommunication networks. From 1990 to 1994, he was with the Computer and Communications Research Center at Washington University in St. Louis and worked as a research assistant on design and analysis of high-speed switching-systems projects.

His research interests are analysis of computer communication networks, design and analysis of switching systems, network design for wireless ad-hoc and sensor systems, and applications of digital integrated circuits in computer communications.

He is a senior member of the IEEE and has served as a member of the Technical Program Committee and the Steering Committee of a number of major IEEE networking conferences, such as WCNC, GLOBECOM, and ICC. Dr. Mir has published numerous technical journal and conference papers, all in the field of communications and networking. He has published a book on video communication engineering and a textbook, *Computer and Communication Networks*, published by Prentice Hall.

Dr. Mir has received a number of prestigious national and university awards, including the university teaching recognition award and research excellence award. He

is also the recipient of the 2004 IASTED Outstanding Tutorial Presentation award. He holds a successful United States patent for his invention of a new switching system for computer networks.

Currently, he has several journal editorial positions: editorial board member of the *International Journal of Internet Technology and Secured Transactions*, editor of the *Journal of Computing and Information Technology*, and associate editor of *IEEE Communication Magazine*.

PART I

FUNDAMENTAL CONCEPTS

This page intentionally left blank

CHAPTER 1

Packet-Switched Networks

Computer and communication networks to provide a wide range of services, from simple networks of computers to remote-file access, digital libraries, videoconferencing, and networking billions of users and devices. Before exploring the world of computer and communication networks, we need to study the fundamentals of *packet-switched networks* as the first step. Packet-switched networks are the backbone of the data communication infrastructure. Therefore, our focus in this chapter is on the big picture and the conceptual aspects of this backbone:

- *Basic definitions in data networks*
- *Types of packet-switched networks*
- *Packet size and optimizations*

We start with the basic definitions and fundamental concepts, such as *messages*, *packets*, and *frames*, and *packet switching* versus *circuit switching*. We learn what the Internet is and how Internet service providers (ISPs) are formed. We then proceed to types of packet-switched networks and how a message can be handled by either *connection-oriented networks* or *connectionless networks*. Because readers must get a good understanding of *packets* as data units, packet size and optimizations are also discussed.

1.1 Basic Definitions in Data Networks

Communication networks have become essential media for homes and businesses. The design of modern computer and communication networks must meet all the requirements for new communications applications. A ubiquitous *broadband network* is the goal of the networking industry. Communication services need to be available anywhere and anytime. The broadband network is required to support the exchange of multiple types of information, such as voice, video, and data, among multiple types of users, while satisfying the performance requirement of each individual application. Consequently, the expanding diversity of high-bandwidth communication applications calls for a unified, flexible, and efficient network. The design goal of modern communication networks is to meet all the networking demands and to integrate capabilities of networks in a broadband network.

Packet-switched networks are the building blocks of computer communication systems in which data units known as *packets* flow across networks. The goal of a broadband packet-switched network is to provide flexible communication in handling all kinds of connections for a wide range of applications, such as telephone calls, data transfer, teleconferencing, video broadcasting, and distributed data processing. One obvious example for the form of traffic is *multirate* connections, whereby traffic containing several different bit rates flows to a communication node. The form of information in packet-switched networks is always digital bits. This kind of communication infrastructure is a significant improvement over the traditional telephone networks known as *circuit-switched networks*.

1.1.1 Packet Switching versus Circuit Switching

Circuit-switched networks, as the basis of conventional telephone systems, were the only existing personal communication infrastructures prior to the invention of packet-switched networks. In the new communication structure, voice and computer data are treated the same, and both are handled in a unified network known as a packet-switched network, or simply an integrated data network. In conventional telephone networks, a circuit between two users must be established for a communication to occur. Circuit-switched networks require resources to be reserved for each pair of end users. This implies that no other users can use the already dedicated resources for the duration of network use. The reservation of network resources for each user results in an inefficient use of bandwidth for applications in which information transfer is bursty.

Packet-switched networks with a unified, integrated data network infrastructure known as the Internet can provide a variety of communication services requiring

different bandwidths. The advantage of having a unified, integrated data network is the flexibility to handle existing and future services with remarkably better performance and higher economical resource utilizations. An integrated data network can also derive the benefits of central network management, operation, and maintenance. Numerous requirements for integrated packet-switched networks are explored in later chapters:

- Having robust routing protocols capable of adapting to dynamic changes in network topology
- Maximizing the utilization of network resources for the integration of all types of services
- Providing quality of service to users by means of priority and scheduling
- Enforcing effective congestion-control mechanisms that can minimize dropping packets

Circuit-switched networking is preferred for real-time applications. However, the use of packet-switched networks, especially for the integration and transmission of voice and data, results in the far more efficient utilization of available bandwidth. Network resources can be shared among other eligible users. Packet-switched networks can span a large geographical area and comprise a web of switching *nodes* interconnected through transmission links. A network provides links among multiple users facilitating the transfer of information. To make efficient use of available resources, packet-switched networks dynamically allocate resources only when required.

1.1.2 Messages, Packets, and Frames

A packet-switched network is organized as a multilevel hierarchy. In such networks, digital messages are fragmented into one or more smaller units of messages, each appended with a *header* to specify control information, such as the source and the destination addresses. This new unit of formatted message is called a *packet*, as shown in Figure 1.1. Packets are forwarded to a data network to be delivered to their destinations. In some circumstances, packets are also required to be attached together or further fragmented, forming a new packet known as a *frame*. Sometimes, a frame may be required to have multiple headers to carry out multiple tasks in multiple layers of a network, as shown in the figure.

As shown in Figure 1.2, two packets, A and B, are being forwarded from one side of a network to the other side. Packet-switched networks can be viewed from either an external or an internal perspective. The external perspective focuses on the network

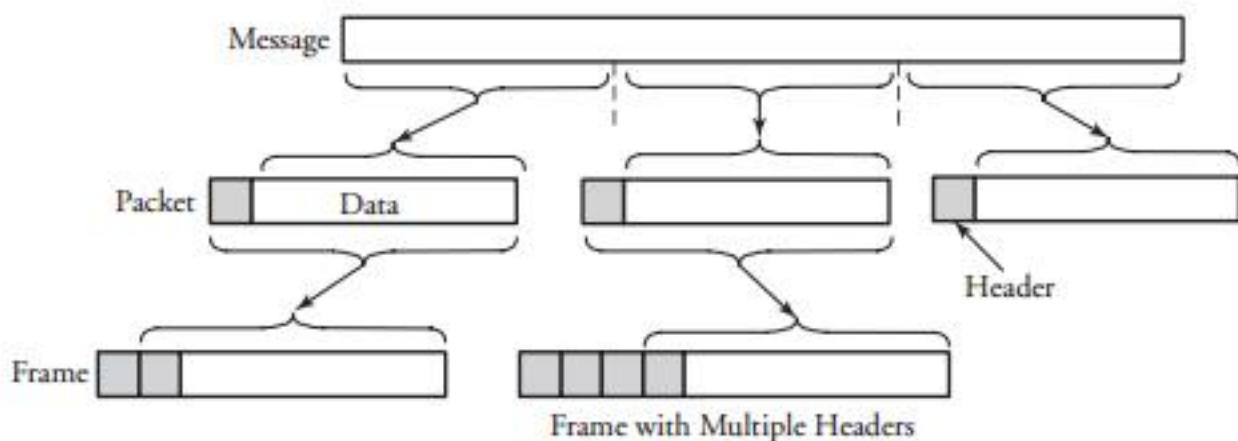


Figure 1.1 Creating packets and frames out of a raw digital message

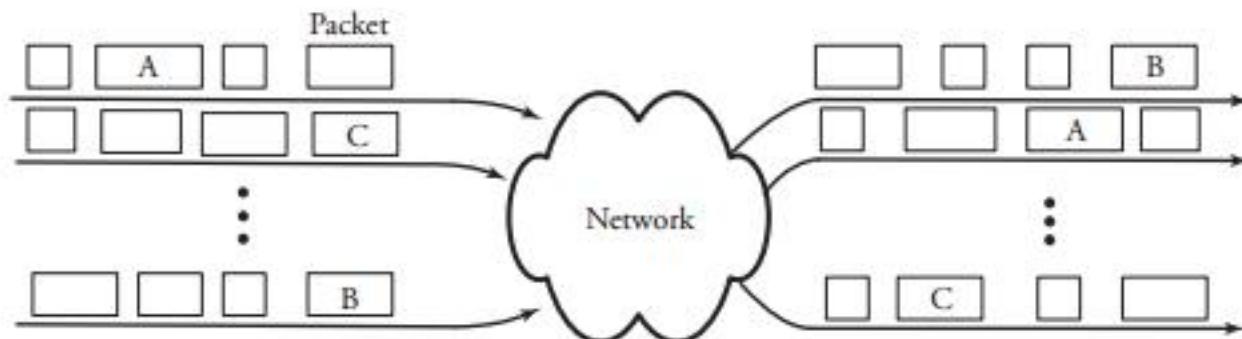


Figure 1.2 A packet-switched network receiving various-sized packets to route out

services provided to the upper layers; the internal perspective, on the fundamentals of *network topology*, structure of communication protocols, and addressing schemes.

A single packet, as the smallest unit of data for networking, may even be split into multiple packets before transmission. This well-known technique is called *packet fragmentation*. Apart from measuring the delay and ensuring that a packet is correctly sent to its destination, we also focus on delivering and receiving packets in a correct sequence when the data is fragmented. The primary function of a network is directing the flow of data among the users.

1.1.3 The Internet

The *Internet* is the collection of hardware and software components that make up our global communication network. The Internet is indeed a collaboration of interconnected communication vehicles that can network all connected communicating devices and equipment and provide services to all distributed applications. It is almost impossible to plot an exact representation of the Internet, since it is continuously being

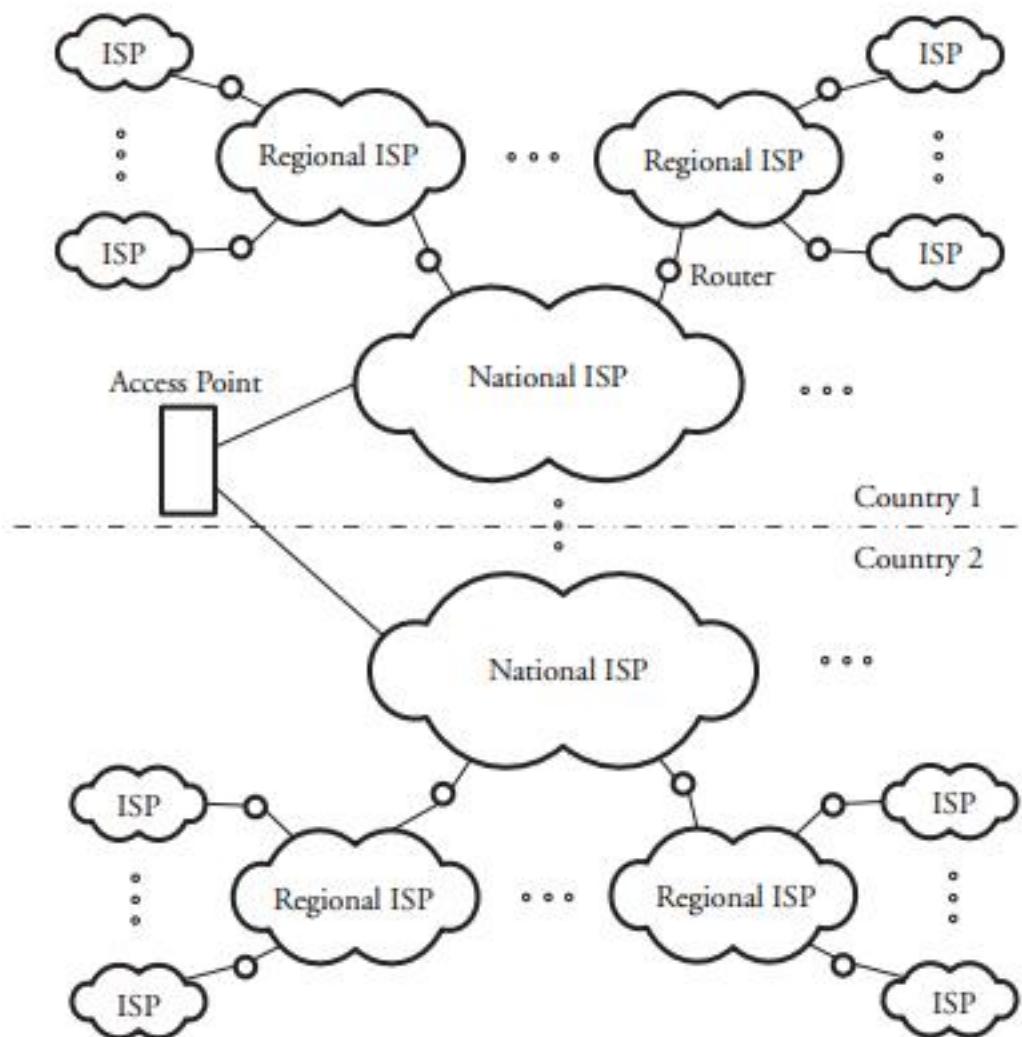


Figure 1.3 The Internet, a global interconnected network

expanded or altered. One way of imagining the Internet is shown in Figure 1.3, which illustrates a big-picture view of the worldwide computer network.

To connect to the Internet, users need the services of an *Internet service provider*. Each country has international or national service providers, regional service providers, and local service providers. At the top of the hierarchy, national Internet service providers connect nations or provinces together. The traffic between each two national ISPs is very heavy. Two such ISPs are connected together through complex switching stations called *network access points* (NAPs). Each NAP has its own system administrator.

In contrast, *regional Internet service providers* are smaller ISPs connected to a national ISP in a hierarchical chart. A *router* can operate as a device to connect to ISPs. Routers operate on the basis of one or more common *routing protocols*. In computer networks, the entities must agree on a protocol, a set of rules governing data communications and defining when and how two users can communicate with each other.

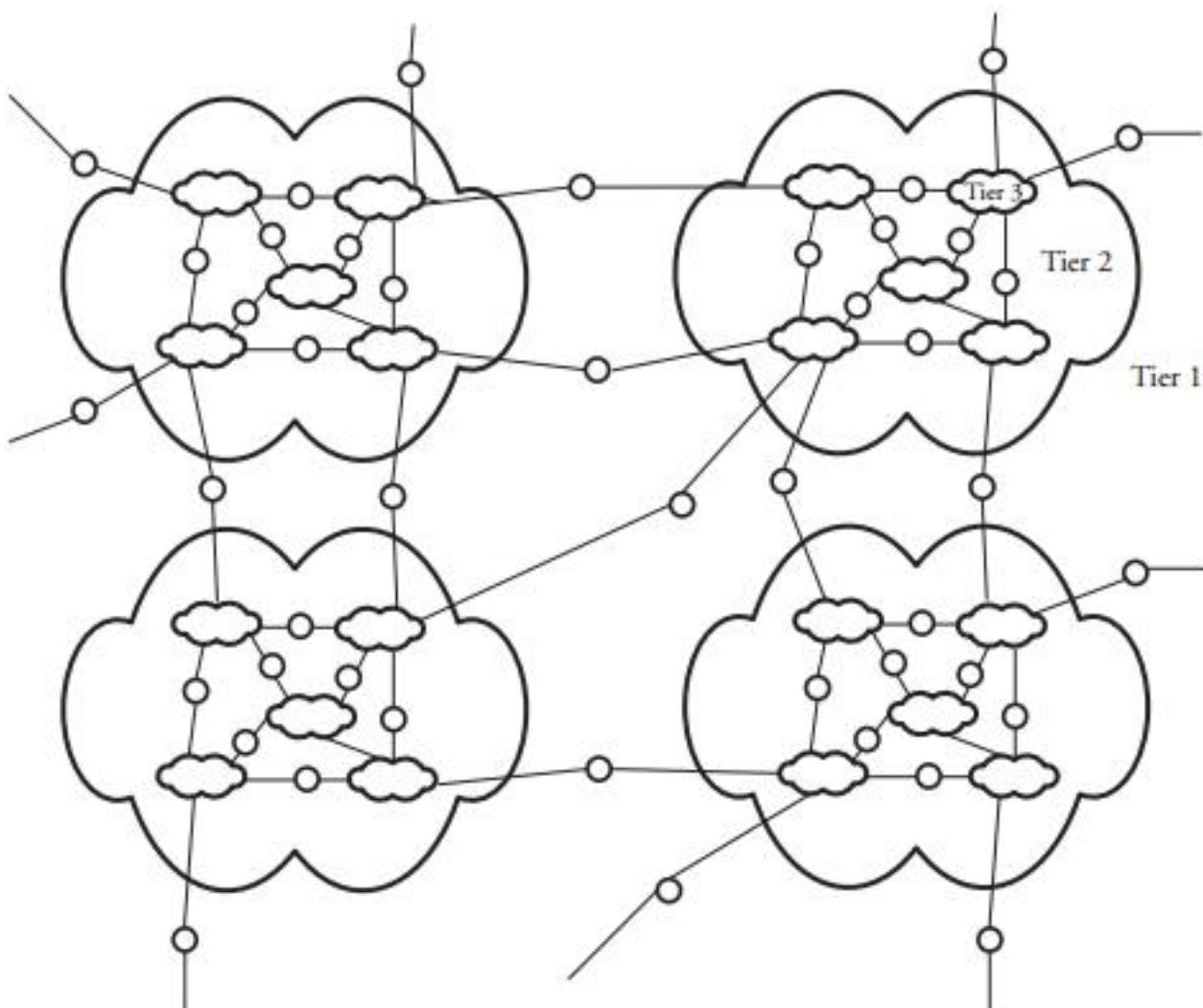


Figure 1.4 Hierarchy of networks

Each regional ISP can give services to part of a province or a city. The lowest networking entity of the Internet is a *local Internet service provider*. A local ISP is connected to a regional ISP or directly to a national service provider and provides a direct service to end users called *hosts*. End users and systems are connected together by communication *links*. An organization that supplies services to its own employees can also be a local ISP.

Figure 1.4 illustrates a different perspective of the global interconnected network. Imagine the global network in a hierarchical structure. Each ISP of a certain hierarchy or tier manages a number of other network domains at its lower hierarchy. The structure of such networks resembles the hierarchy of nature from the universe to atoms and molecules. Here, Tier 1, Tier 2, and Tier 3 represent, respectively, a national ISP, a regional ISP, and a local ISP.

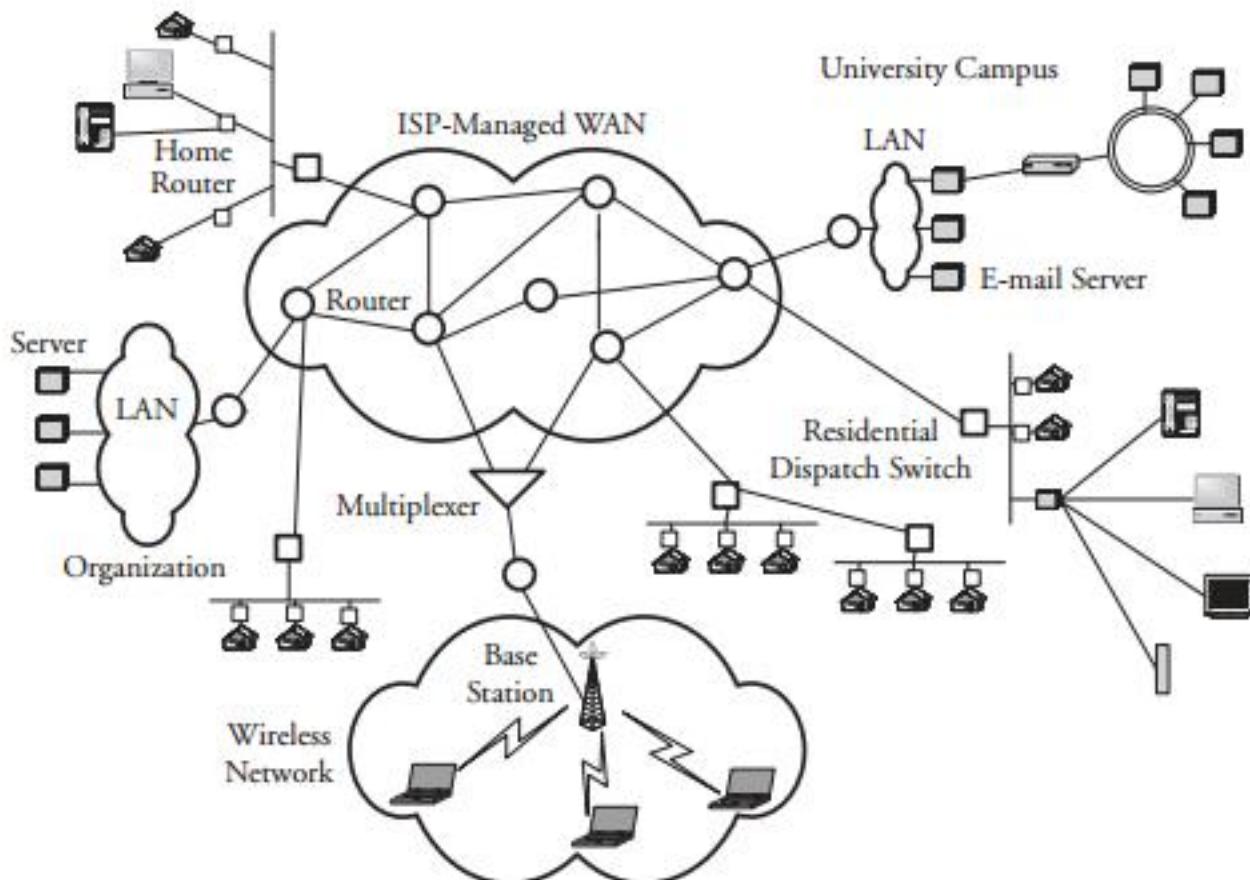


Figure 1.5 Overview of a regional Internet service provider (ISP)

1.1.4 ISPs and Internetwork Components

Figure 1.5 shows an Internet service provider. Networks can be classified into two main categories: *wide area networks* (WANs) and *local area networks* (LANs). A wide area network can be as large as the entire infrastructure of the data network access system known as the Internet. Figure 1.5 shows several networks, including LANs and WANs. End systems are indirectly connected to each other through intermediate *switching nodes* known as *packet routers*, or simply *routers*. Switching devices are key components that allow the flow of information to be switched over other links. Meanwhile, multiple local area networks are interconnected using *border routers* to form a wide area network. These routers contain information about the network routes, and their tasks are to route packets to requested destinations.

Multiple users accessing a single transmission medium at the same time are connected not only by switching nodes and interconnection links but also an access *multiplexer*. The multiplexer combines traffic from several nodes into a cumulative flow. This technique improves the bandwidth utilization efficiently. In Figure 1.5, we can

see that some aggregated links from a WAN are multiplexed into one link directed to a *wireless network*. In most cases, the aggregated packets are forwarded through a major network access node, as seen in Figure 1.5 for the wireless infrastructure. A separate network managed by a network administrator is known as a *domain*, or an *autonomous system*.

As an example of the local area network, a college campus network is connected to the Internet via a border router that connects the campus to an Internet service provider. ISP users are connected to access points of a WAN, as seen in Figure 1.5. Various ISPs can be interconnected through a high-speed router. Service providers have varying policies to overcome the problem of bandwidth allocations on routers. An ISP's *routing server* is conversant with the policies of all other service providers. Therefore, the routing server can direct the received routing information to an appropriate ISP.

When a link failure occurs, in packet-switched networks, the neighboring nodes share the fault information with other nodes, resulting in updating the routing tables. This way, packets may get routed through alternative paths bypassing the fault. Building the *routing table* in a router is one of the principal challenges of packet-switched networks. Designing the routing table for large networks requires maintaining data pertaining to traffic patterns and network topology information.

1.2 Types of Packet-Switched Networks

Packet-switched networks are classified into *datagram* or *connectionless* networks and *virtual-circuit* or *connection-oriented* networks, depending on the technique used for transferring information. The simplest form of a network service is based on the connectionless protocol. In this type of network, a user can transmit a packet anytime, without notifying the network layer. Packets are encapsulated into a certain "formatted" header, resulting in the basic Internet transmission unit of data, or *datagram*. A datagram is then sent over the network, with each router receiving the datagram forwarding it to the best router it knows, until the datagram reaches the destination. In this scheme, packets may be routed independently over different paths. However, the packets may arrive out of sequence. In this case, a certain network function (to be discussed later) takes care of the error control, flow control, and resequencing packets.

A related, though more complex, service is the connection-oriented protocol. Packets are transferred through an established virtual circuit between a source and a destination. When a connection is initially set up, network resources are reserved for the call duration. After the communication is finished, the connection is terminated, using a connection-termination procedure. During the call setup, the network can offer a

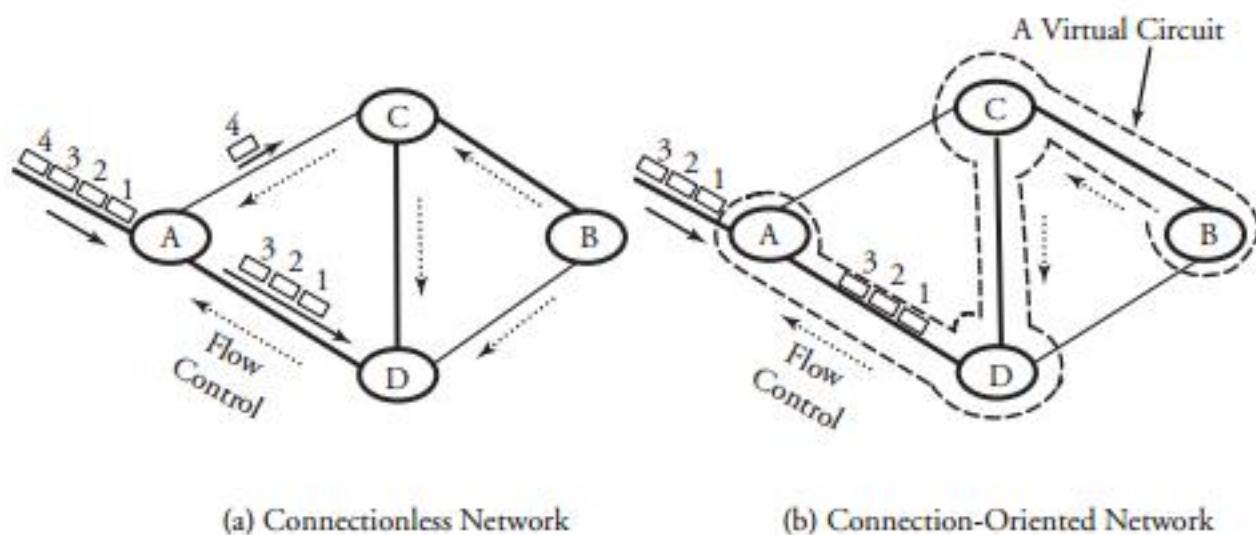


Figure 1.6 Two models of data networks: (a) a connectionless network and (b) a connection-oriented network

selection of options, such as best-effort service, reliable service, guaranteed delay service, and guaranteed bandwidth service, as explained in Chapters 8 and 12.

1.2.1 Connectionless Networks

Connectionless packet switching achieves high throughput at the cost of queuing delay. In this approach, large packets are normally fragmented into smaller packets. Packets from a source are routed independently of one another. The connectionless-switching approach does not require a call setup to transfer packets, but it has error-detection capability. The main advantage of this scheme is its capability to route packets through an alternative path in case a fault is present on the desired transmission link. On the flip side, this may result in the packet's arriving out of order and requiring sequencing at the destination.

Figure 1.6 (a) shows the routing of four packets in a connectionless network from point A to point B. The packets traverse the intermediate nodes in a *store-and-forward* fashion, whereby packets are received and stored at a node on a route; when the desired output of the node is free for that packet, the output is forwarded to its next node. In other words, on receipt of a packet at a node, the packet must wait in a queue for its turn to be transmitted. Nevertheless, packet loss may still occur if a node's buffer becomes full. The node determines the next hop read from the packet header. In this figure, the first three packets are moving along the path A, D, C, and B, whereas the fourth packet moves on a separate path, owing to congestion on path A-D.

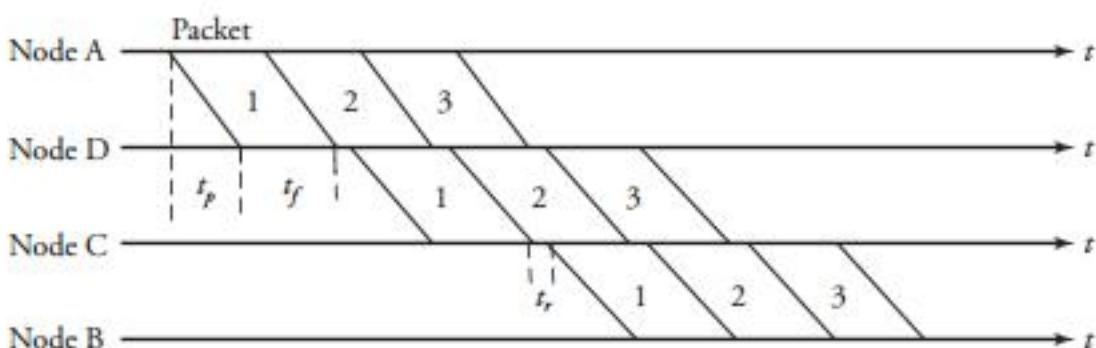


Figure 1.7 Signaling delay in a connectionless environment

The delay model of the first three packets discussed earlier is shown in Figure 1.7. The total transmission delay for a message three packets long traversing from the source node A to the destination node B can be approximately determined. Let t_p be the propagation delay between each two nodes, and let t_f be the packet-transfer time from one node to the next one. A packet is processed once it is received at a node with a processing time t_r . The total transmission delay, D_p , for n_h nodes and n_p packets, in general is

$$D_p = [n_p + (n_h - 2)]t_f + (n_h - 1)t_p + n_h t_r. \quad (1.1)$$

In this formula, D_p gives only the transmission delay. As discussed in Chapter 11, another delay component, known as the *packet-queueing delay*, can be added to D_p . At this point, we focus only on the transmission delay and will discuss the queueing delay later.

Example. Figure 1.7 shows a timing diagram for the transmission of three packets on path A, D, C, B in Figure 1.6. Determine the total delay for transferring these three packets from node A to node B.

Solution. Assume that the first packet is transmitted from the source, node A, to the next hop, node D. The total delay for this transfer is $t_p + t_f + t_r$. Next, the packet is similarly transferred from node D to the next node to ultimately reach node B. The delay for each of these jumps is also $t_p + t_f + t_r$. However, when all three packets are released from node A, multiple and simultaneous transmissions of packets become possible. Thus, the total delay for all three packets to traverse the source and destination via two intermediate nodes is $D_p = 3t_p + 5t_f + 4t_r$.

Connectionless networks demonstrate the efficiency of transmitting a large message as a whole, especially in noisy environments, where the error rate is high. It is obvious

that the large message should be split into packets. Doing so also helps reduce the maximum delay imposed by a single packet on other packets. In fact, this realization in fact resulted in the advent of connectionless packet switching.

1.2.2 Connection-Oriented Networks

In *connection-oriented*, or *virtual-circuit*, networks, a route set up between a source and a destination is required prior to data transfer, as in the case of conventional telephone networks. Figure 1.6 (b) shows a connection-oriented packet-switched network. The connection set-up procedure shown in this figure requires three packets to move along path A, D, C, and B with a prior connection establishment. During the connection set-up process, a virtual path is dedicated, and the forwarding routing tables are updated at each node in the route.

Virtual-circuit packet switching typically reserves the network resources, such as the buffer capacity and the link bandwidth, to provide guaranteed quality of service and delay. The main disadvantage in connection-oriented packet-switched networks is that in case of a link or switch failure, the call set-up process has to be repeated for all the affected routes. Also, each switch needs to store information about all the flows routed through the switch.

The total delay in transmitting a packet in connection-oriented packet switching is the sum of the connection set-up time and the data-transfer time. The data-transfer time is the same as the delay obtained in connectionless packet switching. Figure 1.8 shows the overall delay for the three packets presented in the previous example. The transmission of the three packets starts with *connection request* packets and then *connection accept* packets. At this point, a circuit is established, and a partial path bandwidth is reserved for this connection. Then, the three packets are transmitted. At the end, a *connection release* packet clears and removes the established path.

The estimation of total delay time, D_t , to transmit n_h packets is similar to the one presented for connectionless networks. For connection-oriented networks, the

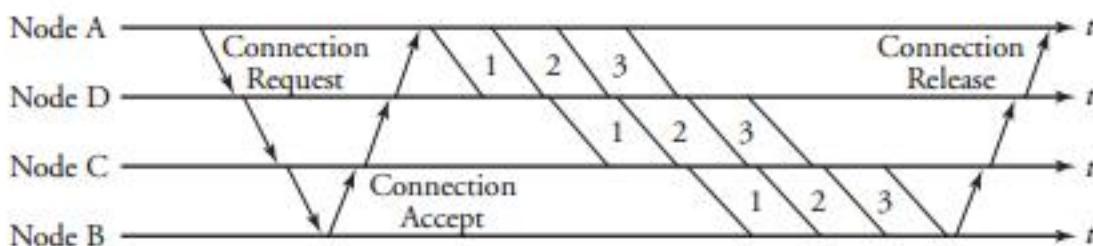


Figure 1.8 Signaling delay in a connection-oriented packet-switched environment

total time consists of two components: D_p , which represents the time to transmit packets, and D_c , which represents the time for the control packets. Control-packets' time includes the transmission delay for the connection-request packet, the connection-accept packet, and the connection-release packet:

$$D_t = D_p + D_c. \quad (1.2)$$

Another feature, called *cut-through switching*, can significantly reduce the delay. In this scheme, the packet is forwarded to the next hop as soon as the header is received and the destination is parsed. We see that the delay is reduced to the aggregate of the propagation times for each hop and the transfer time of one hop. This scheme is used in applications in which retransmissions are not necessary. Optical fiber transmission has a very low loss rate and hence uses cut-through switching to reduce the delay in transmitting a packet.

1.3 Packet Size and Optimizations

Packet size has a substantial impact on the performance of data transmission. Consider Figure 1.9, which compares the transmission of a 60-byte message using two different packet sizes with a packet header of 5 bytes. In the first scheme, the message is fragmented into three pieces of 20 bytes each, resulting in three packets of 25 bytes each. If these three packets are supposed to be transmitted over path A, D, C, and B in Figure 1.6, 75-byte units of delay are required. In contrast, if the message is fragmented into six messages of 10 bytes each, resulting in 15-byte packets, the total delay would be 60-byte units of delay. The reason for the time reduction is the parallel transmission of multiple packets at nodes D and C, since packets are smaller. This trend of delay reduction using smaller packets, however, is reversed at a certain point, owing to the dominance of packet overhead when a packet becomes very small.

To analyze packet size optimization, consider a link with a speed of s b/s or a bit rate of μ packets per second. Assume that packets of size $d + h$ are sent over this link at the rate λ packets per second, where d and h are the sizes of the packet data and the packet header, respectively, in bits. Clearly,

$$\mu = \frac{s}{d + h}. \quad (1.3)$$

We define *link utilization* to be $\rho = \lambda/\mu$. Then the percentage of link utilization used by data, ρ_d , is obtained by

$$\rho_d = \rho \left(\frac{d}{d + h} \right). \quad (1.4)$$

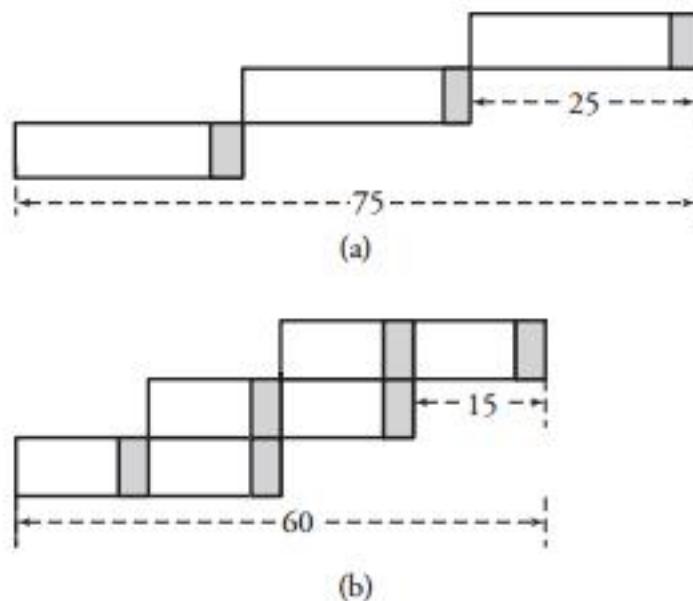


Figure 1.9 Comparison of two cases of transmitting data: (a) using three packets and (b) using six packets

The average delay per packet, D , can be calculated by using $\mu - \lambda$, where this term exhibits how close the offered load is to the link capacity:

$$D = \frac{1}{\mu - \lambda}. \quad (1.5)$$

Using equations (1.3) and (1.4), we can rewrite the average delay per packet as

$$D = \frac{1}{\mu(1-\rho)} = \frac{d+h}{s(1-\rho)} = \frac{d+h}{s \left[1 - \frac{\rho_d}{d} (d+h) \right]}. \quad (1.6)$$

Apparently, the optimum size of a packet depends on several contributing factors. Here, we examine one of the factors by which the delay and the packet size become optimum. For optimality, consider d as one possible variable, where we want

$$\frac{\partial D}{\partial d} = 0. \quad (1.7)$$

This releases the two optimum values:

$$d_{opt} = h \left(\frac{\sqrt{\rho_d}}{1 - \sqrt{\rho_d}} \right) \quad (1.8)$$

and

$$D_{opt} = \frac{h}{s} \left(\frac{\sqrt{\rho_d}}{1 - \sqrt{\rho_d}} \right)^2. \quad (1.9)$$

Note that here, d and D are optimized given only the mentioned variables. The optimality of d and D can also be derived by using a number of other factors that will result in a more accurate approach.

1.4 Summary

This chapter established a conceptual foundation for realizing all upcoming chapters. First, we clearly identified and defined all basic key terms in networking. We showed a big-picture view of computer networks in which from one side, connecting mainframe servers can be connected to a network backbone, and from the other side, home communication devices are connected to a backbone network over long-distance telephone lines. We illustrated how an Internet service provider (ISP) controls the functionality of networks. ISPs have become increasingly involved in supporting packet-switched networking services for carrying all sorts of data, not just voice, and the cable TV industry.

The transfer of data in packet-switched networks is organized as a multilevel hierarchy, with digital messages fragmented into units of formatted messages, or packets. In some circumstances, such as local area networks, packets must be modified further, forming a smaller or larger packet known as a frame. Two types of packet-switched networks are networks using connectionless protocol, in which no particular advanced connection is required, and networks using connection-oriented protocol, in which an advance dedication of a path is required.

A packet size can be optimized. Using the percentage of link utilization used by data, ρ_d , as a main variable, we showed that the optimized packet size and the optimized packet delay depend on ρ_d . The total delay of packet transfer in connectionless networks is significantly smaller than for connection-oriented networks, owing mainly to inclusion of signaling components in connection-oriented networks.

The next two chapters present an overview of the software and hardware foundations of our networking infrastructure. Chapter 2 provides some detail about communication protocols, and Chapter 3 introduces the devices used to construct a computer network.

1.5 Exercises

1. We transmit data directly between two servers 6,000 km apart through a geostationary satellite situated 10,000 km from Earth exactly between the two servers. The data enters this network at 100 Mb/s.
 - (a) Find the propagation delay if data travels at the speed of light (2.3×10^8 m/s).
 - (b) Find the number of bits in transit during the propagation delay.
 - (c) Determine how long it takes to send 10 bytes of data and to receive 2.5 bytes of acknowledgment back.
2. Repeat exercise 1, but this time, suppose that the two servers are 60 meters apart on the same campus.
3. Stored on a CD-ROM is a 200 MB message to be transmitted by an e-mail from one mail server to another, passing three nodes of a *connectionless network*. This network forces packets to be of size 10 KB, including a packet header of 40 bytes. Nodes are 400 miles apart, and servers are 50 miles away from their corresponding nodes. All transmission links are of type 100 Mb/s. The processing time at each node is 0.2 seconds per packet.
 - (a) Find the propagation delays per packet between a server and a node and between nodes.
 - (b) Find the total time required to send this message.
4. Equation 1.2 gives the total delay time for connection-oriented networks. Let t_p be the packet-propagation delay between each two nodes, t_{f1} be the data packet transfer time to the next node, and t_{r1} be the data packet processing time. Also, let t_{f2} be the control-packet transfer time to the next node, and t_{r2} be the control-packet processing time. Give an expression for D in terms of all these variables.
5. Suppose that a 200 MB message stored on a CD-ROM is to be uploaded on a destination through a *virtual-circuit packet-switched network*. This network forces packets to be of size 10 KB, including a packet header of 40 bytes. Nodes are 400 miles apart, and servers are 50 miles away from their corresponding nodes. All transmission links are of type 100 Mb/s. The processing time at each node is 0.2 seconds per packet. For this purpose, the signaling packet is 500 bit long.
 - (a) Find the total connection request/accept process time.
 - (b) Find the total connection release process time.
 - (c) Find the total time required to send this message.

6. We want to deliver a 12 KB message by uploading it in the destination's Web site through a 10-node path of a *virtual-circuit packet-switched network*. For this purpose, the signaling packet is 500 bits long. The network forces packets to be of size 10 KB including a packet header of 40 bytes. Nodes are 500 miles apart. All transmission links are of type 1 Gb/s. The processing time at each node is 100 ms per packet and the propagation speed is 2.3×10^8 m/s.
 - (a) Find the total connection request/accept process time.
 - (b) Find the total connection release process time.
 - (c) Find the total time required to send this message.
7. To analyze the transmission of a 10,000-bit-long packet, we want the percentage of link utilization used by the data portion of a packet to be 72 percent. We also want the ratio of the packet header, h , to packet data, d , to be 0.04. The transmission link speed is $s = 100$ Mb/s.
 - (a) Find the link utilization, ρ .
 - (b) Find the link capacity rate, μ , in terms of packets per second.
 - (c) Find the average delay per packet.
 - (d) Find the optimum average delay per packet.
8. Consider a digital link with a maximum capacity of $s = 100$ Mb/s facing a situation resulting in 80 percent utilization. Equal-size packets arrive at 8,000 packets per second. The link utilization dedicated to headers of packets is 0.8 percent.
 - (a) Find the total size of each packet.
 - (b) Find the header and data sizes for each packet.
 - (c) If the header size is not negotiable, what would the optimum size of packets be?
 - (d) Find the delay for each optimal-sized packet.
9. Develop a signaling delay chart, similar to Figures 1.7 and 1.8, for circuit-switched networks. From required steps, get an idea that would result in the establishment of a telephone call over circuit-switched networks.
10. In practice, the optimum size of a packet estimated in Equation (1.7) depends on several other contributing factors.
 - (a) Derive the optimization analysis, this time also including the header size, h . In this case, you have two variables: d and h .
 - (b) What other factors might also contribute to the optimization of the packet size?

CHAPTER 2

Foundation of Networking Protocols

Users and networks are connected together by certain rules called *network communication protocols*. The Internet Protocol (IP), for example, is responsible for using prevailing rules to establish paths for packets. Communication protocols are the intelligence behind the driving force of packets and are tools by which a network designer can easily expand the capability of networks. One growth aspect of computer networking is clearly attributed to the ability to conveniently add new features to networks. New features can be added by connecting more hardware devices, thereby expanding networks. New features can also be added on top of existing hardware, allowing the network features to expand. The second method of network expansion requires modifications to communication protocols. This chapter focuses on the following:

- *Five-layer TCP/IP model*
- *Seven-layer OSI protocol model*
- *Internet protocols and addressing*
- *Equal-size packet protocol model*

The *five-layer TCP/IP model* is a widely accepted Internet backbone protocol structure. In this chapter, we give an overview of these five layers and leave any further details to be discussed in the remaining chapters. Among these five layers, the basics of *network layer* is designated a separate section in this chapter under Internet protocols. We make this

arrangement since basic definitions related to this layer are required in the following few chapters.

As there are numerous protocols formed together to be used for the movement of packets, the explanation of all other protocols will be spread over almost all upcoming chapters. In the meanwhile, the reader is cautiously reminded that getting a good grasp of the fundamental materials discussed in this chapter is essential for following the future details or extensions described in the remaining of the book. At the end of this chapter, the *equal-size packet protocol model* will also be introduced.

2.1 5-Layer TCP/IP Model

The basic structure of communication networks is represented by the *Transmission Control Protocol/Internet Protocol* (TCP/IP) model. This model is structured in five layers. An end system, an intermediate network node, or each communicating user or program is equipped with devices to run all or some portions of these layers, depending on where the system operates. These five layers, as shown in Figure 2.1, are as follows:

1. Physical layer
2. Link layer
3. Network layer
4. Transport layer
5. Application layer

Layer 1, the *physical layer*, defines electrical aspects of activating and maintaining physical links in networks. The physical layer represents the basic network hardware, such as switches and routers. The details of this layer are explained in later chapters, especially Chapters 3, 4, 6, 13, 15, 17, and 20.

Layer 2, the *link layer*, provides a reliable synchronization and transfer of information across the physical layer for accessing the transmission medium. Layer 2 specifies how packets access links and are attached to additional headers to form frames when entering a new networking environment, such as a LAN. Layer 2 also provides error detection and flow control. This layer is discussed further in Chapters 4, 5, 6, 19, and 20.

Layer 3, the *network layer* (IP) specifies the networking aspects. This layer handles the way that addresses are assigned to packets and the way that packets are supposed to be forwarded from one end point to another. Some related parts of this layer are described in Chapters 5, 6, 7, 10, 12, 14, 15, 16, 19, and 20.

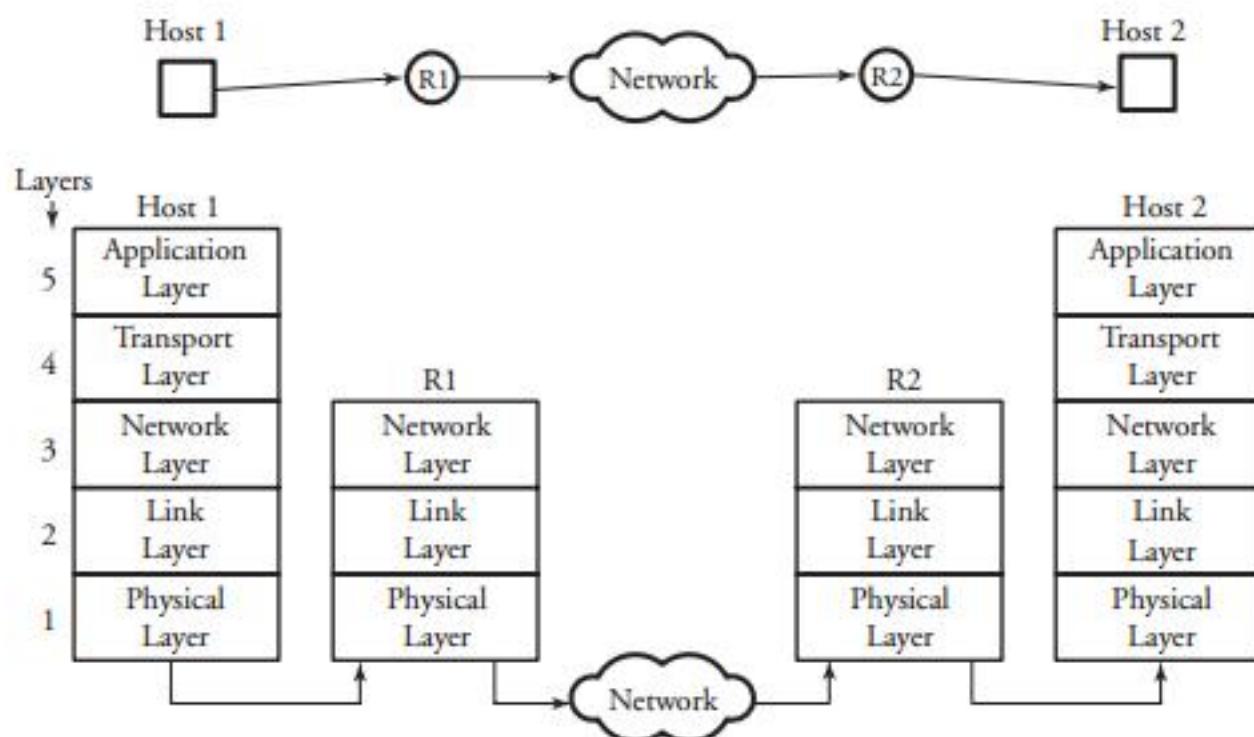


Figure 2.1 Hierarchy of 5-layer communication protocol model

Layer 4, the *transport layer*, lies just above the network layer and handles the details of data transmission. Layer 4 is implemented in the end-points but not in network routers and acts as an interface protocol between a communicating host and a network. Consequently, this layer provides logical communication between processes running on different hosts. The concepts of the transport layer are discussed further in Chapters 8, 9, 12, 16, 18, 19, and 20.

Layer 5, the *application layer*, determines how a specific user application should use a network. Among such applications are the *Simple Mail Transfer Protocol* (SMTP), *File Transfer Protocol* (FTP), and the *World Wide Web* (WWW). The details of layer 5 are described in Chapters 9, 10, 15, 16, 18, 19, and 20.

The transmission of a given message between two users is carried out by (1) flowing down the data through each and all layers of the transmitting end, (2) sending it to certain layers of protocols in the devices between two end points, and (3) when the message arrives at the other end, letting the data flow up through the layers of the receiving end until it reaches its destination. Figure 2.1 illustrates a scenario in which different layers of protocols are used to establish a connection. A message is transmitted from host 1 to host 2, and, as shown, all five layers of the protocol model participate in making this connection. The data being transmitted from host 1 is passed down

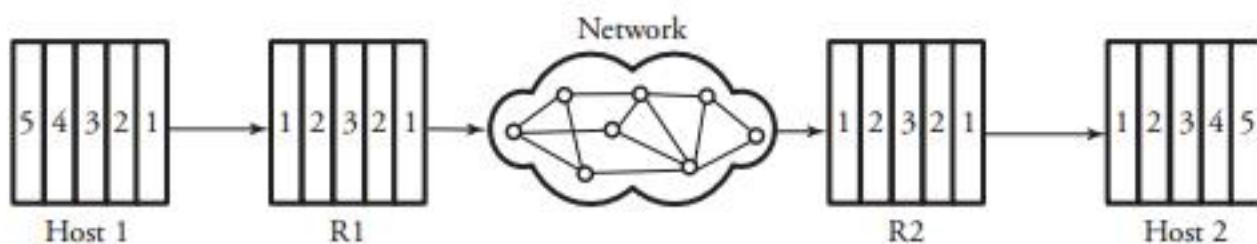


Figure 2.2 Structural view of protocol layers for two hosts communicating through two routers

through all five layers to reach router R1. Router R1 is located as a gateway to the operating regions of host 1 and therefore does not involve any tasks in layers 4 and 5. The same scenario is applied at the other end: router R2. Similarly, router R2, acting as a gateway to the operating regions of host 2, does not involve any tasks in layers 4 and 5. Finally at host 2, the data is transmitted upward from the physical layer to the application layer.

The main idea of the communication protocol stack is that the process of communication between two end points in a network can be partitioned into layers, with each layer adding its own set of special related functions. Figure 2.2 shows a different way of realizing protocol layers used for two hosts communicating through two routers. This figure illustrates a structural perspective of a communication set-up and identifies the order of fundamental protocol layers involved.

2.2 7-Layer OSI Model

The *open systems interconnection* (OSI) model was the original standard description for how messages should be transmitted between any two points. To the five TCP/IP layers, OSI adds the following two layers:

1. *Layer 6*, the *session layer*, which sets up and coordinates the applications at each end
2. *Layer 7*, the *presentation layer*, which is the operating system part that converts incoming and outgoing data from one presentation format to another

The tasks of these two additional layers are dissolved into the application and transport layers in the newer five-layer model. The OSI model is becoming less popular. TCP/IP is gaining more attention, owing to its stability and its ability to offer a better communication performance. Therefore, this book focuses on the five-layer model.

2.3 Internet Protocols and Addressing

The third layer of communication protocol hierarchy is the *network layer*, which specifies the networking aspects of a communication transaction. This *Internet Protocol (IP) layer* handles networking aspects and establishes routes for packets. The network layer, in fact, handles the method of assigning addresses to packets and determines how they should be forwarded from one end point to another.

The Internet Protocol produces a header for packets. An IP header contains the IP addresses of a source node and a destination node, respectively. An IP packet can be encapsulated in the layer 2 frames when the packet enters a LAN. The IP layer normally offers no QoS guarantees and provides a best-effort service. IP is inherently unreliable, relying on the higher layers, such as the transport protocol, to handle issues relating to system reliability.

IP provides seamless Internet connectivity and scalability. This layer is based on the *connectionless*, or so-called datagram switching, approach. The advantages of this kind of service are (1) flexibility to allow interconnection between diverse network topologies, and (2) robustness to node failure. Apart from the ability to connect diverse networks, the IP layer also fragments packets to the *maximum transmission unit (MTU)* and performs reassembly of packet fragments at destinations.

2.3.1 IP Packet

The packet format of IP version 4 (IPv4) is shown in Figure 2.3. Each packet comprises the header and data. The size of the header is variable, with 20 bytes of fixed-length header and an *options* field whose size is variable up to 40 bytes. A brief description of the fields follows.

- *Version* specifies the IP version.
- *Header length (HL)* specifies the length of the header.

Byte:

1	1	2	2	2	1	1	2	4	4	40	
Version	HL	Type of Service	Total Length	Identification	Flag	Frag. Offset	Time to Live	Protocol	Header Checksum	Source IP Address	Dest. IP Address
											Options
											Padding
											Data

Figure 2.3 IP packet format

- *Type of service* specifies the quality-of-service (QoS) requirements of the packet, such as priority level, delay, reliability, throughput, and cost.
- *Total length* specifies the total length of the packet in bytes, including the header and data. A total of 16 bits are assigned to this field.
- *Identification, flags, and fragment offset* are used for packet fragmentation and reassembly.
- *Time to live* specifies the maximum number of hops after which a packet must be discarded.
- *Protocol* specifies the protocol used at the destination.
- *Header checksum* is a method of error detection and is described in Chapter 4.
- *Source address* and *destination address* are 32-bit fields specifying the source address and the destination address, respectively.
- *Options* is a rarely used variable-length field to specify security level, timestamp, and type of route.
- *Padding* is used to ensure that the header is a multiple of 32 bits.

Recall that the 16 bits in the *total length* field express the total length of a packet. Hence, the total length of the packet is limited to 2^{16} bytes. However, the maximum packet size of 2^{16} bytes is rarely used, since the packet size is limited by the physical network capacity. The real physical network capacity per packet is normally less than 10K and even gets smaller, to 1.5K when the packet reaches a LAN. To accomplish packet partitioning, the *identification, flags, and fragment offset* fields perform and keep track of the packet-fragmentation process when needed.

2.3.2 IP Addressing Scheme

The IP header has 32 bits assigned for addressing a desired device in the network. An IP address is a unique identifier used to locate a device on the IP network. To make the system scalable, the address structure is subdivided into the *network ID* and the *host ID*. The *network ID* identifies the network the device belongs to; the *host ID* identifies the device. This implies that all devices belonging to the same network have a single network ID. Based on the bit positioning assigned to the *network ID* and the *host ID*, the IP address is further subdivided into classes A, B, C, D (multicast), and E (reserved), as shown in Figure 2.4.

Class A	0	1		7	8			31
	0		Net ID	7 bits			Host ID	
Class B	0	1	2			15	16	31
	1	0		Net ID	14 bits		Host ID	
Class C	0	1	2	3			23	24
	1	1	0		Net ID	21 bits		Host ID
Class D	1	1	1	0				31
	1	1	1	0	Multicast			
Class E	1	1	1	1				31
	1	1	1	1	Reserved for Experiment			

Figure 2.4 Classes of IP addresses

Consider the lengths of corresponding fields shown in this figure. Class A starts with a 0 and supports 126 networks and 16 million hosts per network. Class B addressing always starts with 10 and supports 16,382 networks and 65,534 hosts per network. Class C addressing starts with 110 and supports 2 million networks and 254 hosts per network. Class D addressing starts with 1110 and is specifically designed for multicasting and broadcasting. Class E always starts with 1111 reserved for network experiments. For ease of use, the IP address is represented in *dot-decimal* notation. The address is grouped into 4 dot-separated bytes.

Example. A host with an IP address of 10001000 11100101 11001001 00010000 belongs to class B, since it starts with 10, and its decimal equivalent is 136.229.201.16.

2.3.3 Subnet Addressing and Masking

The concept of subnetting was introduced to overcome the shortcomings of IP addressing. Managing the large number of hosts is an enormous task. For example, a company that uses a class B addressing scheme supports 65,534 hosts on one network. If the

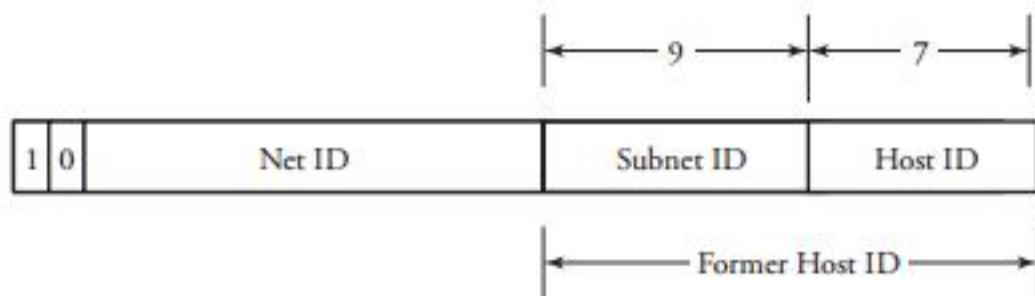


Figure 2.5 A subnet ID and host ID in class *B* addressing

company has more than one network, a multiple-network address scheme, or *subnet scheme*, is used. In this scheme, the host ID of the original IP address is subdivided into *subnet ID* and *host ID*, as shown in Figure 2.5.

Depending on the network size, different values of subnet ID and host ID can be chosen. Doing so would prevent the outside world from being burdened by a shortage of new network addresses. To determine the subnetting number, a subnet *mask*—logic AND function—is used. The subnet mask has a field of all 0s for the host ID and a field of all 1s for the remaining field.

Example. Given an IP address of 150.100.14.163 and a subnet mask of 255.255.255.128, determine the maximum number of hosts per subnet.

Solution. Figure 2.6 shows the details of the solution. Masking 255.255.255.128 on the IP address results in 150.100.14.128. Clearly, the IP address 150.100.14.163 is a class B address. In a class B address, the lower 16 bits are assigned to the subnet and host fields. Applying the mask, we see that the maximum number of hosts is $2^7 = 128$.

Example. A router attached to a network receives a packet with the destination IP address 190.155.16.16. The network is assigned an address of 190.155.0.0. Assume that the network has two subnets with addresses 190.155.16.0 and 190.155.15.0 and that both subnet ID fields have 8 bits. Explain the details of routing the packet.

Solution. When it receives the packet, the router determines to which subnet the packet needs to be routed, as follows: The destination IP address is 190.155.16.16, the subnet mask used in the router is 255.255.255.0, and the result is 190.155.16.0. The router looks up its routing table for the next subnet corresponding to the subnet 190.155.16.0, which is subnet 2. When the packet arrives at subnet 2, the router determines that the destination is on its own subnet and routes the packet to its destination.

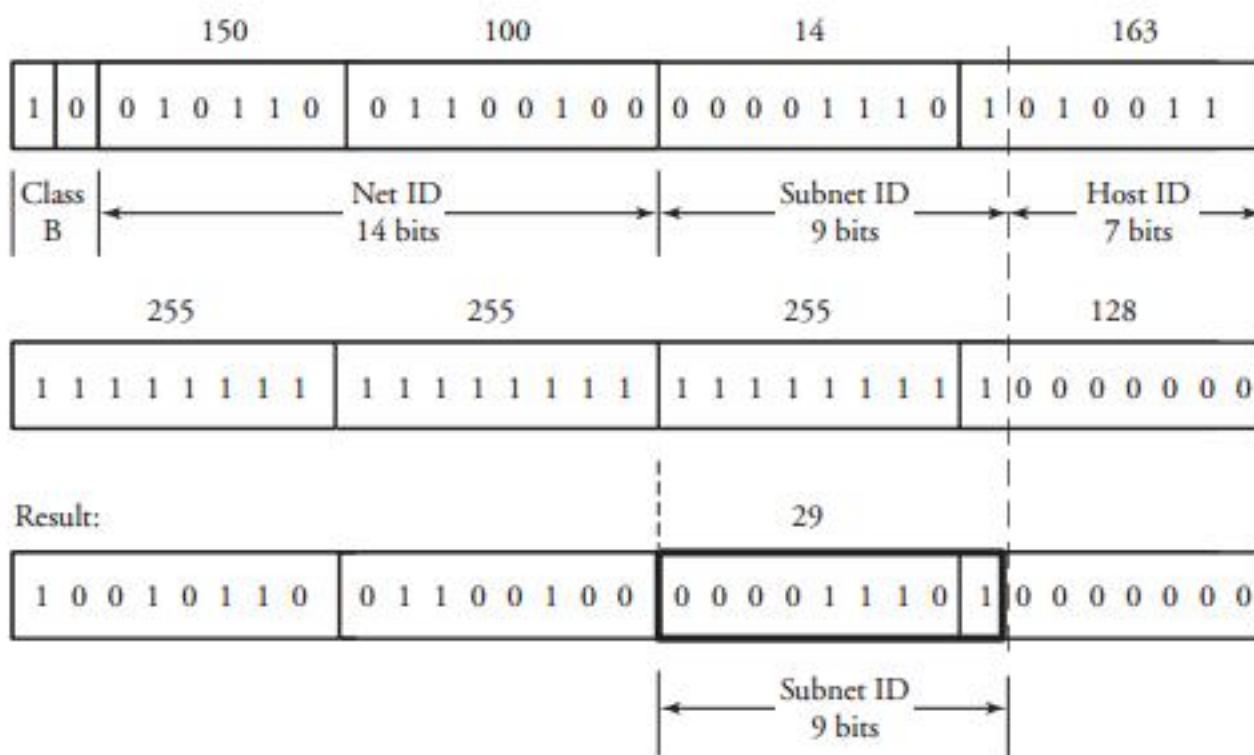


Figure 2.6 An example of subnet and masking

2.3.4 Classless Interdomain Routing (CIDR)

The preceding section described an addressing scheme requiring that the address space be subdivided into five classes. However, giving a certain class C address space to a certain university campus does not guarantee that all addresses within the space can be used and therefore might waste some addresses. This kind of situation is inflexible and would exhaust the IP address space. Thus, the classful addressing scheme consisting of classes A, B, C, D, and E results in an inefficient use of the address space.

A new scheme, with no restriction on the classes, emerged. *Classless interdomain routing* (CIDR) is extremely flexible, allowing a variable-length *prefix* to represent the network ID and the remaining bits of the 32-field address to represent the hosts within the network. For example, one organization may choose a 20-bit network ID, whereas another organization may choose a 21-bit network ID, with the first 20 bits of these two network IDs being identical. This means that the address space of one organization contains that of another one.

CIDR results in a significant increase in the speed of routers and has greatly reduced the size of routing tables. A routing table of a router using the CIDR address space has entries that include a pair of network IP addresses and the mask. *Supernetting* is a CIDR technique whereby a single routing entry is sufficient to represent a group of adjacent

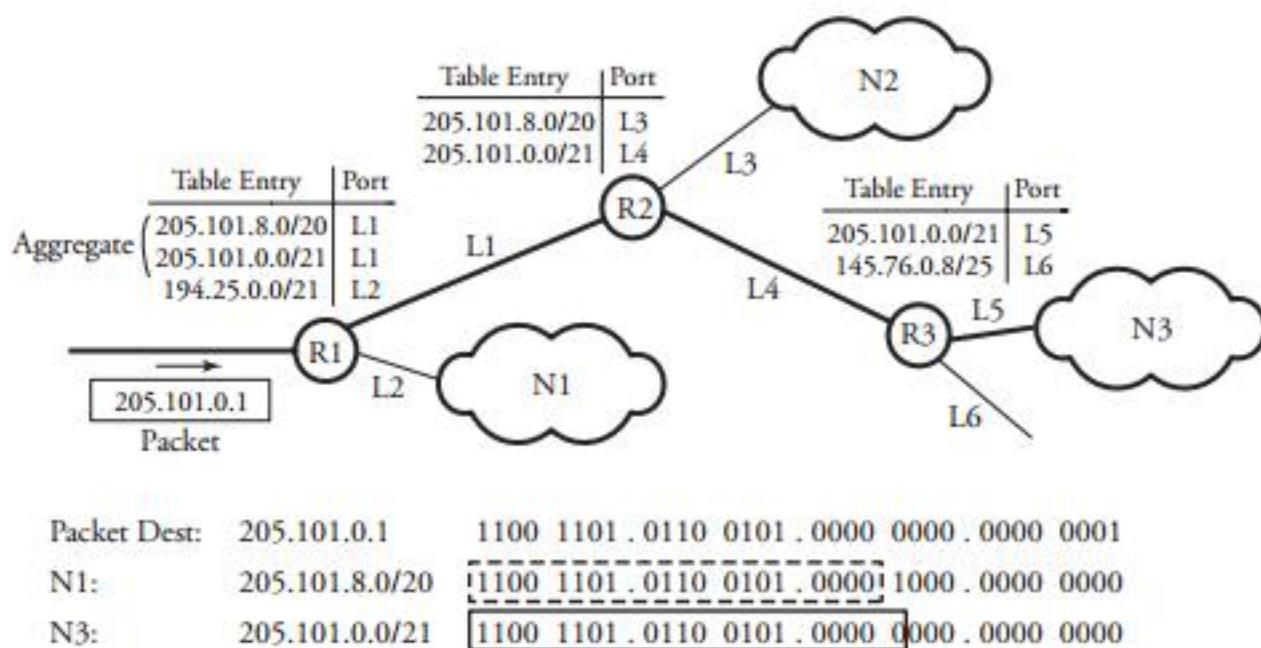


Figure 2.7 CIDR routing

addresses. Because of the use of a variable-length prefix, the routing table may have two entries with the same prefix. To route a packet that matches both of these entries, the router chooses between the two entries, using the longest-prefix-match technique.

Example. Assume that a packet with destination IP address 205.101.0.1 is received by router R1, as shown in Figure 2.7. In the entries of this router, two routes, L1 and L2, belonging to 205.101.8.0/20 and 205.101.0.0/21, respectively, are matched. CIDR dictates that the longer prefix be the eligible match. As indicated at the bottom of this figure, link L1, with its 21-bit prefix, is selected, owing to a longer match. This link eventually routes the packet to the destination network, N3.

CIDR allows us to reduce the number of entries in a router's table by using an *aggregate technique*, whereby all entries that have some common partial prefix can be combined into one entry. For example, in Figure 2.7, the two entries 205.101.8.0/20 and 205.101.0.0/21 can be combined into 205.101.0.0/19, saving one entry on the table. Combining entries in routing tables not only saves space but also enhances the speed of the routers, as each time, routers need to search among fewer addresses.

2.3.5 Packet Fragmentation and Reassembly

The physical capacity of networks enforces an upper bound on the size of packets. The *maximum transmission unit* (MTU) represents this restriction. For example, as a LAN

standard, Ethernet limits the size of flowing frames to be 1,500 bytes. The objective of inducing this method is that we need a mechanism that avoids requiring large buffers at intermediate routers to store the fragments. This restriction necessitates the Internet Protocol to break up large messages into fragments. The fragment sizes are limited to the MTU of the underlying physical network. The fragments could in turn be split into smaller fragments, depending on the physical network being used. Each fragment is routed independently through the network. Once all the fragments are received, they are reassembled at the final destination to form the original packet.

The identification, flag, and offset fields of the IP header help with the fragmentation and reassembly process. The identification field is used to distinguish between various fragments of different packets. The flag field has a more-fragment (MF) bit. When the MF bit is set, it implies that more fragments are on their way. The offset field indicates the position of a fragment in the sequence of fragments making up the packet. The lengths of all the fragments, with the exception of the last one, must be divisible by 8.

To be successfully reassembled, all fragments making up a packet must arrive at the destination. In the case of a missing fragment, the rest of the fragments have to be discarded, and thus the packet needs to be retransmitted. In such cases, the retransmission of packets results in an inefficient use of the network bandwidth.

Example. Suppose that a host application needs to transmit a packet of 3,500 bytes. The physical layer has an MTU of 1,500 bytes. The packet has an IP header of 20 bytes plus another attached header of 20 bytes. Fragment the packet, and specify the ID, MF, and offset fields of all fragments.

Solution. The allowable data length = $1,500 - 20 - 20 = 1,460$ bytes. Because 1,460 is not divisible by 8, the allowable data length is limited to 1,456 bytes. Including the headers, the data to be transmitted is then 3,540 bytes to be split into fragments of 1,456, 1,456 and 628 bytes. Here, fragment 1 = total length 1,456, MF 1, offset 0; fragment 2 = total length 1,456, MF 1, offset 182; and fragment 3 = total length 628, MF 0, and offset 364.

2.3.6 Internet Control Message Protocol (ICMP)

In connectionless routing, routers operate autonomously. They forward and deliver packets without requiring any coordination with the source. In large communication networks, IP may not be able to deliver a packet to its destination, owing to possible failures in the connectivity of a destination. Besides the hardware failure, other factors

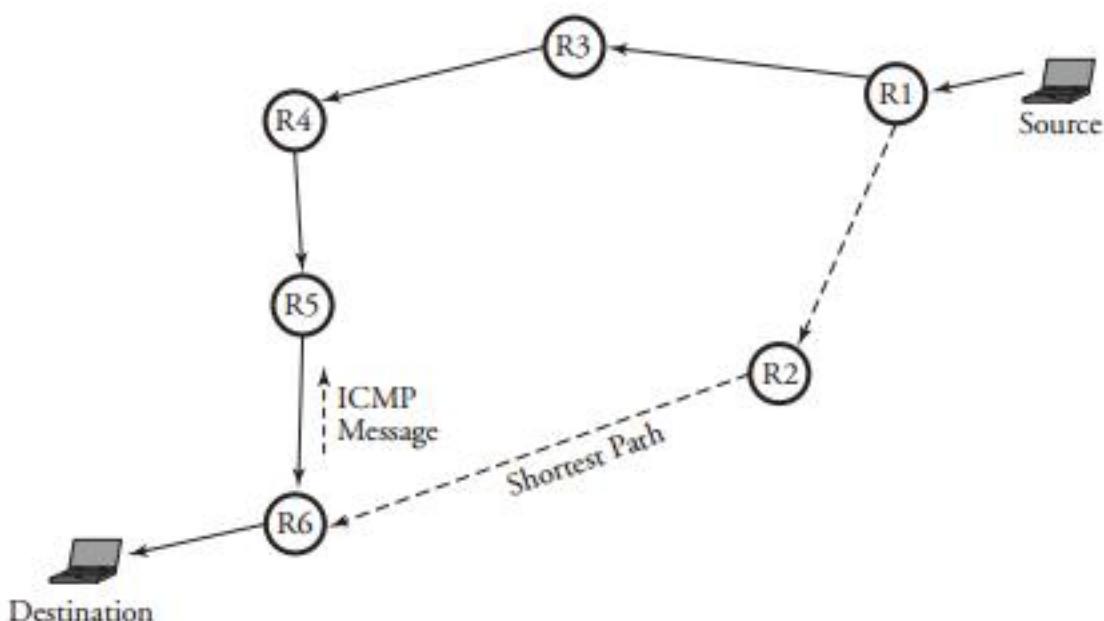


Figure 2.8 With ICMP, a redirect message cannot be sent to R1, since R6 does not know the address of R1.

may be present to create this problem. For example, as noted in Section 2.3.1, the *time-to-live* field in an IP packet specifies the maximum number of hops after which a packet must be discarded. If the counter of this field expires, packet delivery too can become impossible.

Another issue—related and equally important—is that a sender cannot know whether a delivery failure is a result of a local or a remote technical difficulty. With TCP/IP, routers in a network can report errors through the *Internet Control Message Protocol* (ICMP). An ICMP message is encapsulated in the data portion of an IP datagram (packet). When an error occurs, ICMP reports it to the originating source of the connection. This is compatible with the fact that an IP datagram header itself specifies only the original source and not any routers. The source must interpret the error.

One of the important ICMP messages is the *redirect* message. In Figure 2.8, a source tries to send a message to a destination. But R1 incorrectly sends the message to a wrong path (R1-R3-R4-R5-R6) instead of to the short one (R1-R2-R6). In this case, if in the middle of routing, R5 or R6 finds out about this error, it cannot issue an ICMP message to R1 to correct the routing, as they do not know the address of R1. Instead, they issue a redirect ICMP message to the source.

2.3.7 IP Version 6 (IPv6)

The use of IPv4 has resulted in the exhaustion of the 32-bit address space to the extent that IPv4 has run out of addressing spaces. Therefore, 128-bit address spacing was

Byte:

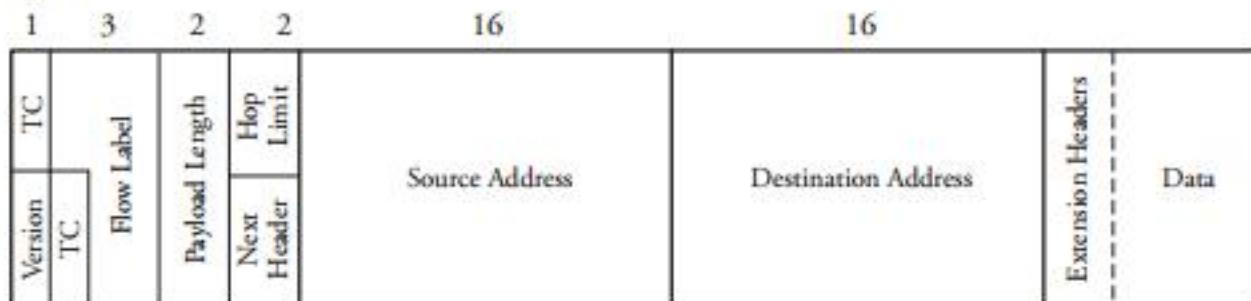


Figure 2.9 An IPv6 packet format

introduced with *Internet Protocol version 6* (IPv6). It enjoys tremendous popularity because of its simplicity and flexibility in adapting to diverse network technologies. Compatible with IPv4, IPv6 also supports real-time applications, including those that require guaranteed QoS. Figure 2.9 shows the IPv6 header. A brief description of the fields in the header follows.

- *Version* is the same as in IPv4, indicating the version number of the protocol.
- *Traffic class* specifies the priority level assigned to a packet.
- *Flow label* indicates the delay period within which application packets, such as real-time video, must be delivered.
- *Payload length* is the 16-bit specification of the length of the data, excluding the header.
- *Next header* specifies the type of extension header used. The functionality of the option field in IPv4 is specified in the extension header. In addition, the extension header is more flexible than the options field.
- *Hop limit* is the same as the time-to-live field in IPv4.
- *Source address* and *destination address* are each identified by a 128-bit field address.

The IPv4 and IPv6 header formats have some notable differences. First, IPv6 uses a 128-bit address field rather than the 32-bit field in IPv4. The 128-bit field can support a maximum of 3.4×10^{38} IP addresses. IPv6 has a simpler header format, eliminating the fragmentation, the checksum, and header length fields. The removal of the checksum field in IPv6 allows for faster processing at the routers without sacrificing functionality. In IPv6, *error detection* and *correction* are handled at the data link and the TCP layers. Note also that IPv6 can accommodate the QoS requirements for some applications. Besides all these significant advantages, IPv6 can provide built-in security features such as confidentiality and authentication. These features are discussed in Chapter 10.

IPv6 Addressing Format

With its large address spacing, IPv6 network addressing is very flexible. To efficiently represent the 128-bit address of IPv6 in a compact form, hexadecimal digits are used. A colon separates each of the four hexadecimal digits. For example, [2FB4 : 10AB : 4123 : CEBF : 54CD : 3912 : AE7B : 0932] can be a source address. In practice, IPv6 addresses contain a lot of bits that are zero. The address is commonly denoted in a more compact form. For example, an address denoted by [2FB4 : 0000 : 0000 : 0000 : 54CD : 3912 : 000B : 0932] can be compressed to [2FB4 :: : 54CD : 3912 : B : 932].

The network address space is classified into various types, each of which is assigned a binary prefix. Currently, only a small portion of the address space has been assigned, with the remaining reserved for future use. One of the address types with a leading byte of 1s is assigned for multicast; the rest of the currently assigned types are used for unicast applications. Apart from the unicast and multicast addresses, IPv6 introduces *anycast* addresses. An anycast address is similar to a multicast address and identifies a group of network devices for making connections. However, unlike with multicast addressing, a packet needs to be forwarded to any one device in the group. Anycast addresses share the address space with unicast address types. IPv6 reserves some addresses for special purposes.

Extension Header

Extension headers are positioned between the header and the payload. If multiple extension headers are used, they are concatenated, as shown in Figure 2.10, making it mandatory for them to be processed in the sequence in which they are listed. Figure 2.10 specifies the sequence in which the extension headers are to be listed.

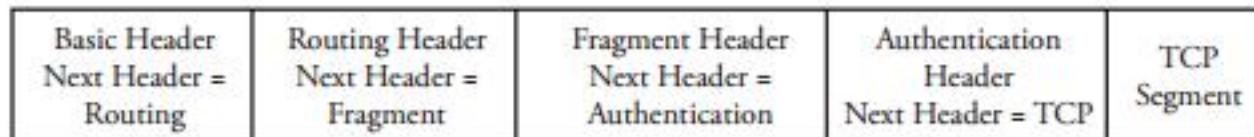
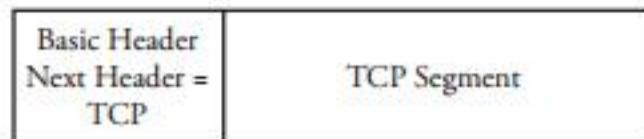


Figure 2.10 Concatenated IPv6 extension header

Packet Fragmentation

In IPv6, fragmentation is permitted only at the source. The result of this restriction is faster processing of packets at routers. Before transmitting a packet, a host performs a *maximum transmission unit* (MTU) discovery in the route of the packet. The minimum MTU obtained determines the packet size and thus requires the route from the host to the destination to remain steady. If this minimum value of the physical network is less than the packet size to be transmitted, the intermediate router discards the packet and sends an error message back to the source. In rare cases, the packet needs to be fragmented, and the extension header contains the fragmentation information.

2.4 Equal-Sized Packets Model: ATM

A networking model in which packets are of equal size can be constructed. Equal-sized packets, or *cells*, bring a tremendous amount of simplicity in the networking hardware, since buffering, multiplexing, and switching of cells become extremely simple. However, a disadvantage of this kind of networking is the typically high overall ratio of header to data. This issue normally arises when the message size is large and the standard size of packets is small. As discussed in Section 1.3, the dominance of headers in a network can cause delay and congestion. Here, we describe *Asynchronous Transfer Mode* technology as an example of this model.

The objective of *Asynchronous Transfer Mode* (ATM) technology is to provide a homogeneous backbone network in which all types of traffic are transported with the same small fixed-sized *cells*. One of the key advantages of ATM systems is flexible multiplexing to support multiple forms of data. ATM typically supports such *bursty* sources as FAX, coded video, and bulk data. Regardless of traffic types and the speed of sources, the traffic is converted into 53-byte ATM cells. Each cell has a 48-byte data payload and a 5-byte header. The header identifies the virtual channel to which the cell belongs.

Similar to a telephone network, ATM is a set of connection-oriented protocols, which means that a connection must be preestablished between two systems in a network before any data can be transmitted. ATM is capable of supporting and integrating data, voice, and video over one transmission medium with high bit data rate delivery services into a single network. ATM is bandwidth-on-demand networking and is scalable in bandwidth with the ability to support real multimedia applications.

The use of fixed-size cells can greatly reduce the overhead of processing ATM cells at the buffering and switching stages and hence increase the speed of routing,

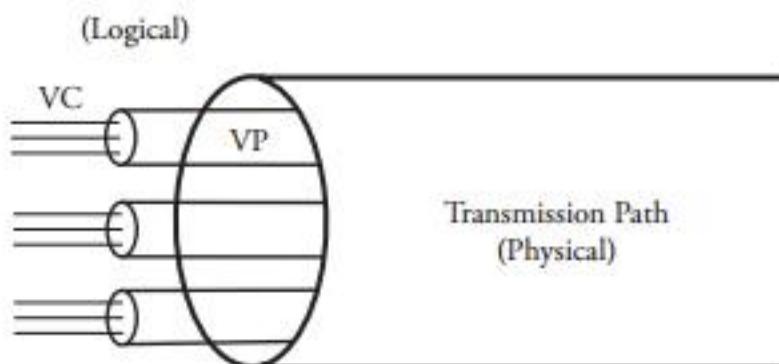


Figure 2.11 Overview of a typical ATM transmission medium

switching, and multiplexing functions. However, the great ratio of header to data makes the technology unsuitable for wide area networks and limits its applications in small networks. Like IPv6, ATM supports QoS mainly to reserve resources that guarantee specified maximum delay, minimum throughput, and maximum data loss. The QoS support allows ATM to concurrently handle all kinds of traffic.

ATM connections are identified by a *virtual channel identifier* (VCI) and a *virtual path identifier* (VPI). VCI and VPI are combined to be used in a switch to route a cell. As shown in Figure 2.11, the identity of a “physical” link is identified by two “logical” links: virtual channel (VC) and virtual path (VP). When a connection is set up, the values of these identifiers remain unchanged for the lifetime of the ATM connection.

Example. Figure 2.12 shows a routing table in an ATM switch, with routing information for all active connections passing through the switch. The routing information consists of the new VPI/VCI and new outgoing link for every incoming VC. Link 5, with VPIs 1, 3, and 5, can be switched on link 10 with VPIs 3, 7, and 8 through the ATM switch. A routing table provides the detail of the switching function. For example, a cell with VPI 3 and VCI 9 on link 5 is set to be forwarded with VPI 7 and VCI 2 on link 10.

2.4.1 ATM Protocol Structure

The ATM protocol structure is shown in Figure 2.13. The three-dimensional model includes four layers in the vertical dimension. The tightly linked layers consist of the *physical layer*, the *ATM layer*, the *ATM adaptation layer* (AAL), and *higher layers*. The physical layer includes two sublayers: the *physical medium* and *transmission convergence*. The physical medium sublayer defines the physical and electrical/optical interfaces with the transmission media on both the transmitter and the receiver. This layer also provides

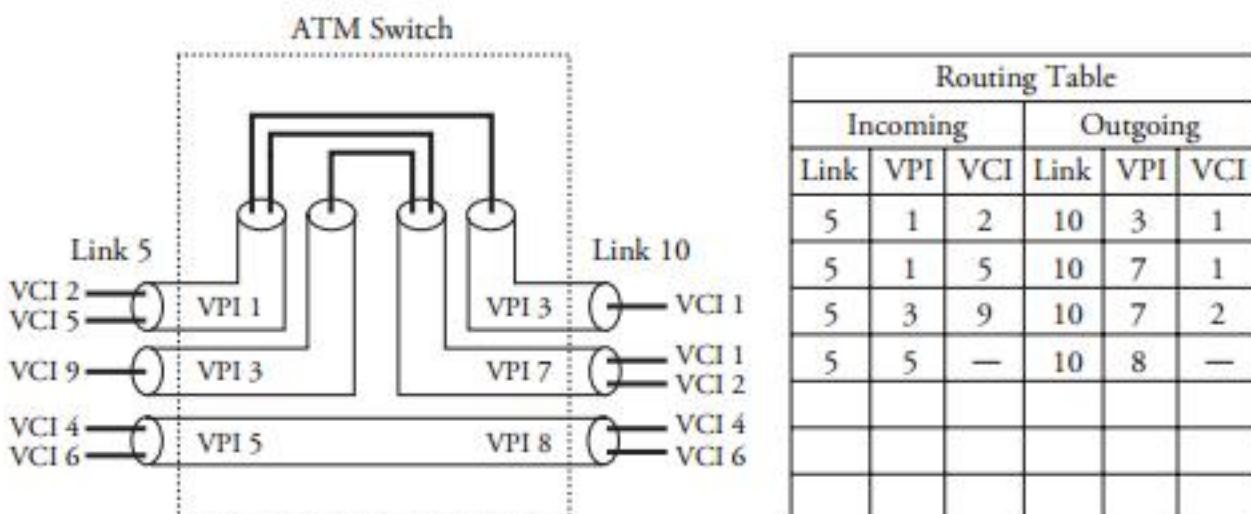


Figure 2.12 A routing table in an ATM switch

timing information and line coding. The transmission convergence sublayer provides frame adaptation and frame generation/recovery.

The ATM layer provides services, including cell multiplexing and demultiplexing, generic flow control, header cell check generation and extraction, and most important, remapping of VPIs and VCIs. The AAL layer maps higher-layer service data units, which are fragmented into fixed-size cells to be delivered over the ATM interface. In addition, this layer collects and reassembles ATM cells into service data units for transporting to higher layers. The four types of AALs support different classes of services.

1. AAL1 supports class A traffic, the required timing between a transmitter and a receiver, and the *constant bit rate* (CBR) traffic.
2. AAL2 supports class B traffic and time-sensitive—between source and sink—but *variable bit rate* (VBR) data traffic.
3. AAL3/4 supports class C or class D traffic and VBR data traffic.
4. AAL5 supports class D traffic in which VBR traffic can be transported and no timing relationship between source and sink is required.

The higher layers incorporate some of the functionality of layers 3 through 5 of the TCP/IP model. The control plane at the top of the cube shown in Figure 2.13 involves all kinds of network signaling and control. The *user plane* involves the transfer of user information, such as the flow-control and error-control mechanisms. The *management plane* provides management function and an information-exchange function between the *user plane* and the *control plane*. The management plane includes

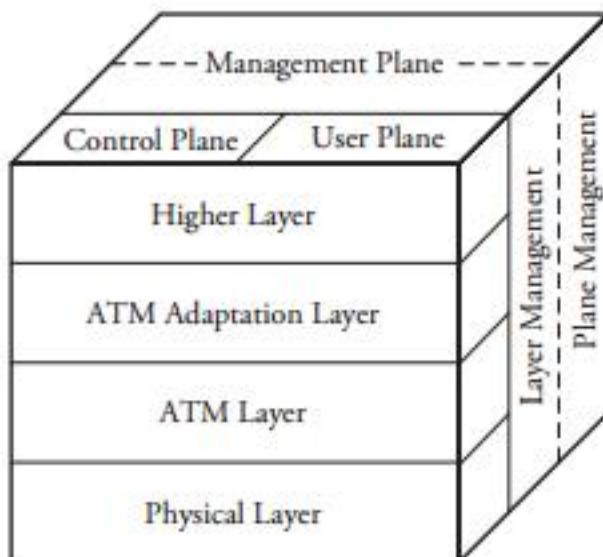


Figure 2.13 ATM protocol reference model

(1) plane management that performs management and coordination functions related to a system as a whole, and (2) the layer management that monitors bit error rates on a physical communications medium.

An ATM network can support both a *user-network interface* (UNI) and a *network-node interface* (NNI). A UNI is an interface connection between a terminal and an ATM switch, whereas an NNI connects two ATM switches. A summary of these two interfaces is shown in Figure 2.14. If privately owned switches are in the network, the interface between the public and private parts of the network is called the public NNI, and the connection between two private switches is known as the private NNI (P-NNI).

2.4.2 ATM Cell Structure

An ATM cell has two parts: a 48-byte payload and a 5-byte header, as shown in Figure 2.15. The choice of a 48-byte payload was a compromise among various design teams considering two important factors: packetization delay and transmission efficiency. (Refer back to Section 1.3 on packet size optimization.) The header consists of several fields. However, the ATM cell header has two different formats: UNI and NNI. The details of the UNI 5-byte header are as follows.

- The 4-bit *generic flow control* (GFC) field is used in UNI only for controlling local flow control. This field enables the participating equipment to regulate the flow of traffic for different grades of service. Two modes are defined for this field: the

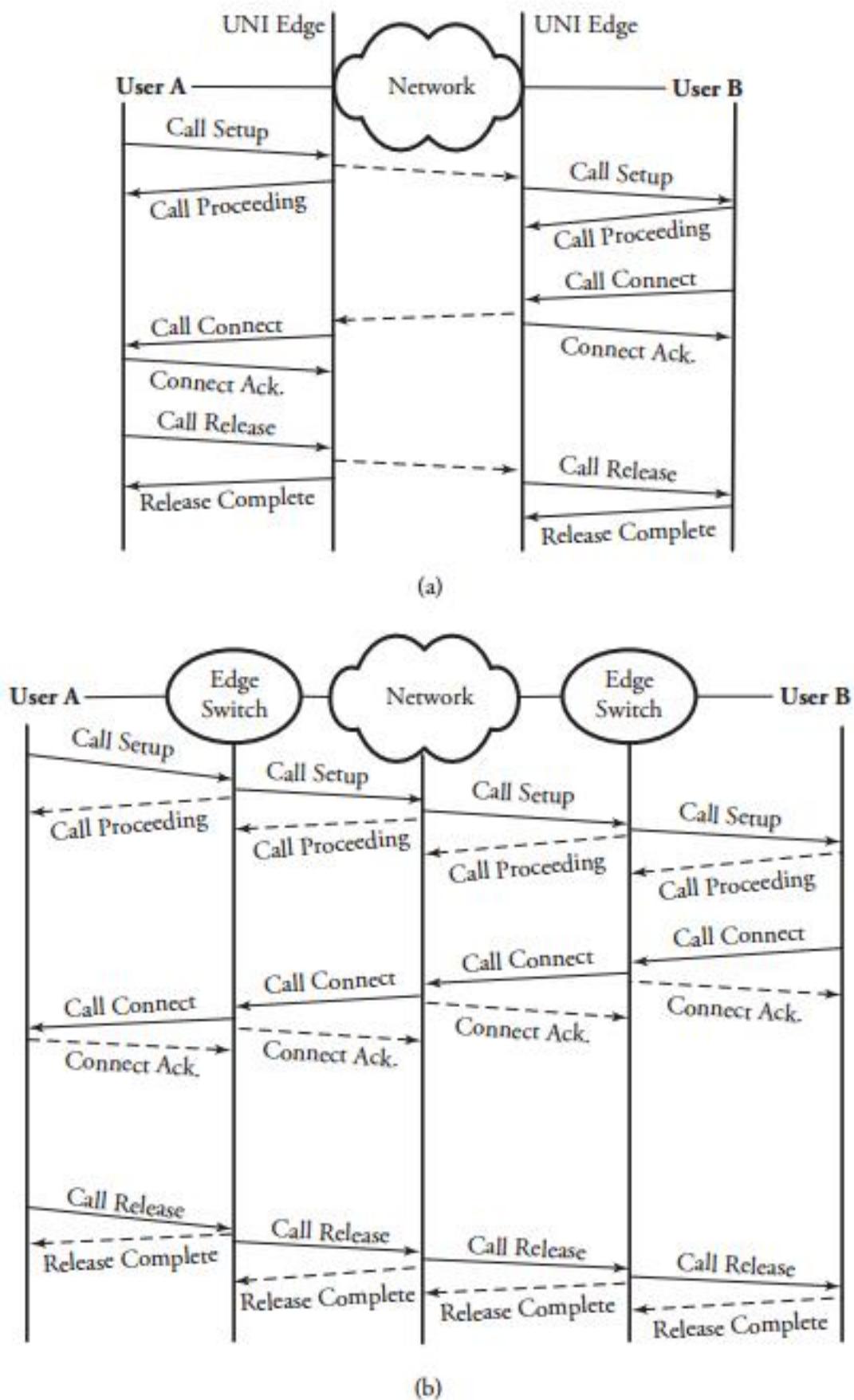


Figure 2.14 Overview of signaling: (a) UNI format; (b) NNI format

Byte:

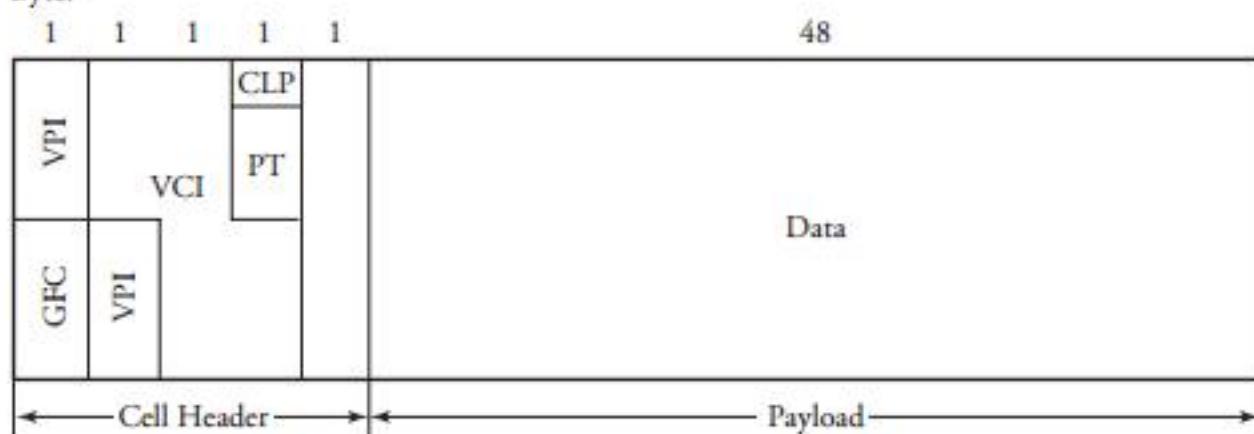


Figure 2.15 An ATM cell and its header structure

controlled GFC, to provide flow control between a user and a network, and the *uncontrolled* GFC, to indicate that the GFC function is not used.

- Together, the *virtual path identifier* and the *virtual channel identifier* represent an ATM address. A VPI identifies a group of virtual channels with the same end point. A VCI identifies a virtual channel within a virtual path.
- The 3-bit *payload type* field is used to indicate the type of data located in the payload portion of the cell: for example, 000, the current cell is a data cell and no congestion is reported; 010, this is a user data cell, and congestion is experienced. The payload could be congestion information, network management message, signaling information, or other forms of data.
- The 1-bit *cell-loss priority* (CLP) field is used to prioritize cells. When congestion occurs, cells with CLP set to 1 (considered low priority) are discarded first. If the bit is set to 0, the cell gets higher priority and should be discarded only if it could not be delivered.
- The 8-bit *header error control* (HEC) field is used for error checking. HEC functions include correcting single-bit errors and detecting multiple-bit errors.

The main difference between NNI and UNI formats is that the 4 bits used for the GFC field in the UNI cell header are added to the VPI field in the NNI cell header. Thus, for NNI, VPI is 12 bits, allowing for more VPs to be supported within the network. VCI is 16 bits for both cases of UNI and NNI. The values of VPI and VCI have local significance only with a transmission link. Each switching node maps an incoming VPI/VCI to an outgoing VPI/VCI, based on the connection setup or routing table, as shown in Figure 2.12.

HEC works like other checking methods. First, the transmitter calculates the HEC field value, and the receiver side runs an algorithm consisting of two modes of operation. At initialization step, this field starts with an error-correction mode. If a single-bit error is detected in the header, the error-correction algorithm identifies the error bit and then corrects it. If a multibit error is detected, the mode moves to detection mode; errors are discarded but not corrected. Error-detection mode remains whenever cells are received in error, moving back to correction mode only when cells are received without error.

2.5 Summary

This chapter covered a tremendous amount of fundamental networking protocol material. We presented the basic structure of the Internet network protocols and an overview of the TCP/IP layered architecture. This architectural model provides a communication service for peers running on different machines and exchanging messages.

We also covered the basics of protocol layers: the *network layer* and the structure of IPv4 and IPv6. IP addressing is further subdivided as either *classful* or *classless*. Classless addressing is more practical for managing routing tables. Finally, we compared the equal-sized packet networking environment to IP networks. Although packet multiplexing is easy, the traffic management is quite challenging.

The next chapter presents another fundamental discussion in networking. Chapter 3 focuses on the fundamental operations of networking devices.

2.6 Exercises

1. Specify the class of address and the subnet ID for the following cases:
 - (a) A packet with IP address 127.156.28.31 using mask pattern 255.255.255.0
 - (b) A packet with IP address 150.156.23.14 using mask pattern 255.255.255.128
 - (c) A packet with IP address 150.18.23.101 using mask pattern 255.255.255.128
2. Specify the class of address and the subnet ID for the following cases:
 - (a) A packet with IP address 173.168.28.45 using mask pattern 255.255.255.0
 - (b) A packet with IP address 188.145.23.1 using mask pattern 255.255.255.128
 - (c) A packet with IP address 139.189.91.190 using mask pattern 255.255.255.128
3. Apply CIDR aggregation on the following IP addresses: 150.97.28.0/24, 150.97.29.0/24, and 150.97.30.0/24.

4. Apply CIDR aggregation on the following IP addresses: 141.33.11.0/22, 141.33.12.0/22, and 141.33.13.0/22.
5. Use the subnet mask 255.255.254.0 on the following IP addresses, and then convert them to CIDR forms:
 - (a) 191.168.6.0
 - (b) 173.168.28.45
 - (c) 139.189.91.190
6. A packet with the destination IP address 180.19.18.3 arrives at a router. The router uses CIDR protocols, and its table contains three entries referring to the following connected networks: 180.19.0.0/18, 180.19.3.0/22, and 180.19.16.0/20, respectively.
 - (a) From the information in the table, identify the exact network ID of each network in binary form.
 - (b) Find the right entry that is a match with the packet.
7. Part of a networking infrastructure consists of three routers R1, R2, and R3 and six networks N1 through N6, as shown in Figure 2.16. All address entries of each router are also given as seen in the figure. A packet with the destination IP address 195.25.17.3 arrives at router R1:
 - (a) Find the exact network ID field of each network in binary form.
 - (b) Find the destination network for packet (proof needed).
 - (c) Specify how many hosts can be addressed in network N1.

R1 Table Entry	Port	R2 Table Entry	Port	R3 Table Entry	Port
195.25.0.0/21	L11	195.25.24.0/19	L21	111.5.0.0/21	L31
195.25.16.0/20	L12	195.25.16.0/20	L22	Else	L32
195.25.8.0/22	L13	195.25.8.0/22	L23	195.25.16.0/20	L33
135.11.2.0/22	L14				

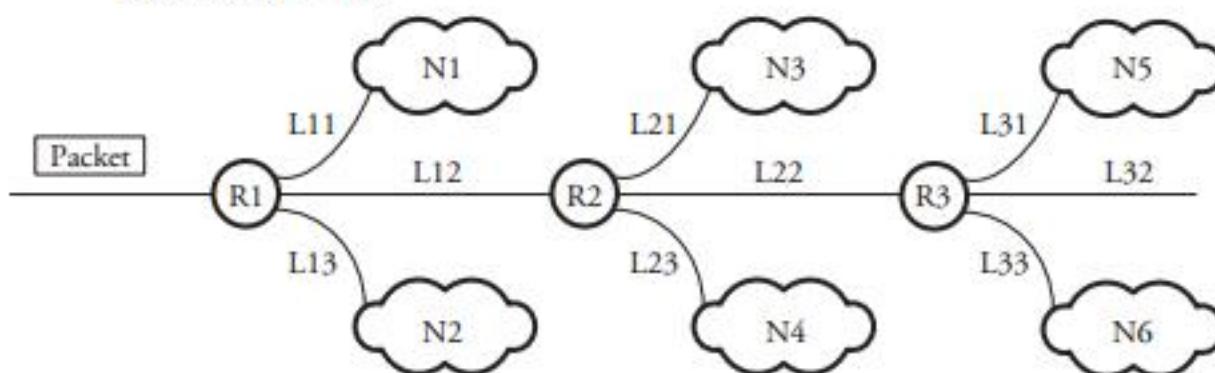


Figure 2.16 Exercise 7 network example

8. Consider an estimated population of 620 million people.
 - (a) What is the maximum number of IP addresses that can be assigned per person using IPv4?
 - (b) Design an appropriate CIDR to deliver the addressing in part (a).
 - (c) What is the maximum number of IP addresses that can be assigned per person using IPv6?
9. For each of the following IPv6 addresses, give an abbreviated form and then convert the result to a binary form:
 - (a) 1111:2A52:A123:0111:73C2:A123:56F4:1B3C
 - (b) 2532:0000:0000:0000:FB58:909A:ABCD:0010
 - (c) 2222:3333:AB01:1010:CD78:290B:0000:1111
10. Research why IPv6 allows fragmentation only at the source.
11. Suppose that virtual paths are set up between every pair of nodes in an ATM network. Explain why connection setup can be greatly simplified in this case.
12. Suppose that the ATM network concept is generalized so that packets can be variable in length. What features of ATM networking are retained? What features are lost?

This page intentionally left blank

CHAPTER 3

Networking Devices

This chapter focuses on networking devices. Familiarity with networking hardware devices is essential for understanding how a local area or a wide area network operates. This chapter covers the following aspects of network component functionality.

- *Multiplexers*
- *Modems and Internet access devices*
- *Switching and routing devices*
- *Router structure*

The three main categories of networking devices are *multiplexers*, *modems*, and *switching devices*. We start with the architecture of *multiplexers*, introducing types of multiplexers and some useful analytical methods of multiplexers. Next, we consider networking modems for accessing the Internet from remote and residential areas. Then, we analyze switching devices, such as *repeaters*, *bridges*, and *routers*, by which packets are switched from one path to another.

3.1 Multiplexers

Multiplexers are used in a network for maximum transmission capacity of a high-bandwidth line. Regardless of the type of multiplexer, multiplexing is a technique that allows many communication sources to transmit data over a single physical line.

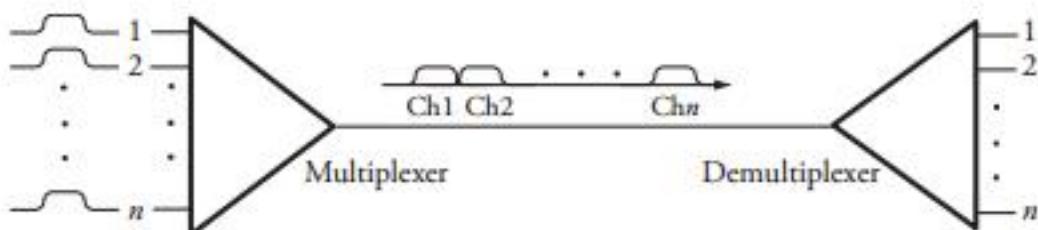


Figure 3.1 A frequency-division multiplexer (FDM) with n inputs

Multiplexing schemes can be divided into three basic categories: *frequency-division multiplexing*, *wavelength-division multiplexing*, and *time-division multiplexing*.

3.1.1 Frequency-Division Multiplexing (FDM)

In *frequency-division multiplexing* (FDM), the frequency spectrum is divided into frequency bands, or *channels*, in which each user can be assigned a band. Figure 3.1 shows how n frequency channels are multiplexed using FDM. When many channels are multiplexed together, a certain guard band is allocated to keep the channels well separated.

To implement a multiplexer, the original frequencies at any of n inputs of the multiplexer are raised, each by a different constant amount. Then, the n new frequency bands are combined to let no two channels occupy the same portion of the spectrum. Despite the guard bands between the channels, any two adjacent channels have some overlap because channel spectra do not have sharp edges. This overlap normally creates spike noise at the edge of each channel. FDM is normally used over copper wires or microwave channels and is suitable for analog circuitry.

3.1.2 Wavelength-Division Multiplexing (WDM)

Wavelength-division multiplexing (WDM) is fundamentally the same as FDM, as depicted in Figure 3.2. WDM was invented as a variation of frequency-division multiplexing and is basically a multiplexing method of different wavelengths instead of frequencies. In the figure, n optical fibers come together at an optical multiplexer, each with its energy present at a different wavelength. The n optic lines are combined onto a single shared link for transmission to a distant destination. At the demultiplexer, each frame, including n channels, is split up over as many optical fibers as there were on the input side. At each output of the demultiplexer, a tuned filter refines the desired signal at the tuned wavelength, and thus all other wavelengths are bypassed.

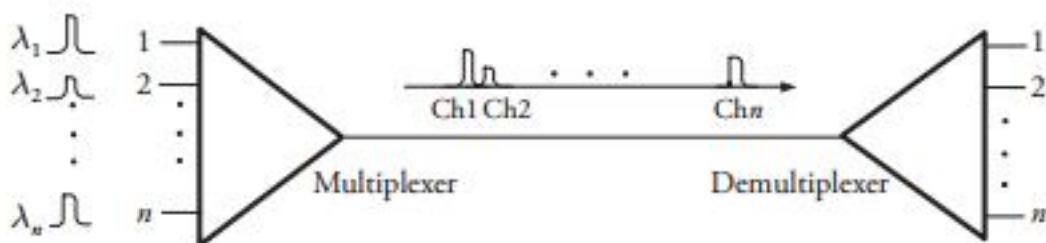


Figure 3.2 A wavelength-division multiplexer (WDM) with n inputs

The main issue of WDM compared with FDM is that an optical system using a diffraction grating is completely passive and thus highly reliable. With higher-speed variations of WDM, the number of channels is very large, and the wavelengths are as close as 0.1 nm. Such systems are referred to as DWDM (dense WDM). Chapter 14 presents much more detail about WDM and its applications.

Example. Consider a practical multiplexing system with 100 channels, each at rate 10 Gb/s. Compute the number of full-length movies per second that can be transferred with this WDM system.

Solution. The total bit rate of this WDM is 100×10 , or 1,000 Gb/s. Since a movie (MPEG-2 technology) requires 32 Gb/s bandwidth, the system can carry approximately 34 full-length movies per second.

3.1.3 Time-Division Multiplexing

With a *time-division multiplexing* (TDM), users take turns in a predefined fashion, each one periodically getting the entire bandwidth for a portion of the total scanning time. Given n inputs, time is divided into frames, and each frame is further subdivided into time slots, or channels. Each channel is allocated to one input. (See Figure 3.3.) This type of multiplexing can be used only for digital data. Packets arrive on n lines, and the multiplexer scans them, forming a frame with n channels on its outgoing link. In practice, packet size is variable. Thus, to multiplex variable-sized packets, additional hardware is needed for efficient scanning and synchronization. TDM can be either *synchronous* or *statistical*.

Synchronous TDM

In *synchronous* TDM, the multiplexer scans all lines without exception. The scanning time for each line is preallocated; as long as this time for a particular line is not altered by the system control, the scanner should stay on that line, whether or not there is

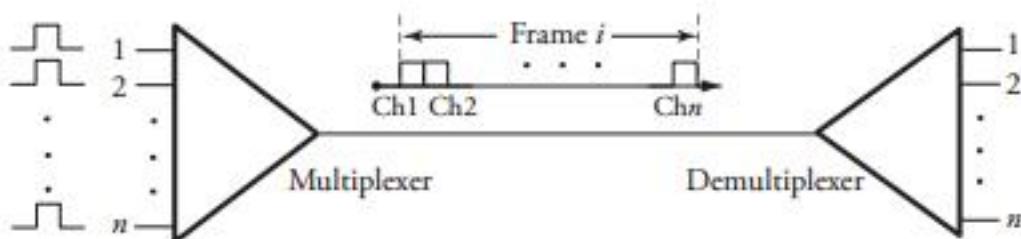
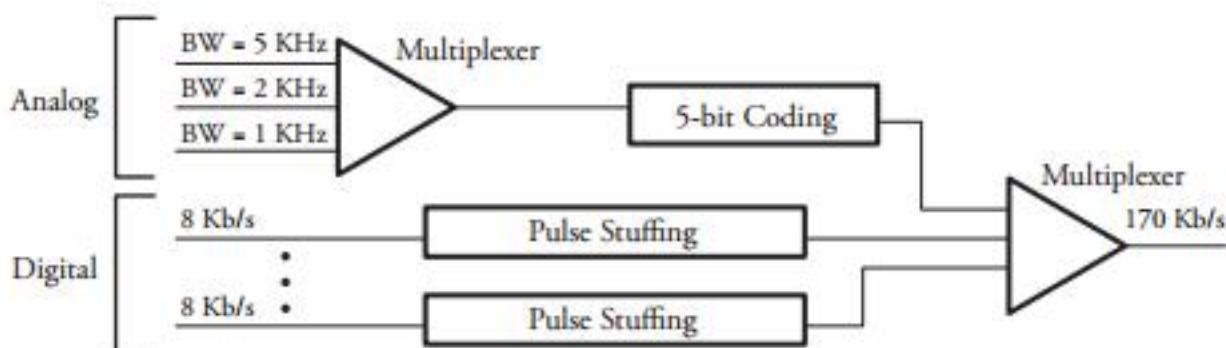
Figure 3.3 A time-division multiplexer with n inputs

Figure 3.4 Integrated multiplexing on analog and digital signals

data for scanning within that time slot. Therefore, a synchronous multiplexer does not operate efficiently, though its complexity stays low.

Once a synchronous multiplexer is programmed to produce same-sized frames, the lack of data in any channel potentially creates changes to average bit rate on the ongoing link. In addition to this issue, the bit rates of analog and digital data being combined in a multiplexer sometimes need to be synchronized. In such cases, dummy pulses can be added to an input line with bit rate shortcoming. This technique of bit-rate synchronization is called *pulse stuffing*.

Example. Consider the integration of three analog sources and four identical digital sources through a time-division multiplexer, as shown in Figure 3.4. We would like to use the entire 240 Kb/s maximum capacity of this multiplexer. The analog lines with bandwidths 5 KHz, 2 KHz, and 1 KHz, respectively, are sampled, multiplexed, quantized, and 5-bit encoded. The digital lines are multiplexed, and each carries 8 Kb/s. Find the pulse-stuffing rate.

Solution. Each analog input is sampled by a frequency two times greater than its corresponding bandwidth according to the Nyquist sampling rule. Therefore, we have $5 \times 2 + 2 \times 2 + 1 \times 2 = 16$ K samples per second. Once it is encoded, we get a

total $16,000 \times 5 = 80$ Kb/s on analog lines. The total share of digital lines is then $170 - 80 = 90$ Kb/s. Since each digital line must generate 22.5 Kb/s while the actual rate of 8 Kb/s exists, each digital line must add a difference of 14.5 Kb/s pulse stuffing in order to balance the ultimate bit rate of multiplexer bit rate.

Consider a multiplexer with n available channels. If the number of requesting input sources, m , is greater than n channels, the multiplexer typically reacts by *blocking* where unassigned sources are not transmitted and therefore remain inactive. Let t_a and t_d be the two mean times during which a given input becomes active and idle, respectively. Assume that the transmission line has n channels available the transmission line but that multiplexer m has inputs, where $m > n$.

If more than n inputs are active, we can choose only n out of m active sources and permanently block others. If one of the n chosen channels goes to idle, we can give service to one of the other requests. Typically, blocking is used when channels must be held for long time periods. The traditional telephone system is one example; channels are assigned at the start of a call, and other callers are blocked if no channel is available. Assuming that values of t_a and t_d are random and are exponentially distributed, the probability that a source is active, ρ , can be obtained by

$$\rho = \frac{t_a}{t_d + t_a}. \quad (3.1)$$

Let p_j be the probability that j out of m inputs are active; for $1 \leq j \leq m$,

$$p_j = \binom{m}{j} \rho^j (1-\rho)^{m-j}. \quad (3.2)$$

Thus, the probability that j out of n channels on the transmission line are in use, P_j , can be expressed by normalizing p_j over n inputs as $P_j = p_j / \sum_{i=1}^n p_i$, or

$$\begin{aligned} P_j &= \frac{\binom{m}{j} \rho^j (1-\rho)^{m-j}}{\sum_{i=0}^n \binom{m}{i} \rho^i (1-\rho)^{m-i}} \quad \text{for } 1 \leq j \leq n \\ &= \frac{\binom{m}{j} (\frac{\rho}{1-\rho})^j}{\sum_{i=0}^n \binom{m}{i} (\frac{\rho}{1-\rho})^i} \quad \text{for } 1 \leq j \leq n. \end{aligned} \quad (3.3)$$

The reason behind the normalization is that $\sum_{i=1}^n p_i$ can never be equal to 1, since $n \leq m$; according to the rule of total probability, we can say only $\sum_{i=1}^m p_i = 1$. The blocking probability of such multiplexers, P_n , can be calculated by simply letting j be n in Equation (3.3), denoting that all n channels are occupied. If we also substitute $\rho = t_a/(t_d + t_a)$, the blocking probability can be obtained when $j = n$:

$$P_n = \frac{\binom{m}{n} (\frac{t_a}{t_d})^n}{\sum_{i=0}^n \binom{m}{i} (\frac{t_a}{t_d})^i}. \quad (3.4)$$

The preceding discussion can be concluded by using basic probability theory to find the average number of busy channels or the expected number of busy channels as

$$E[C] = \sum_{j=0}^n j P_j. \quad (3.5)$$

Example. A 12-input TDM becomes an average of 2 μs active and 1 μs inactive on each input line. Frames can contain only five channels. Find the probability that a source is active, ρ , and probability that three channels are in use.

Solution. We know that $m = 12$ and $n = 5$; since $t_a = 2 \mu s$, and $t_d = 1 \mu s$:

$$\rho = \frac{t_a}{t_d + t_a} = 0.66,$$

and the probability that all five channels are in use is

$$P_{n=5} = \frac{\binom{12}{3} (\frac{2}{1})^5}{\sum_{i=0}^5 \binom{12}{i} (\frac{2}{1})^i} = 0.44.$$

Statistical TDM

In *statistical* TDM, a frame's time slots are dynamically allocated, based on demand. This method removes all the empty slots on a frame and makes the multiplexer operate more efficiently. Meanwhile, the trade-off of such a technique is the requirement that additional overhead be attached to each outgoing channel. This additional data is needed because each channel must carry information about which line it belonged to. As a result, in the statistical multiplexer, the frame length is variable not only because of different channel sizes but also because of the possible absence of some channels.

Consider a multiplexer with n available channels. If the number of requesting input sources, m , is greater than n channels, the multiplexer typically reacts by *clipping*, whereby unassigned sources are partially transmitted, or clipped. Designing multiplexers with $m > n$ stems from the fact that all m sources may not be practically active simultaneously. In some multiplexing systems, *clipping* is exploited especially for applications in which sources do not transmit continuously, and the best usage of the multiplex output line is targeted. By assigning active sources to channels dynamically, the multiplex output can achieve more efficient channel usage. For example, some satellite or microwave systems detect information energy and assign a user to a channel only when a user is active.

Consider the same definitions as presented in the blocking case for t_a , t_d , m , n ($m > n$), and ρ . Similarly, assume that values of t_a and t_d are random and are exponentially distributed. Assume also that the outgoing transmission line has n channels available but the multiplexer has m inputs, where $m > n$. If more than n inputs are active, we can dynamically choose n out of m active sources and temporarily block other sources. With temporary blocking, the source is forced to lose, or clip, data for a short period of time, but the source may return to a scanning scenario if a channel becomes free. This method maximizes the use of the common transmission line and offers a method of using the multiplexer bandwidth in silence mode. Note that the amount of data lost from each source depends on t_a , t_d , m , and n . Similar to the blocking case, ρ can be defined by $\rho = t_a/(t_d + t_a)$, and the probability that exactly k out of m sources are active is

$$P_k = \binom{m}{k} (\rho)^k (1 - \rho)^{m-k}. \quad (3.6)$$

Therefore, P_C , the clipping probability, or the probability that an idle source finds at least n sources busy at the time it becomes active, can be obtained by considering all m sources beyond n active sources minus 1 (the examining source):

$$P_C = \sum_{i=n}^{m-1} \binom{(m-1)}{i} \rho^i (1 - \rho)^{m-1-i}. \quad (3.7)$$

Clearly, the average number of used channels is $\sum_{i=1}^n i P_i$. The average number of busy channels can then be derived by

$$E[C] = \sum_{i=1}^n i P_i + n \sum_{i=n+1}^m P_i. \quad (3.8)$$

Example. For a multiplexer with ten inputs, six channels, and $\rho = 0.67$, find the clipping probability.

Solution. Since $m = 10$, $n = 6$, $P_c \approx 0.65$.

3.2 Modems and Internet Access Devices

Users access the Internet from residential areas primarily through *modems*. A modem (*modulation/demodulation unit*) is a device that converts the digital data to a modulated form of signal that takes less bandwidth. A modem is required to create an appropriate digital signal to access the Internet. A user can access the Internet by using either the existing telephone link infrastructure or the existing cable TV infrastructure. As a result, an Internet user has several choices of modems. Two commonly used ones are the digital subscriber line (DSL) *modem* and the *cable modem* (Figure 3.5). A DSL company uses the existing twisted-pair copper lines to provide Internet access; a cable TV company uses optical fibers and coaxial cables to provide that access.

3.2.1 Line Coding Methods

Before processing a raw signal for modulation, a *line coding process* is performed on binary signals for digital transmission. With line coding, a binary information sequence is converted into a digital code. This process is required to maximize bit rate in digital transmission. Encoding process is essential to also recover the bit timing information from the digital signal so that the receiving sample clock can synchronize with the

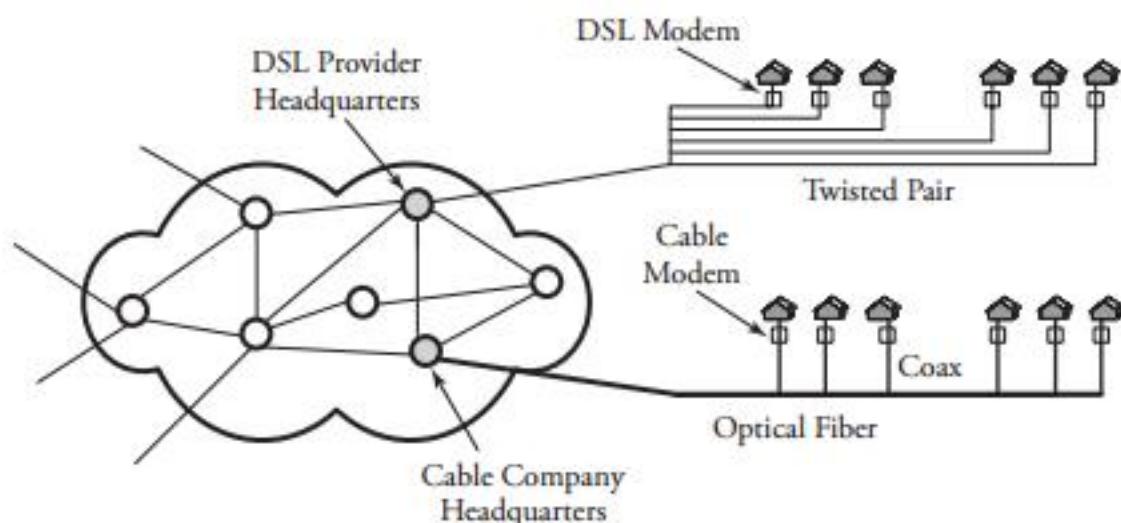


Figure 3.5 The choice of DSL modem or cable modem in residential areas for Internet access

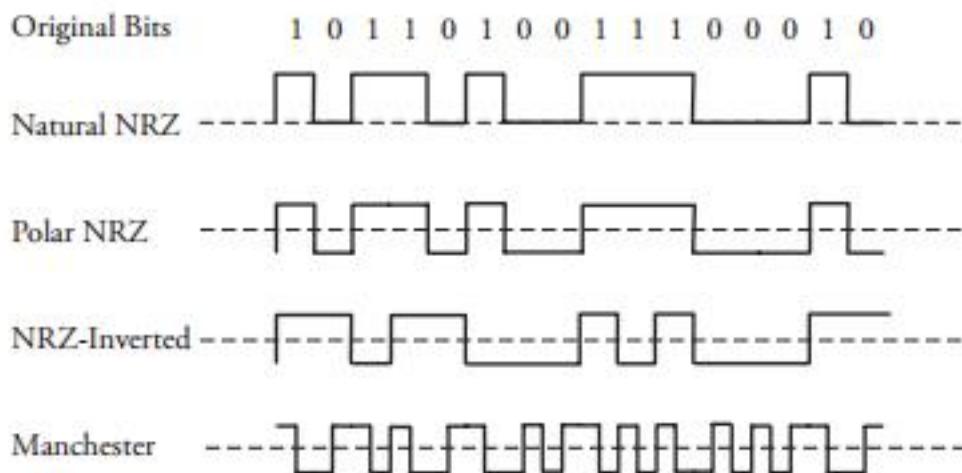


Figure 3.6 Typical line coding techniques for computer communications.

transmitting clock. The timing synchronization is especially crucial in the performance of LANs. Other reasons for line coding are reduction in transmitted power, and removal of DC voltage from transmission lines.

Typically, the cost and complexity of a line encoder is the main factor in the selection of encoder for a given application. Figure 3.6 shows several practical line coding techniques for computer communications. Encoded signals are produced by the line codes for the binary sequence 1011 0100 1110 0010. The simplest form of line coding is the *natural nonreturn-to-zero* (NRZ) where a binary 1 is represented by a $+V$ voltage level, and a 0 is represented by a 0 voltage. The average transmitted power in this method is $(1/2)V^2 + (1/2)0^2 = V^2/2$. This method creates an average of $V/2$ DC voltage on the transmission lines which is not popular for LAN systems.

A more power-efficient line coding method is known as *polar NRZ*. In this method, a binary 1 is mapped to $+V/2$ and a binary 0 is represented by $-V/2$. The average power is then given by $(1/2)(+V/2)^2 + (1/2)(-V/2)^2 = V^2/4$. On average, this method has no DC component and is suitable in most networking applications.

A problem with natural and polar NRZ coding methods is that a polarity error can cause all 1s to be delivered as 1s and all 0s as 0s. As a solution to this problem the *NRZ-inverted* method is introduced. With the NRZ-inverted coding, the binary information is mapped into transitions at the beginning of each interval so that a binary 1 is converted to a transition at the beginning of a bit time and a 0 having no transition, and the signal remains constant during the actual bit time. Notice that, errors in this method of encoding occur in pairs. This means that any error in one bit time generates a wrong basis for the next time leading to a new error in the next bit.

From the frequency stand-point, both the natural NRZ and NRZ-inverted methods produce spectrum starting from very low frequencies close to zero due to existence of either DC components or less frequent transitions of 0s to 1ns or vice versa. Although, the bipolar NRZ has a better spectrum distribution in this sense, the immunity to noise can still be an issue for all three types of NRZ coding. Low frequencies can also be a bottleneck in some communication systems, as telephone transmission systems, do not pass the frequencies below 200 Hz. To overcome this issue, the *Manchester encoding method* is introduced.

With the Manchester encoding method, a binary 1 is represented by a 1 plus a transition to 0 and then a 0; and a binary 0 is represented by a 0 plus a transition to 1 and then a 1. A great feature of the Manchester encoding is that it is self-clocking. The fact that each binary bit contains a transition at the middle of its timing makes the timing recovery very easy. From the frequency standpoint, the bit rate is doubled compared with NRZ methods that significantly enhances the shape of its spectrum where lower frequencies are shifted up. This method of line encoding is suitable for LANs especially for Gigabit Ethernets to be discussed in Chapter 5.

3.2.2 Digital Modulation Techniques

In order to reduce the bandwidth of digital signals, *digital modulation technique* is required before any transmission. In a modulation process, the amplitude, phase, or frequency of a carrier signal varies with the variations of a digital information signal. Depending on the nature and objective of a modem, the following schemes can be used:

- *Amplitude shift keying (ASK)*
- *Frequency shift keying (FSK)*
- *Phase shift keying (PSK)*
- *Quadrature amplitude modulation (QAM)*

Knowing the types of modulation techniques, we now study the two types of modems for accessing the Internet: digital subscriber line (DSL) modems and cable modems.

Figure 3.7 shows the implementation of ASK, FSK, PSK, and QAM systems. In an ASK system, incoming data, $D(t)$, containing binary 0s and 1s, is modulated over a constant-frequency and constant amplitude sinusoidal carrier signal, $\cos 2\pi f t$, where f is the frequency of the carrier. The resulting modulated signal is represented by a cosine with the same frequency f where a binary 1 is present, and no signal when a binary 0 is present. In other words, if we multiply our binary data by a constant cosine,

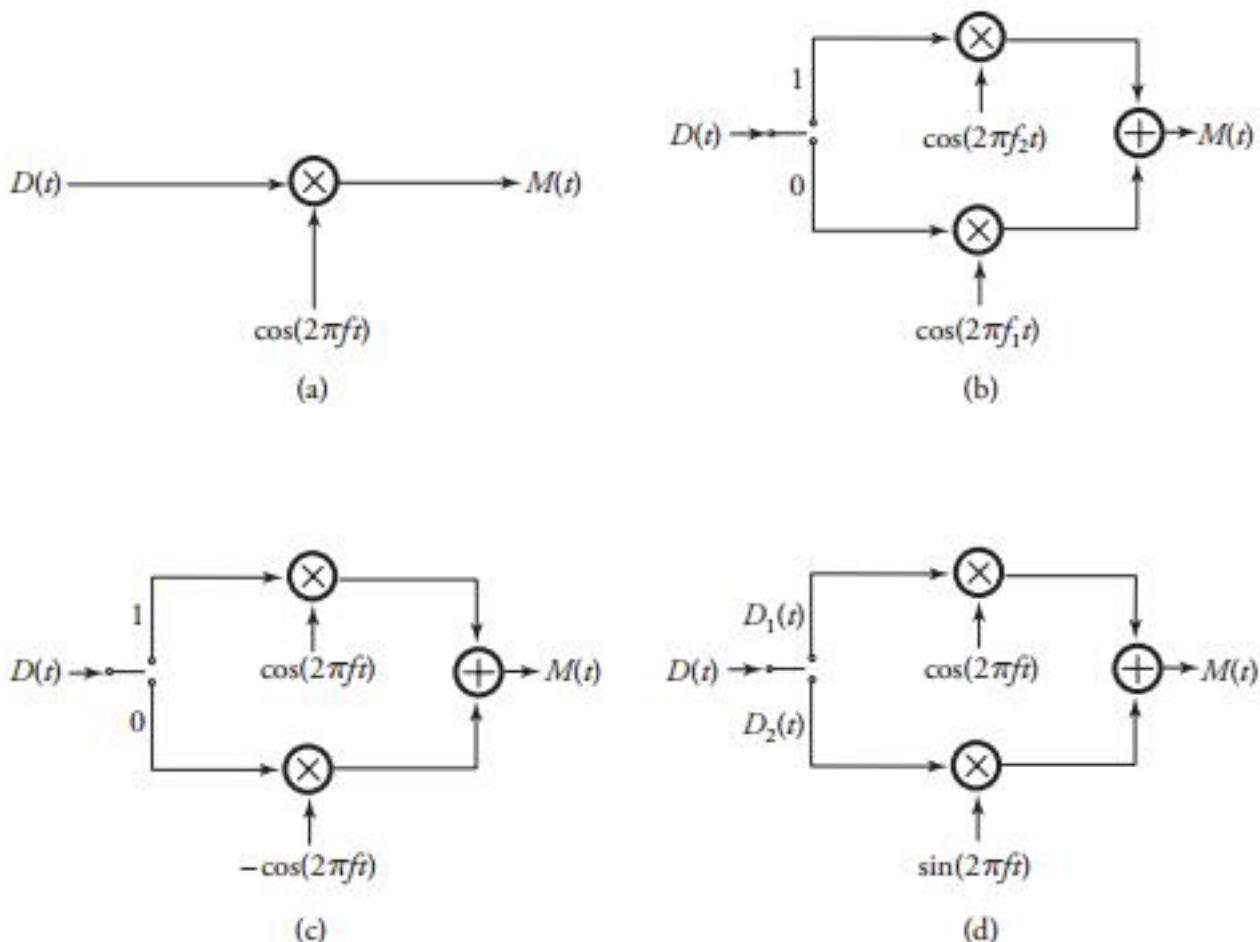


Figure 3.7 The four modulation techniques: (a) ASK, (b) FSK, (c) PSK, and (d) QAM

we obtain the ASK-modulated version of our data, $M(t) = D(t) \cos(2\pi ft)$. At the receiver, the ASK demodulator only needs to determine the presence or absence of the sinusoid in a given time interval in order to detect the original data. In practice, $D(t)$ can be extracted at the receiver if $M(t)$ is multiplied by the same carrier signal but with doubled amplitude, $2 \cos(2\pi ft)$, as follows:

$$[D(t) \cos(2\pi f t)][2 \cos(2\pi f t)] = 2D(t) \cos^2(2\pi f t) = D(t)[1 + \cos(4\pi f t)] \quad (3.9)$$

Note that $\cos(4\pi ft)$ component can be easily filtered out. The FSK scheme is a bit more complicated compared to the ASK scheme. In a FSK modulator, there are two different sinusoidal carriers: f_1 to represent a binary 1 and f_2 to represent binary 0. Therefore, FSK and ASK are similar in the sense that we multiply binary data by a constant sinusoid to obtain the modulated version of data, and they are difference is the sence that three is a sinusoid when 0 appears. Clearly, in the FSK system, the frequency

of the carrier varies according to the information such that we have $\cos(2\pi f_1 t)$ instead of a binary 1 and $\cos(2\pi f_2 t + \pi)$ instead of a binary 0.

In a PSK system, the phase of the sinusoidal carrier signal changes according to the information sequence as shown in Figure 3.7 (c). A binary 1 is represented by $\cos(2\pi f t)$ and a binary 0 by $\cos(2\pi f t + \pi)$. Similarly, we can rephrase the definition of a PSK system: in a PSK modulator we multiply the sinusoidal signal by +1 when the information is a 1, and by -1 when a 0 is present.

QAM is another modulator in which we split the original information stream into two equal sequences, $D_1(t)$ and $D_2(t)$, consisting of the odd and even symbols, respectively. Each sequence has a rate of s symbols/second and the number of bits per symbol is typically constant. As shown in Figure 3.7 (d), we take $D_1(t)$ and produce a modulated signal by multiplying it by $\cos(2\pi f t)$ for a T -second interval. Similarly, we take $D_2(t)$ and produce a modulated signal by multiplying it by $\sin(2\pi f t)$ for a T -second interval. The first component $D_1(t)$ is known as *in-phase component*, and the second component $D_2(t)$ is known as *quadrature-phase component*. Therefore, at the output of the modulator we have:

$$M(t) = D_1(t) \cos(2\pi f t) + D_2(t) \sin(2\pi f t) \quad (3.10)$$

The QAM scheme can be realized as the simultaneous modulation of the amplitude and the phase of a carrier signal as Equation (3.10) can be rearranged as:

$$M(t) = \sqrt{D_1^2(t) + D_2^2(t)} \cos \left(2\pi f t + \tan^{-1} \frac{D_2(t)}{D_1(t)} \right) \quad (3.11)$$

Similar to what was explained for the ASK system, the original data, $D(t)$, can be extracted at the receiver if $M(t)$ is multiplied by the same carrier signal but with doubled amplitude. However, since $M(t)$ has two terms in this case, $D_1(t) \cos(2\pi f t)$ must be multiplied by $2 \cos(2\pi f t)$, and $D_2(t) \sin(2\pi f t)$ must be multiplied by $2 \sin(2\pi f t)$.

3.2.3 Digital Subscriber Line (DSL) Modems

Digital subscriber line (DSL) technology is a convenient option for home users to access the Internet. This technology offers various versions of DSL technology: ADSL, VDSL, HDSL, and SDSL, or, in general, xDSL.

Among the xDSL types, *asymmetric* DSL (ADSL) is popular and is designed for residential users. A modem is designed to be connected to telephone links. These links are capable of handling bandwidths up to 1.1 MHz. Out of this bandwidth, only 4 KHz are used for a phone conversation. Consequently, the remaining bandwidth can become available to be allocated to data communications. However, other factors, such

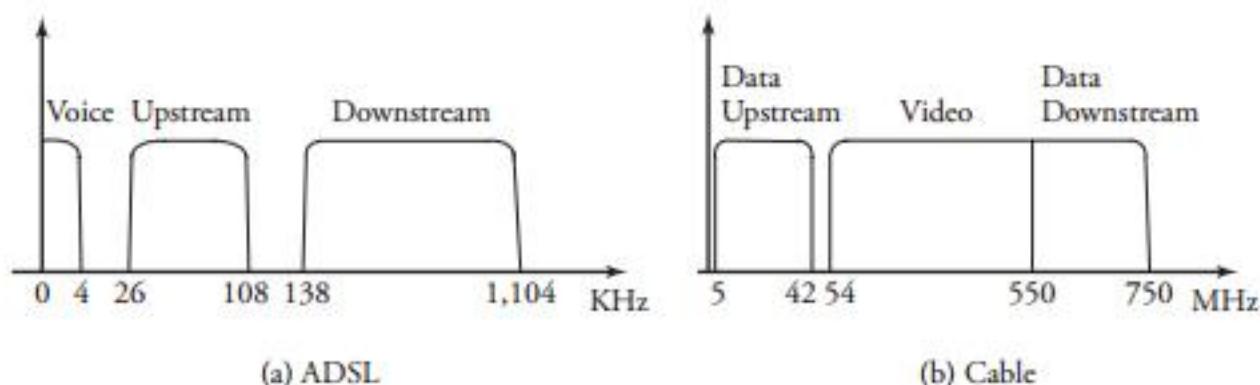


Figure 3.8 The frequency bands of (a) ADSL modems and (b) cable modems

as the distance between a user and a switching office, and the condition and size of the link might restrict this remaining bandwidth from being completely available.

The details of spectrum division for an ADSL modem are shown in Figure 3.8 (a). The standard modulation technique for ADSL is QAM. The available bandwidth of 1.1 MHz is divided into 256 channels, each using a bandwidth of approximately 4.312 KHz. Voice communication uses channel 0. Channels 1–5 remain idle and together act as a guard band between voice and data communication. Because data communication bandwidth is split into two bandwidths—*upstream* for communications from the user to the Internet and *downstream* for communications from the Internet to the user—the technique is said to be asymmetric. Channels 6–30 are allocated to the upstream bandwidth, with 1 channel dedicated to control and 24 channels assigned to data transfer. Thus, 24 channels with QAM offer $24 \times 4 \text{ KHz} \times 15 = 1.44 \text{ Mb/s}$ bandwidth in the upstream direction, as QAM requires 15-bit encoding. Channels 31–255 are assigned to the downstream bandwidth, with 1 channel for control and the remaining 224 channels for data. With QAM, up to $224 \times 4 \text{ KHz} \times 15 = 13.4 \text{ Mb/s}$ is achieved for the downstream bandwidth.

Another type of DSL technique is *symmetric digital subscriber line* (SDSL). This technique divides the available bandwidth equally between downstream and upstream data transfer. *High-bit-rate digital subscriber line* (HDSL), as another option, was designed to compete with T-1 lines (1.544 Mb/s). A T-1 line is susceptible to attenuation at high frequencies and thus limits the length of line to 1 km. HDSL uses two twisted-pair wires and 2B1Q, an encoding technique less susceptible to attenuation. As a result, HDSL can achieve a data rate of 2 Mb/s without needing repeaters for up to 3.6 km. *Very high bit-rate digital subscriber line* (VDSL) is similar to ADSL but uses coaxial or fiber-optic cable for a bit rate of 50 Mb/s to 55 Mb/s downstream and 1.5 Mb/s to 2.5 Mb/s upstream data transfer.

3.2.4 Cable Modems

As mentioned in the previous section, the DSL modem uses the existing twisted-pair telephone cables for providing residential users with access to the Internet, as shown in Figure 3.5. This type of cable clearly has limited bandwidth and is susceptible to errors. An alternative is to use the cable TV network for Internet access. A cable company lays out very high-speed backbone optical fiber cables all the way to the residential buildings, each of which can then be connected to the optical infrastructure for TV, radio, and the Internet through either a coaxial (coax) cable or optical fiber, depending on its demand and budget. This network is called *hybrid fiber-coaxial* (HFC). Video signals are transmitted downstream from headquarters to users. Communication in an HFC cable TV network is bidirectional. The cable company divides the bandwidth into video/radio, downstream data, and upstream data. Coaxial cables can carry signals up to 750 MHz.

Details of the spectrum division for a cable modem are shown in Figure 3.8 (b). About 500 MHz of this bandwidth is assigned to TV channels. As the bandwidth of each TV channel is 6 MHz, the assigned bandwidth can accommodate more than 80 channels. Some technical methods allow this number to increase to 180 TV channels. About 200 MHz of the coax bandwidth, from 550 MHz to 750 MHz, is allocated to the downstream data transfer: from the Internet side to a user. This bandwidth is also divided to about 33 channels, each with 6 MHz bandwidth. The cable modem uses the 64-QAM or 256-QAM modulation technique for the downstream data transfer. These modulation techniques use 5-bit encoding, so the bandwidth of the downstream data channel can be $5 \text{ b/Hz} \times 6 = 30 \text{ Mb/s}$.

The upstream data premises communicate to the Internet and occupy 37 MHz, from 5 MHz to 42 MHz, including 6 MHz-wide channels. The upstream data is modulated using the QPSK (Quadrature PSK) technique, which is less susceptible to noise in the lower frequency range. Using 2 bits per Hz offers the downstream data rate at $2 \text{ b/Hz} \times 6 \text{ MHz} = 12 \text{ Mb/s}$. The protocol for upstream communication is summarized as follows.

Begin Upstream Communication Protocol

1. The cable modem checks the downstream channels to determine whether any packet periodically sent by the cable company seeks any new modems attached to the cable.
2. The cable company sends a packet to the cable modem, specifying the modem allocated downstream.

3. The cable modem sends a packet to the cable company, requesting the Internet address.
4. The modem and the cable company go through a handshaking process and exchange some packets.
5. The cable company delivers an identifier to the modem.
6. Internet access in the allocated upstream channel bandwidth can then be granted to the modem in the allocated upstream channel. ■

Note that a user needs to use one 6-MHz-wide channel assigned by the cable company. To better use the optical fiber bandwidth, all users of the same neighborhood need to timeshare an FDM channel. Therefore, a user must contend for a channel with others and must wait if the channel is unavailable.

3.3 Switching and Routing Devices

Switching devices are categorized by their complexity, as follows:

- *Layers 1 and 2 switches* are typically simple. For example, *repeaters* and *hubs* are known as layer 1 switches; *bridges*, as layer 2 switches.
- *Layer 3 or higher switches* are complex. Routers, for example, are layer 3 switches.

Figure 3.9 depicts interconnections and switching functions at layer 1, layer 2, layer 3, and upper layers of the five layer protocol stack.

3.3.1 Repeaters, Hubs, and Bridges

Repeaters and *hubs*, the simplest switching devices, are designed primarily to interconnect very small LAN units without any involvement in the complex routing processes. Chapter 5 provides several examples of repeaters and hubs.

Repeaters are used to connect two segments of a LAN. A repeater's essential function, *signal regeneration*, differentiates it from a piece of cable. Signal regeneration is needed when the LAN length is extended. When a LAN is extended, bits can be corrupted and decayed. As shown in Figure 3.9, two LANs are interconnected using a repeater at the physical layer (layer 1). The repeater assumes that the connecting LANs have the same protocol and simply accepts bits from one LAN and transmits them on to the other LANs.

A *hub* is another simple device and is used to provide connections among multiple users in layer 1 of a protocol stack. A hub is similar to the repeater but connects several pieces of a LAN. In fact, a hub is a multipoint repeater. To perform this type of

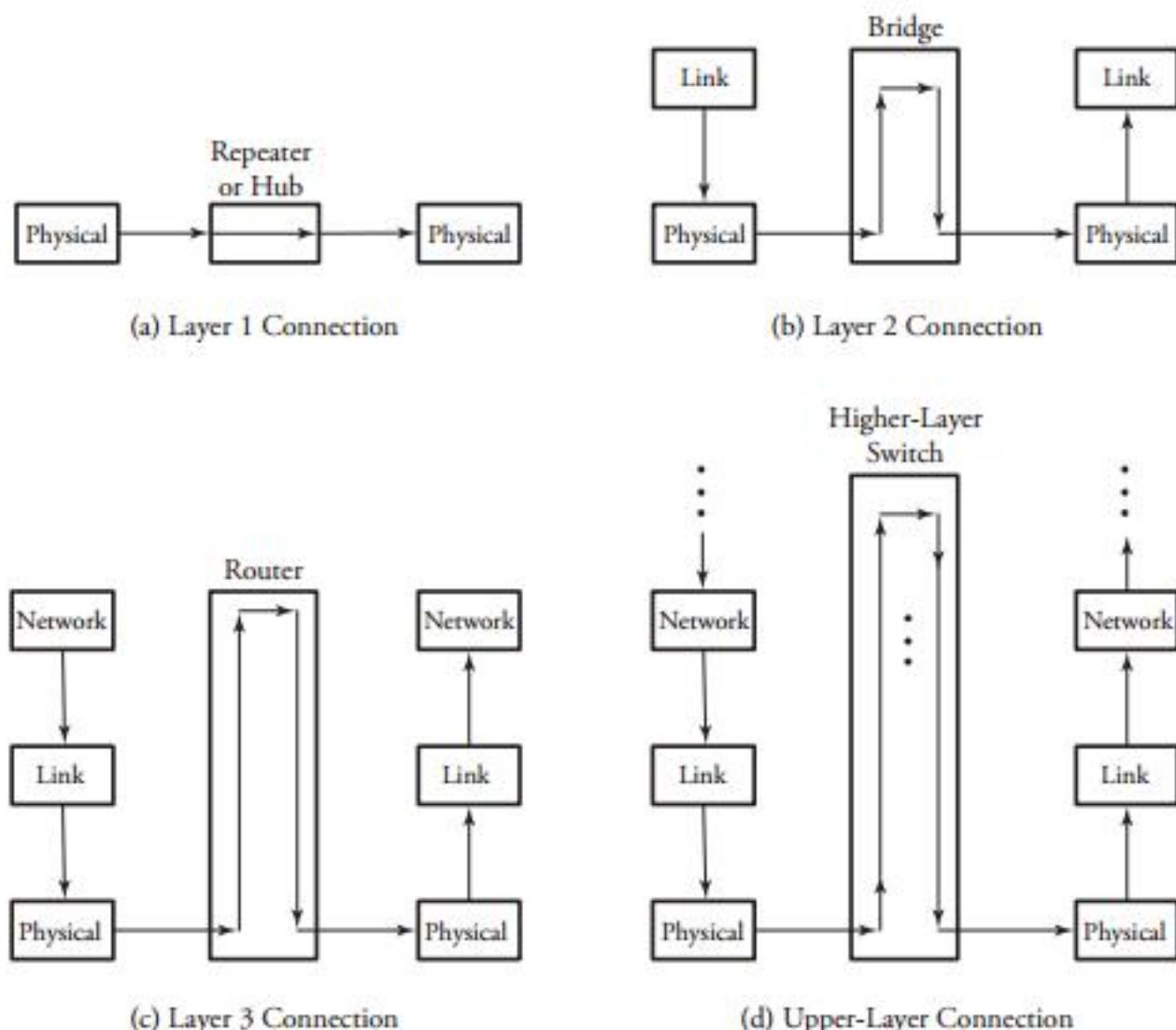


Figure 3.9 Connections in different layers of protocols

connection, a copying element is installed to copy and forward a packet or a frame to all connected users. As a typical repeater, a hub has regeneration capability to strengthen incoming data to prevent any decay.

A *bridge* is a switch that connects two pieces of a LAN or two LANs, especially those operating in layer 2 of the protocol stack. However, a bridge can also be used in layer 1 for signal regeneration. As it operates at layer 2, a bridge does more than simply extend the range of the network. A bridge checks the physical address of any destination user and enhances the efficiency of networks by facilitating simultaneous transmissions within multiple LANs. Unlike repeaters, which transmit a frame to all associated users within a LAN, a bridge does not forward a frame to all LAN users and

thus can isolate traffic between two LANs. Bridges can make decisions about where to forward frames. Bridges perform data link functions, such as forwarding, formatting, and error detection.

Bridges can forward only one frame at a time in a store-and-forward fashion. However, sophisticated layer 2 switches can forward multiple frames simultaneously through multiple parallel data paths. Layer 2 switches can also operate as cut-through devices, significantly enhancing communication performance. These advantages have made switches more popular than typical bridges. LAN bridges or other intelligent layer 2 switches are devices that use layer 2 data to forward or filter traffic.

3.3.2 Routers and Higher-Layer Switches

A *router* is a layer 3 switch that connects other routing nodes, providing a virtual or nonvirtual circuit. A router is dependent on protocols and establishes physical circuits for individual node-pair connection. In packet-switched networks, several pairs of communicating end points can share a circuit virtually, through their own dedicated channels, instead of occupying the entire physical circuit. Bandwidth in a packet-switched network is then dynamically released and shared as needed. A router has a routing look-up table for routing packets. If a communication is connectionless, packets of a message are sent individually but not in order. Owing to the presentation of more sophisticated input and output processors in routers, a router is not concerned about the order of the packets. Layer 3 switches are of two types:

- *Packet-by-packet switches*, by which packet forwarding is handled based on each individual packet.
- *Flow-based switches*, by which a number of packets having the same source and destination are identified and forwarded together. This scheme speeds up the forwarding process.

Layer 2 switches are developed to replace routers at a local area network. Typically, the general strategy of the network management is to use the technology of layer 2 switching at layer 3 routing to improve the quality of forwarding packets of routers and to make routers provide sufficient network guarantee. A layer 4 switch uses the information from higher levels for routing decisions. Basically, layer 4 switches are the application switches for both layer 2 and layer 3. In layer 4, packets are normally forwarded on a connectionless system.

3.4 Router Structure

Routers are the building blocks of wide area networks. Figure 3.10 shows an abstract model of a router as a layer 3 switch. Packets arrive at n input ports and are routed out from n output ports. The system consists of four main parts: *input port processors*, *output port processors*, *switch fabric* (switching network), and *switch controller*.

3.4.1 Input Port Processor (IPP)

Input and *output port processors*, as interfaces to switch fabric, are commercially implemented together in *router line cards*, which contain some of the task of the physical and data link layers. The functionality of the data link layer is implemented as a separate chip in IPP, which also provides a buffer to match the speed between the input and the switch fabric. Switch performance is limited by processing capability, storage elements, and bus bandwidth. The processing capability dictates the maximum rate of the switch. Owing to the speed mismatch between the rate at which a packet arrives on the switch and the processing speed of the switch fabric, input packet rate dictates the amount of required buffering storage. The bus bandwidth determines the time taken for a packet to be transferred between the input and output ports.

An *input port processor* (IPP) typically consists of several main modules, as shown in Figure 3.11. These modules are *packet fragmentation*, *main buffer*, *multicast process*, *routing table*, *packet encapsulator*, and a comprehensive *QoS*.

Packet Fragmentation

The *packet fragmentation unit*, converts packets to smaller sizes. Large packets cause different issues at the network and link layers. One obvious application of packet

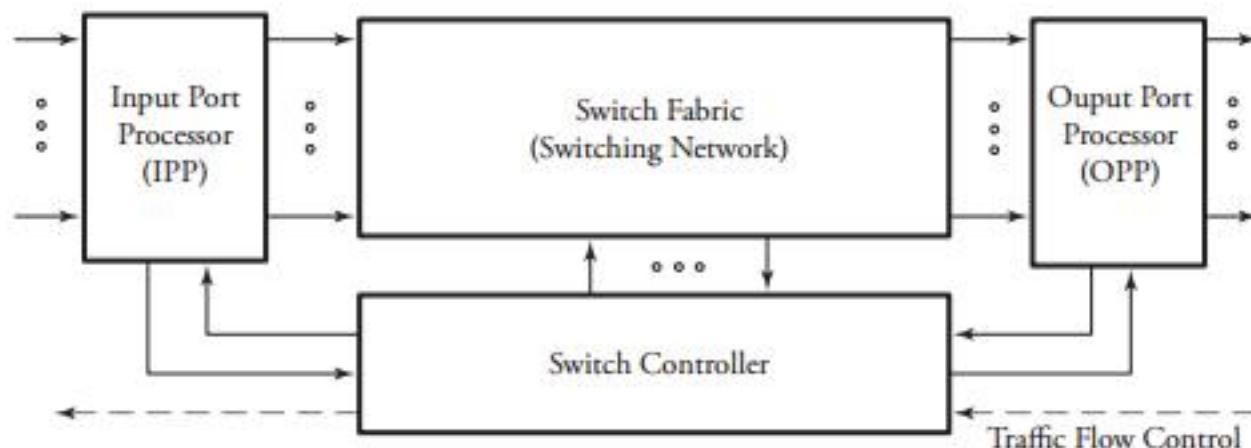


Figure 3.10 Overview of a typical router

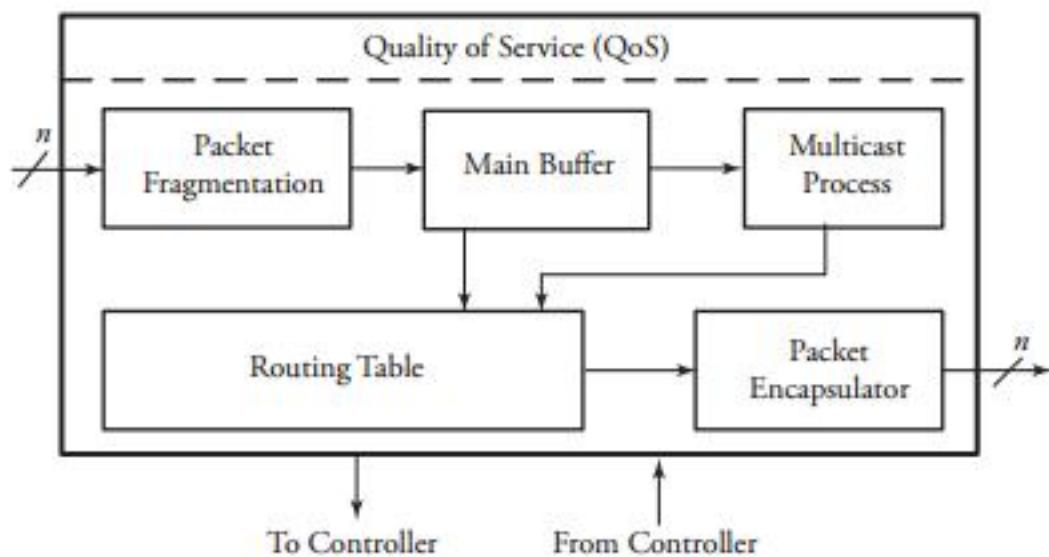


Figure 3.11 Overview of a typical IPP in routers

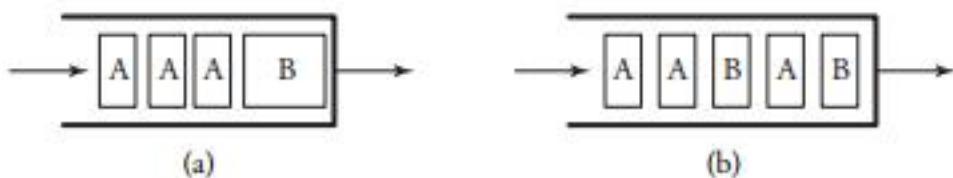


Figure 3.12 Packet fragmentation: (a) without fragmentation; (b) with fragmentation

fragmentation occurs in typical LANs, in which large packets must be fragmented into smaller frames. Another example occurs when large packets must be buffered at the input port interface of a router, as buffer slots are usually only 512 bytes long. One solution to this problem is to partition packets into smaller fragments and then reassemble them at the output port processor (OPP) after processing them in the switching system. Figure 3.12 shows simple packet fragmentation at the input buffer side of a switch. It is always desirable to find the optimum packet size that minimizes the delay.

Routing Table

The *routing table* is a look-up table containing all available destination addresses and the corresponding switch output port. An external algorithm fills this routing look-up table. Thus, the purpose of the routing table is to look up an entry corresponding to the destination address of the incoming packet and to provide the output network port. As soon as a routing decision is made, all the information should be saved on the routing table. When a packet enters an IPP, the destination port of the switch should

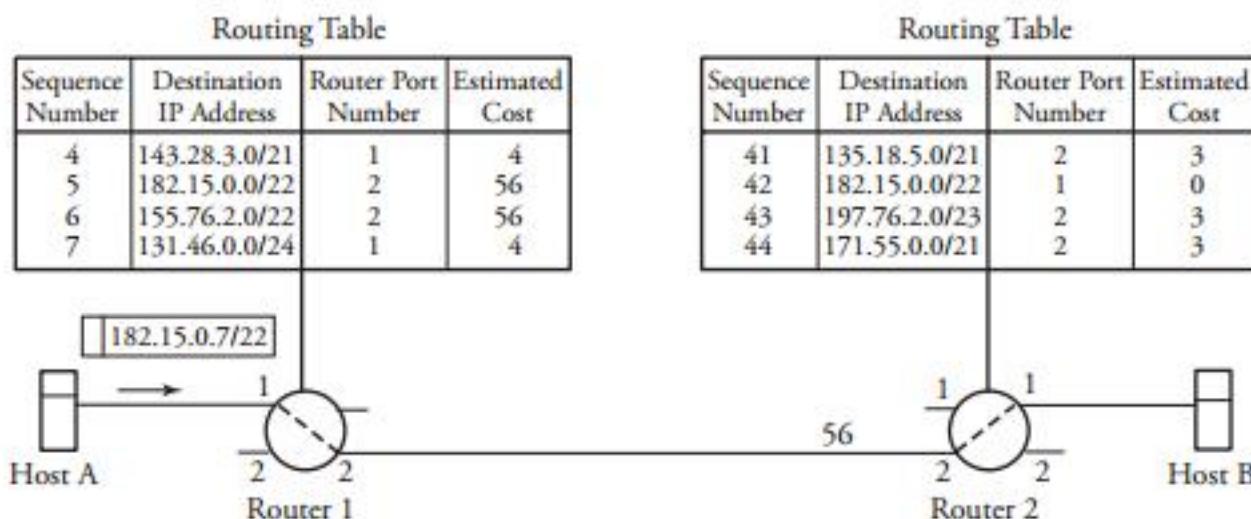


Figure 3.13 Routing tables at routers

be chosen, based on the destination address of the incoming packet. This destination port needs to be appended to the incoming packet as part of the switch header.

The look-up table management strategy takes advantage of first in, first out (FIFO) queues' speed and memory robustness. To increase memory performance, queue sizes are fixed to reduce control logic. Since network packets can be of various lengths, a memory device is needed to store packet payloads while a fixed-length header travels through the system. Since packets can arrive and leave the network in different order, a memory monitor is necessary to keep track of which locations in memory are free for use. Borrowing a concept from operating systems principles, a free-memory list serves as a memory manager implemented by a stack of pointers. When a packet carrying a destination address arrives from a given link i , its destination address is used to identify the corresponding output port j .

Figure 3.13 shows an example of routing tables at routers between hosts A and B. Assume that host B's address is requested by a packet with destination address 182.15.0.0/22 arriving at router 1. The routing table of this router stores the best-possible path for each destination. Assume that for a given time, this destination is found in entry row 5. The routing table then indicates that port 2 of the router is the right output to go. The table makes the routing decision, based on the estimated cost of the link, which is also stated in the corresponding entry. The cost of each link, as described in Chapter 7, is a measure of the load on each link. When the packet arrives at router 2, this switch performs the same procedure.

Multicast Process

A *multicast process* is necessary for copying packets when multiple copies of a packet are expected to be made on a switching node. Using a memory module for storage, copying is done efficiently. The copying function can easily be achieved by appending a counter field to memory locations to signify the needed number of copies of that location. The memory module is used to store packets and then duplicate multicast packets by holding memory until all instances of the multicast packet have exited IPP. Writing to memory takes two passes for a multicast packet and only one pass for a unicast packet. In order to keep track of how many copies a multicast packet needs, the packet counter in the memory module must be augmented after the multicast packet has been written to memory. Each entry in the memory module consists of a valid bit, a counter value, and memory data. The multicast techniques and protocols are described in a greater detail in Chapter 15.

Packet Encapsulation

Packet encapsulation instantiates the routing table module, performs the routing table lookups, and inserts the switch output port number into the network header. The *serial-to-parallel multiplexing* unit converts an incoming serial byte stream into a fully parallel data stream. This unit also processes the incoming IP header to determine whether the packet is unicast or multicast and extracts the type-of-service field. Once the full packet is received, it is stored into memory. The packet encapsulation unit formats the incoming packet with a header before forwarding the packet to the crossbar.

Congestion Controller

The *congestion controller* module shields the switching node from any disorders in the traffic flow. Congestion can be controlled in several ways. Sending a reverse-warning packet to the upstream node to avoid exceeding traffic is one common technology installed in the structure of advanced switching systems. Realistically, spacing between incoming packets is irregular. This irregularity may cause congestion in many cases. Congestion control is explained in Chapters 7, 8, and 12.

3.4.2 Switch Fabric

In the switch fabric of a router, packets are routed from input ports to the desired output ports. A packet can also be multicast to more than one output. Finally, in the output port processors, packets are buffered and resequenced in order to avoid packet

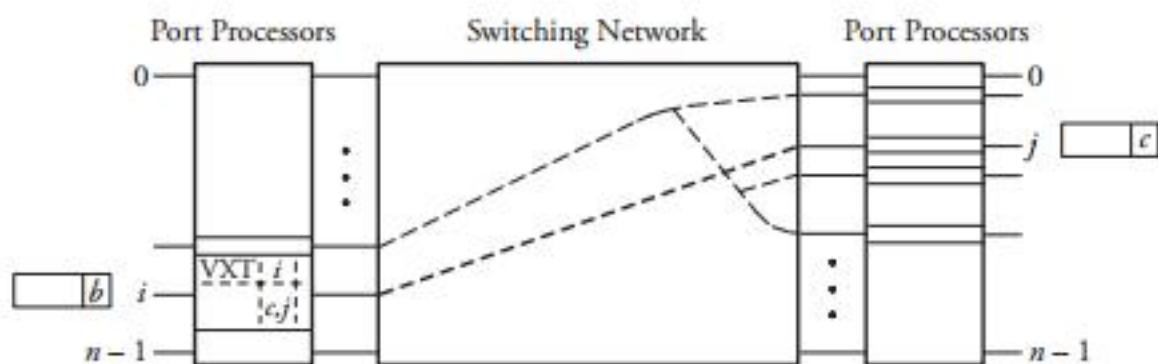


Figure 3.14 Interaction between an IPP and its switch fabric in a virtual-circuit switching router

misordering. In addition, a number of other important processes and functions take place in each of the mentioned blocks.

Figure 3.14 shows an abstract model of a virtual-circuit switching router, another example of switching systems. This model can work for ATM technology: Cells (packets) arrive at n input ports and are routed out from n output ports. When a cell carrying VCI b arrives from a given link i , the cell's VCI is used to index a *virtual-circuit translation table* (VXT) in the corresponding input port processor to identify the output link address j and a new VCI c . In the switching network, cells are routed to the desired outputs. As shown in Figure 3.14, a cell can also be multicast to more than one output. Finally, in output port processors, cells are buffered; in some switch architectures, cells are resequenced in order to avoid misordering.

3.4.3 Switch Controller

The *controller* part of a switching system makes decisions leading to the transmission of packets to the requested output(s). The details of the controller are illustrated in Figure 3.15. The controller receives packets from an IPP, but only the headers of packets are processed in the controller. In the controller, the *header decoder* first converts the control information of an arriving packet into an initial requested output vector. This bit vector carries the information pertaining to the replication of a packet so that any bit of 1 represents a request for one of the corresponding switch outputs.

The initial request vector ultimately gets routed to the *buffer control* unit, which generates a priority value for each packet to enable it for arbitration. This information, along with the request vector, enters an array of *arbitration elements* in the *contention resolution unit*. Each packet in one column of an arbitration array contends with other packets on a shared bus to access the switch output associated with that column. After a packet wins the contention, its identity (buffer index number) is transmitted out to

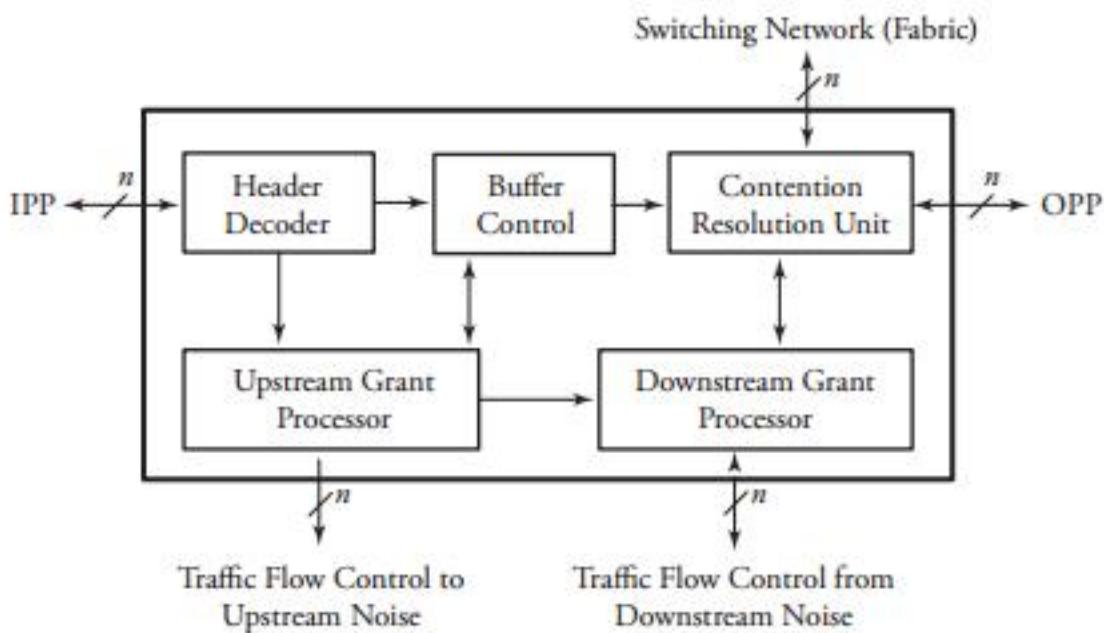


Figure 3.15 Overview of a switching system controller

an OPP. This identity and the buffer-control bit explained earlier are also transferred to the switching fabric (network), signaling them to release the packet. This mechanism ensures that a losing packet in the competition remains in the buffer. The buffer-control unit then raises the priority of the losing packet by 1 so that it can contribute in the next round of contention with a higher chance of winning. This process is repeated until eventually, the packet wins.

The identities of winning packets are transmitted to the switch fabric if traffic flow control signals from downstream neighboring nodes are active. The *upstream grant processor* in turn generates a corresponding set of traffic flow control signals, which are sent to the upstream neighboring nodes. This signal is an indication that the switch is prepared to receive a packet on the upstream node. This way, network congestion comes under control.

3.4.4 Output Port Processors (OPP)

Implementing *output port processors* in switches includes parallel-to-serial multiplexing, main buffer, local packet resequencer, global packet resequencer, error checker, and packet reassembler, as shown in Figure 3.16. Similar to IPP, OPP also contributes to congestion control. *Parallel-to-serial multiplexing* converts the parallel-packet format into serial packet format.

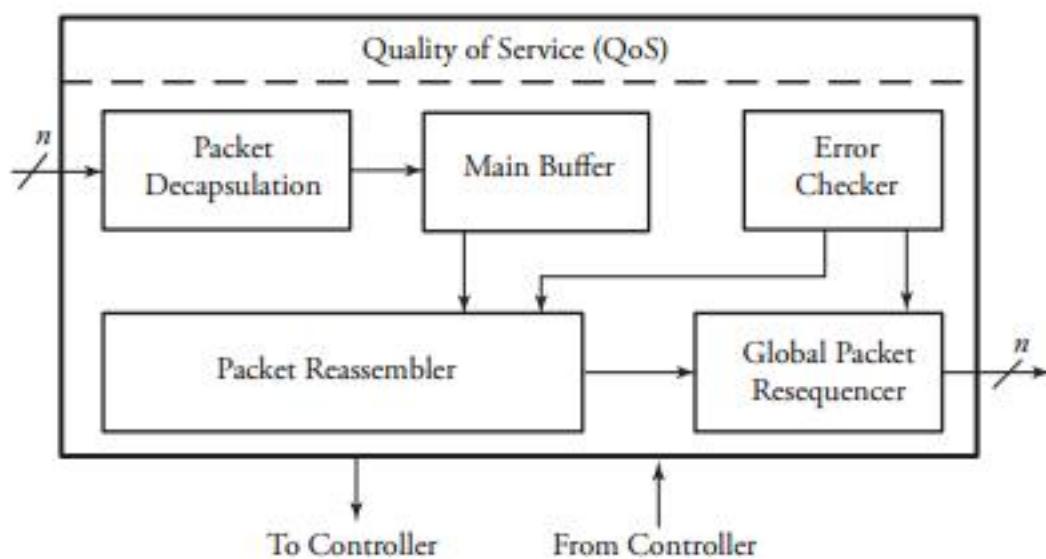


Figure 3.16 Overview of a typical OPP in routers

Main Buffer

The *buffer* unit serves as the OPP central shift register. The purpose of this buffer is to control the rate of the outgoing packets, which impacts the quality of service. After collecting signals serially from the switch fabric, the buffer forwards packets to resequencers. The queue runs on a clock driven by the link interface between the switch and an external link. This buffer must have features that support real-time and non-real-time data.

Reassembler and Resequencer

The output port processor receives a stream of packet fragments and has to identify and sort out all the related ones. The OPP reassembles them into a single packet, based on the information obtained from the fragment field of headers. For this process, the OPP must be able to handle the arrival of individual fragments at any time and in any order. Fragments may arrive out of order for many reasons. Misordered packets can occur because individual fragments, composed of a fairly large number of interconnections with different delay times, are independently routed through the switch fabric.

A *packet reassembler* buffer is used to combine fragments of IP packets. This unit resequences receiving packet fragments before transmitting them to external circuits, updates the total-length field of the IP header, and decapsulates all the local headers. The resequencer's internal buffer stores misordered fragments until a complete sequence is obtained. The in-sequence fragments are reassembled and transmitted to the external circuit. A *global packet resequencer* uses this same procedure to enforce another reordering, this time on sequences, not fragments, of packets that belong to a single user.

Error Checker and CRC

When a user sends a packet or a frame, a *cyclic redundancy check* (CRC) field is appended to the packet. The CRC is generated from an algorithm and is based on the data being carried in the packet. The CRC algorithms divide the message by another fixed-binary number in a polynomial form, producing a *checksum* as the remainder. The message receiver can perform the same division and compare the remainder with the received checksum. The *error checker* applies a series of error-checking processes on packets to ensure that no errors are on the packets and creates a stream of bits of a given length, called frames. A frame produces a *checksum bit*, called frame check sequence, which is attached to the data when transmitted.

3.5 Summary

This chapter introduced the hardware building blocks of computer networks. These building blocks, or nodes, are directly connected by physical links and allow message exchange in communication networks.

A *multiplexer* is a device that provides cost-effective connectivity by collecting data from multiple links and carrying that data on one link. Multiplexers are of various types and are subject to some useful analytical methods. Networking *modems* are used to access the Internet from remote and residential areas. A *DSL modem* uses twisted-pair telephone lines to access the Internet, whereas a *cable modem* uses optical fiber cables to provide Internet access.

Switching devices are organized according to the layer at which they are connected in networks. *Repeaters* are layer 1 switching devices that primarily extend the geographic distance spanned by a LAN protocol. A *hub* or a *bridge* supports more ports than a repeater does. The most important component of the Internet, a *router*, is treated as a layer 3 switching device. A router is made up of *input port processors*, *output port processors*, *switch fabric*, and a *switch controller*.

In the next chapter, we look at issues related to data links. Data links can be evaluated both at the physical layer and the data link layer.

3.6 Exercises

1. Human voice frequency ranges from 16 Hz to about 20 KHz. Telephone companies use the most-significant 4,000 KHz portion of this spectrum to deliver voice conversation between two users. This downgrade in the quality of conversation allows the transmission links to save bandwidth remarkably. Using a three-level hierarchy of multiplexing (12:1, 5:1, and 10:1):

- (a) How many voice conversations can this system carry?
(b) What would be the final transmission capacity of this system?
2. Consider the integration of three analog sources and four identical digital sources through a time-division multiplexer that uses its entire 170 Kb/s maximum capacity. The analog lines—with bandwidths of 5 KHz, 2 KHz, and 1 KHz, respectively—are sampled, multiplexed, quantized, and 5-bit encoded. The digital lines are multiplexed, and each carries 8 Kb/s.
 - (a) For the analog sources, find the total bit rate.
 - (b) For the digital sources, find the pulse-stuffing rate.
 - (c) Find the frame rate if a frame carries eight sampled analog channels, four digital data channels, a control channel, and a guard bit.
3. Assume that two 600 b/s terminals, five 300 b/s terminals, and a number of 150 b/s terminals are to be time-multiplexed in a character-interleaved format over a 4,800-b/s digital line. The terminals send 10 bits/character, and one synchronization character is inserted for every 99 data characters. All the terminals are asynchronous, and 3 percent of the line capacity is allocated for pulse stuffing to accommodate variations in the terminal clock rate.
 - (a) Determine the number of 150 b/s terminals that can be accommodated.
 - (b) Sketch a possible framing pattern for the multiplexer, assuming three characters per 150 b/s terminal.
4. Consider a time-division multiplexer having a frame time of $26 \mu s$. Each user channel has 6 bits, and each frame has 10 bits of overhead information. Assume that the transmission line carries 2 Mb/s of information.
 - (a) How many user channels can be accommodated on the line?
 - (b) Consider ten sources in this system, and assume a probability of 0.9 that a source is busy. What is the clipping probability?
5. Consider a synchronous TDM with eight inputs, each becoming an average of $2 \mu s$ active and $6 \mu s$ inactive. Frames can receive four channels only.
 - (a) Find the probability that a source is active.
 - (b) Find the probability that three channels of the frame are in use.
 - (c) Find the blocking probability for this multiplexer.
 - (d) Find the average number of used channels in the frame.
6. For a four-input statistical TDM, consider two different frame sizes of 2 and 3. Sketch one set of three plots, each showing the clipping probability versus $\rho = 0.2, 0.4, 0.6, 0.8$.

7. Consider a statistical TDM in which 11 sources and 10 channels are present. Find the clipping probability, assuming a probability of 0.9 that a given source is busy.
8. Find the *average clipping time* for each burst of information in statistical TDMs.
9. A string of 110011101 arrives at the line coder of a modem. Give the output form if the line coder is designed by:
 - (a) Natural NRZ
 - (b) Polar NRZ
 - (c) Manchester NRZ
10. A string of 100101011 arrives at the modulation unit of a modem. Give the output signal if the modulator is designed by:
 - (a) ASK
 - (b) FSK
 - (c) PSK
11. We want to design the input port processor of a high-speed router and analyze the delay. Incoming packets to IPP are fragmented into smaller segments, with each segment consisting of d bits data plus 50 bits of header. The switch fabric requires segments to be transmitted at r b/s. To achieve the highest possible speed:
 - (a) Is there any way to optimize the transmission delay D of each segment in terms of d and r ? How?
 - (b) Is propagation delay in the switch fabric significant compared to D ? Why?
12. *Computer simulation project.* To simulate the routing table of routers, write a computer program to construct a look-up routing table with ten entries. Each entry must contain a sequence number, time, destination address, cost of a destination node (given a certain network), and router port number. Your program should demonstrate the updating mechanism on a frequent basis.

This page intentionally left blank

CHAPTER 4

Data Links and Transmission

So far, we have discussed basic networking protocols and devices. This chapter focuses on data *links*, especially on methods of data transmission, both wired and wireless. After introducing general properties of transmission media, we investigate issues in the link layer of the overall protocol stack. The highlights of this chapter are

- *Data links*
- *Wired links and transmission*
- *Wireless links and transmission*
- *Methods of channel access on links*
- *Error detection and correction*
- *Flow control at the link layer*

We begin by discussing wired and wireless transmission media. We examine most guided and unguided link alternatives, briefly discuss their applications, and summarize key transmission characteristics. Our next major topic is methods of focusing on channel access, the physical and link layers. We also discuss *error detection* and *error correction* on data links. Finally, we explain the *stop-and-wait* and *sliding-window* protocols, which guarantee the control of link flows.

4.1 Data Links

A *data link* is the physical path between a data transmitter and data receiver. Figure 4.1 shows the range of electromagnetic-spectrum frequencies for various applications in data communications.

- The *low-frequency* subspectrum covers all the frequencies in the range 0 to approximately 15,000 Hz, which is the range of human-voice frequencies generally used in *telephone systems*.
- The *radio* frequency (RF) subspectrum covers frequencies from several KHz to several GHz. RF applications in telecommunications includes *radio systems*, *television systems*, *Bluetooth communications*, and *cell phones*.
- The *microwave* frequency subspectrum ranges from several GHz up to more than 10^{11} Hz and is used for such applications as *microwave systems*, *radar*, and *satellite communications*.
- The *infrared* frequency subspectrum ranges from more than 10^{11} Hz to less than 10^{14} Hz. The infrared signal can be used for *remote controls*, *lasers*, and *guided missiles*.
- The *light* frequency subspectrum covers all the visible-light components and is used mainly for *fiber-optic communications*.

Data transmission may use either *wired links* or *wireless links*, depending on the application and the available bandwidth of links. Transmission links can also be classified as *guided*, or directional, and *unguided*. A wired link is normally considered a guided medium for the propagation of data. A wireless medium can be designed to propagate signals in more than one direction, causing the medium to become unguided. Signals travel on the link in the form of electromagnetic waves. Let c be the speed of electro-

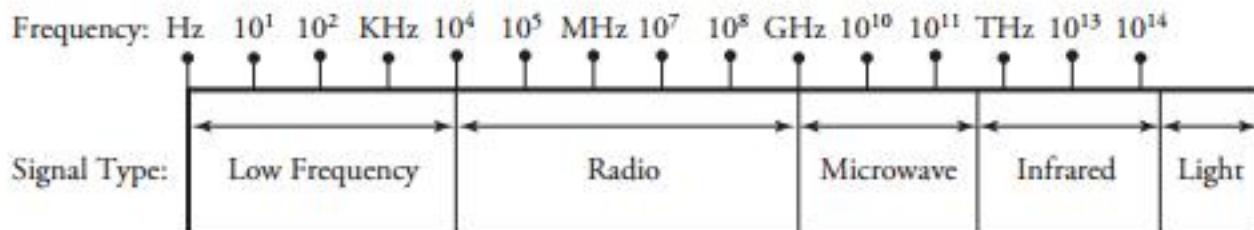


Figure 4.1 Frequency range for various data communications applications

magnetic waves, f be the frequency of the traveling signal, and λ be the wavelength of the signal. Then:

$$\lambda = \frac{c}{f}. \quad (4.1)$$

A guided link, whether wired or wireless, can be *full-duplex*, whereby two bitstreams in opposite directions can be transmitted, or *half-duplex*, whereby only one bitstream in one direction can be carried at any time. The following section explores the most commonly used data links.

4.2 Wired Links and Transmission

Wired links provide a physical path for signals to propagate. Three types of wired links are *twisted pair*, *coaxial cable*, and *optical fiber*.

4.2.1 Twisted-Pair Links

A *twisted-pair link* is the simplest form of guided medium used for data transmission. A twisted pair is normally manufactured using copper and consists of two insulated wires. The twisting action on wires reduces the cross-talk interferences generated between each two pairs of transmission links. To increase the capacity of transmission cables, especially for long-distance applications, several pairs of such links are bundled together and wrapped in a protective sheath. One of the most common applications of the twisted-pair link is for telephone network transmission links. The frequency range of twisted-pair cables is approximately 0 to 1 MHz.

4.2.2 Coaxial Cable

A higher data rate for longer-distance applications can be achieved with *coaxial cable*, a hollow outer cylindrical conductor surrounding an inner wire. The outer conductor is spaced tightly with inner wire by a solid dielectric material. The outer conductor is also shielded from outside. This concentric construction makes coaxial cables susceptible to interference. Coaxial cables have a wide variety of applications, such as cable television distribution, long-distance telephone transmission, and local area networks. The frequency range that coaxial cables can carry is 0 to 750 MHz.

4.2.3 Optical Fiber

Remarkably higher-bandwidth communication links can be achieved using optical fibers. An optical fiber is a thin glass or plastic wire that can guide an optical ray. One

of the best substances used to make optical fibers is *ultrapure fused silica*. These fibers are, however, more expensive than regular glass fibers. Plastic fibers are normally used for short-distance links where higher losses are tolerable.

Similar to coaxial cables, optical fiber cables have a cylindrical layout. The three concentrics are the *core*, the *cladding*, and the *jacket*. The core consists of several very thin fibers, each of which is surrounded by its own cladding. Each cladding also has a glass or plastic coating but different from those of the core. This difference is the key mechanism that confines the light in the cable. Basically, the boundary between the core and cladding reflects the light into the core and runs it through the cable.

The combined core and cladding is surrounded by a jacket. Jackets are made of materials that can protect the cable against interference and damage. Optical fibers are superior to coaxial cables mainly because of their higher bandwidths, lighter weights, lower signal attenuation, and lower impact by external interferences. Optical fiber links are used in all types of data communication LAN and WAN applications. The frequency range of fiber optics is approximately 180 THz to 330 THz.

4.3 Wireless Links and Transmission

Computer networks can take advantage of the wireless infrastructure where physical wires cannot be laid out. An obvious example is mobile data communication, whereby mobile users attempt to connect and stay connected to the Internet. Wireless class education is another example; an instructor teaches class through wireless media, and students can follow the lecture with their portable computers from any location within a defined vicinity.

One of the key challenges in wireless networking is the efficient utilization of the available transmission spectrum. Because the frequency spectrum available for wireless communication is normally limited, frequencies must be reused within the same geographic area. The spectrum used for wireless communications typically ranges up to several GHz. Security is also a concern in wireless networks. The open-air interface makes it difficult to prevent snooping.

The link-level design techniques involve making trade-offs among the various parameters relevant to the link layer. The optimum design would involve the use of minimum bandwidth and transmit power while maintaining a high data rate, low latency, and low bit error rates (BER). These design challenges must be achieved in the presence of channel imperfections, such as flat fading, multipath effects, shadowing, and interference.

Wireless links, both guided and unguided, are used for data communications. Wireless links use devices as an antenna for transmitting signals through *vacuum, space, air, or substances*. Electromagnetic waves can be propagated through the first three, as well as through water and wood. The frequency range depends on the type of substance. The two key challenges faced in overall design of efficient wireless links and transmission systems are the *choice of antenna* and *wireless channels*.

4.3.1 Choice of Antenna

A good antenna in a wireless system can potentially enhance the signal-to-noise ratio. Antennas are classified in several ways. The two main types of antennas used in wireless systems are *isotropic antennas* and *directional antennas*.

Isotropic Antennas

Isotropic antennas transmit signals equally in all directions. Consider an isotropic transmitter that radiates P_t watts equally in all directions, forming a sphere of flux with radius d . Given that the surface area of the sphere is $4\pi d^2$, power-flux density measured on the surface of the sphere used by a receiver located at distance d is

$$\phi_r = \frac{P_t}{4\pi d^2}. \quad (4.2)$$

At the other end of communication systems, P_r , as the captured power, depends on the size and orientation of the antenna with respect to the transmitter. If we let a be the effective area of the receiving antenna, P_t and P_r are related by

$$P_r = \phi_r a = \left(\frac{P_t}{4\pi d^2} \right) a. \quad (4.3)$$

According to electromagnetic theory, the effective area of an isotropic antenna is obtained by $a = \frac{\lambda^2}{4\pi}$. Thus, Equation (4.3) can be rewritten as

$$P_r = \frac{P_t}{\left(\frac{4\pi d}{\lambda} \right)^2}, \quad (4.4)$$

where λ is the wavelength of the signal obtained from Equation (4.1). In most propagation media other than free space, the received signal power varies inversely with d^3 or d^4 , compared to d^2 for free space.

Directional Antennas

Directional antennas are used to mitigate unwanted effects. Directional antennas amplify the signal in a small angular range but attenuate the signal at all other angles. This helps reduce the power in the various multipath components at the receiver. Directional antennas can be used to reduce the effects of interference from other users. The antenna must be accurately pointed to the correct user and must follow the user's path. This fact should lead to the development of smart antennas that can be used to track mobile users accurately by antenna steering.

4.3.2 Wireless Channels

Wireless communication is characterized by several channel impediments. A wireless channel is a portion of transmission bandwidth through which communication can be established. Channels are susceptible to interference and noise. The characteristics of a wireless channel vary with time and user movement. Most commercial wireless systems use radio waves in the ultrahigh frequency (UHF) band for communication. The UHF band ranges from 0.3 GHz to about 3 GHz. Satellite communication typically uses super-high frequency (SHF) band ranging from 3 GHz to 30 GHz. The transmitted signal reaches a receiver via three different paths: *direct*, *scattering*, and *reflection*. Signals arriving at a receiver through scattering and reflection are normally shifted in amplitude and phase. Wireless channels are characterized by four main characteristics: *path loss*, *shadowing*, *multipath fading*, and *interference*.

Path Loss

Path loss is a measure of degradation in the received signal power. The path loss depends on the transmitted power and the propagation distance. An important measure of the strength of the received signal is the signal-to-noise ratio (*SNR*). If the average noise power at the receiver is P_n , the signal-to-noise ratio is given by

$$SNR = \frac{P_r}{P_n}, \quad (4.5)$$

where we defined P_r to be the captured power in Equation (4.4). The received signal power decreases for higher frequencies, since these signals carry more power. Thus, the path loss increases with higher frequencies. The error rate in the channel is reduced

when the signal-to-noise ratio is maintained at a high level. The path loss, L_p , is obtained from Equation (4.6):

$$L_p = \left(\frac{4\pi d}{\lambda} \right)^2. \quad (4.6)$$

Note that the path loss is the ratio of transmitted power to received power.

Example. Consider a commercial wireless mobile telephone system. Assume a transmitter operating at a frequency of 850 MHz and with a power of 100 miliwatts communicates with a mobile receiver with received power of 10^{-6} microwatts. Find the distance between the transmitter and the receiver.

Solution. We can first find the path loss by

$$L_p = \frac{P_t}{P_r} = \frac{100 \times 10^{-3}}{10^{-6} \times 10^{-6}} = 10^{11}. \quad (4.7)$$

Knowing that $\lambda = c/f$, where f is the operating frequency of 850 MHz and c is the speed of light in free space estimated at 3×10^8 m/s, we can use Equation (4.6) to obtain d . The answer is $d = 8.8$ km.

Shadow Fading

As it propagates through the wireless medium, a signal encounters various obstructions, such as buildings, walls, and other objects. Physical obstructions make the transmitted signal face signal attenuation. The variation of the received signal power due to these obstructions is called *shadow fading*. In typical cases, the variation in the received signal power because of shadow fading follows a Gaussian distribution. From Equation (4.3), the received signal power seems to be the same at equal distance from the transmitter. However, even when d is the same in Equation (4.3), the received signal power varies, since some locations face greater shadow fading than do others. Normally, the transmit power P_t should be increased to compensate for the shadow-fading effect. Figure 4.2 shows the shadow-fading effect of a received signal.

Flat and Deep Fading

At each wireless receiver, the received signal power fluctuates rapidly with time and slightly with distances, a phenomenon called *flat fading*. Figure 4.2 shows a received signal power that varies with distance. The figure shows that the received signal power

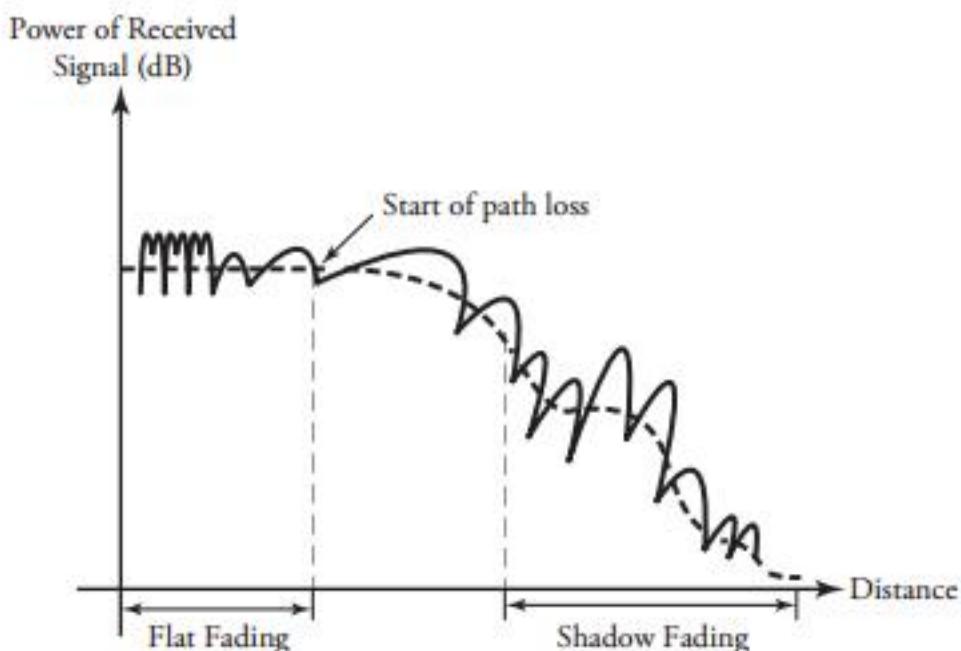


Figure 4.2 Flat fading, path loss, and shadow fading

falls from its average value. This phenomenon, whereby the received signal power falls below the value required to meet link-performance constraints, is called *deep fading*.

Doppler Frequency Shift

Let v_r be the relative velocity between a transmitter and a receiver. The shift in the frequency of the transmitted signal caused by the relative velocity is called *Doppler shift* and is expressed by f_D :

$$f_D = \frac{v_r}{\lambda}, \quad (4.8)$$

where λ is the wavelength of the transmitted signal. As the relative velocity changes with time, the Doppler shift also varies. In frequency modulations, this shift could result in an increase in the bandwidth of the signal. In most scenarios, the Doppler shift can be of the order of several Hz, whereas the signal bandwidth is of the order of several KHz. Thus, the effect of Doppler shift is negligible in these cases.

Interference

The limited frequency spectrum in wireless channels leads to frequency reuse at spatially separated locations. Frequency reuse can lead to *interference*. The interference can be reduced by using more complex systems, such as dynamic channel allocation, multiuser detection, and directional antennas. Interference can also result from adjacent

channels if they occupy frequencies outside their allocated frequency bands, although interference can be reduced by having a guard band between channels. Interference can also result from other users and other systems operating in the same frequency band. Certain filters and spread-spectrum techniques are used to eliminate this type of interference.

4.3.3 Capacity Limits of Wireless Channels

Claude Shannon derived an analytical formula for the capacity of communication channels. The capacity of a channel in bits per second is given by

$$C = B \log_2(1 + SNR), \quad (4.9)$$

where B is the channel bandwidth, and SNR is the signal-to-noise ratio at the receiver. Shannon's formula gives only a theoretical estimate and assumes a channel without shadowing, fading, and intersymbol interference effects. For wired networks, Shannon's formula gives a good estimate of the maximum achievable data rates. For wireless channels, the achievable data rate is much lower than the one suggested by Shannon's formula. The reason is that the channel characteristics vary with time, owing to shadowing, fading, and intersymbol interference.

4.3.4 Channel Coding

Channel coding is a mechanism used to make channels immune to noise and to correct errors introduced by the channel. This process involves adding some redundant bits to the transmitted information. These redundant bits can be used for error detection and correction. The use of channel coding can eliminate the need for retransmissions when channel-errors occur. The redundant bits caused can be used to correct the errors and thereby reduce the transmit power and achieve a lower BER.

Forward error correction (FEC) is a commonly used scheme for channel coding. FEC schemes normally increase the signal bandwidth and lower the data rate. The *automatic repeat request* (ARQ) scheme explained in previous chapters is normally used along with FEC, as FEC is not sufficient for implementing channel coding. *Turbo codes* have also been successful in achieving data rates near Shannon's capacity. Turbo codes, however, are very complex and have large delays.

4.3.5 Flat-Fading Countermeasures

The common techniques used to combat flat fading are *diversity, coding and interleaving*, and *adaptive modulation*. With *diversity* multiple independent fading paths are

combined at the receiver to reduce power variations. These independent fading paths can be obtained by separating the signal in time, frequency, or space. Space diversity is one of the most commonly used and effective diversity techniques. An antenna array is used to achieve independent fading paths. Antenna elements in the array are spaced at least one-half wavelength apart.

Coding and interleaving is another technique used to counter flat fading. In general, flat fading causes errors to occur in bursts. With coding and interleaving, these burst errors are spread over multiple code words. The adjacent bits from a single code word are spread among other code words to reduce the burst of errors, because burst errors affect adjacent bits. The code words passed to the decoder of the interleaving process contain at most one bit error. FEC channel coding can be used to correct these errors.

Adaptive modulation schemes adjust to channel variations. The transmission scheme adapts to the varying channel conditions, based on an estimate that is sent back to the transmitter. The data rate, transmit power, and coding scheme are tailored, based on the received channel estimate. The channel estimate varies, depending on the amount of flat fading. These adaptive schemes help reduce BER and increase efficiency. The adaptive schemes do not function properly if a channel cannot be estimated or if the channel characteristics change very rapidly. It should be noted that the feedback scheme for conveying the channel estimate to the transmitter requires additional bandwidth.

4.3.6 Intersymbol Interference Countermeasures

The techniques used to combat *intersymbol interference* (ISI) can be classified into signal-processing techniques and antenna solutions. The signal-processing techniques, which attempt to compensate for ISI or reduce the influence of ISI on the transmitted signal, include equalization, multicarrier modulation, and spread-spectrum techniques. The antenna solutions attempt to reduce ISI by reducing the delay between the multipath components and include directive beams and smart antennas.

The equalization method compensates for ISI at the receiver through channel inversion. The received signal is passed through a linear filter with inverse frequency response, making ISI zero. The noise has to be reduced before passing the signal through the inverse filter. This is done by a linear equalizer called the minimum mean square equalizer. Given a large variation in the channel frequency response, nonlinear *decision-feedback equalizer* (DFE) is used. DFE uses the ISI information from the previously detected symbols to achieve equalization.

DFE is more complex and achieves a much lower BER. Other equalization techniques are the maximum-likelihood sequence and turbo equalization. These schemes

perform better than DFE but are much more complex. The equalizer techniques require an accurate channel estimate to compensate correctly for ISI. As a result, equalizer techniques may not work well for channels in which the characteristics change rapidly.

Multicarrier modulation is another technique used to reduce the effect of ISI. The transmission bandwidth is divided into a number of narrow slots called subchannels. The message signal containing the information to be transmitted is also divided into an equal number of slots. Each of these slots is modulated on one of the subchannels. The resulting sequence is transmitted in parallel. The subchannel bandwidth is maintained less than the coherent bandwidth of the channel. This results in a flat fading instead of a frequency-selective fading in each channel, thereby eliminating ISI. The subchannels can be either nonoverlapping or overlapping. The overlapping subchannels are referred to as *orthogonal frequency division multiplexing* (OFDM). This technique improves the spectral efficiency but results in a greater frequency selective fading, thereby decreasing the signal-to-noise ratio.

4.3.7 Orthogonal Frequency Division Multiplexing (OFDM)

In *orthogonal frequency division multiplexing* (OFDM), transmitters generate both the carrier and the data signal simultaneously. OFDM is not a multiple-access technique, as there is no common medium to be shared. The entire bandwidth is occupied by a single source of data. Instead of being transmitted serially, data is transmitted in parallel over a group of subcarriers. Since the carriers are orthogonal to one another, the nulls of one carrier coincide with the peak of another subcarrier, resulting in the possibility of extracting the subcarrier of interest. Contrary to the traditional FDM technique, the spectrum of channels in OFDM significantly overlap.

The process of digital signal generation used in OFDM is based on inverse *fast Fourier transform* (FFT) (see Figure 4.3). A *serial-to-parallel converter* (S/P) converts the serial bits to be transmitted into parallel format. The number of subcarriers and the type of digital communication technique used determine the typical number of bits in parallel. Thus, the incoming data is divided into a large number of carriers. Then *map* and *demap* act as digital modulator and demodulator, respectively. Typically, a *quadrature amplitude modulation* (QAM) or *quadrature phase shift keying* (QPSK) is used as modulator.

The heart of this technique is an inverse *fast Fourier transform* (FFT), in which a carrier frequency is generated based on the location of the stored value. This produces a set of time-domain samples. At the receiver, individual subchannels can be completely separated by FFT only when there is no intersymbol interference in an *add cyclic prefix*.

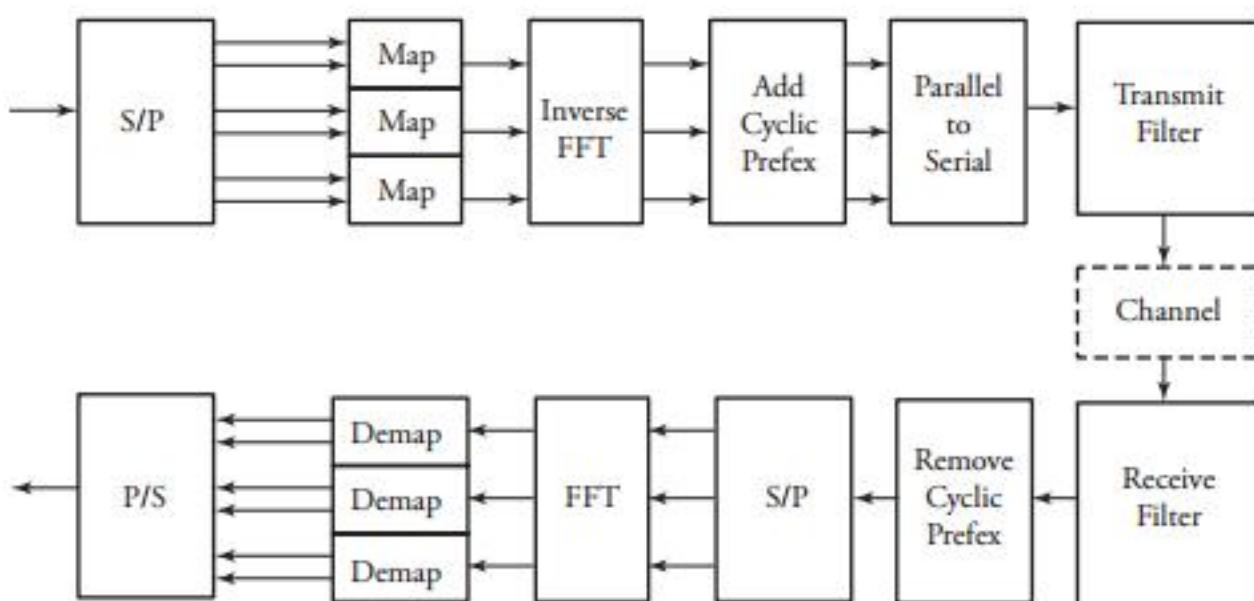


Figure 4.3 A block diagram of the OFDM technique

acting as a band guard. Note that as the signal travels through the medium, the signal arrives at the receiver at different instances, owing to multipath propagation, resulting in a delay spread leading to ISI. A simple solution to overcome ISI is to widen the symbol duration. This can be achieved by increasing the number of carriers, causing the distortion to become insignificant.

At the receiver, the time waveform is digitized and converted back to a symbol, using FFT. The incoming carrier signal is tracked coherently and sampled in order to be sent to FFT. The time-domain samples are needed at FFT in order to extract the amplitude and phase of the signals. Depending on the amplitude and phase extracted over one symbol time, the frequency-domain representation of the signals obtained. The results are demapped into their original bitstreams, depending on the digital demodulator applied. OFDM signal reception involves a challenging task of time-frequency domains, sampling, and clock synchronization, as well as channel estimations.

4.4 Methods of Channel Access on Links

Voice and video applications that run on wireless networks could require continuous transmission, requiring dedicated allocation of available resources for that application. The sharing of the available bandwidth among applications requiring dedicated resources is called *multiple access*. But most wireless applications involve transmission of random bursty data, requiring some form of random channel allocation, which does

not guarantee resource availability. *Multiple-access* techniques assign bandwidth to users by allocating a portion of available spectrum to each independent user.

4.4.1 Frequency-Division Multiple Access

In *frequency-division multiple access* (FDMA), the available bandwidth is divided into nonoverlapping, or orthogonal, slots. In other words, each user is assigned a portion of the channel spectrum. FDMA is the simplest multiple-access mechanism that supports multiple users.

4.4.2 Time-Division Multiple Access

In *time-division multiple access* (TDMA), each user is assigned a time slot for transmitting. These time slots are nonoverlapping. TDMA techniques are more complex, as they require synchronization in timing among the users. TDMA is also affected by intersymbol interference because of channel distortions, such as multipath delay effects. TDMA divides each channel into orthogonal slots and hence limits the number of users, based on the available bandwidth. TDMA places a hard limit on the number of supported users and bandwidth available for each user. A practical version of TDMA is the *random-access technique* and is mainly used in local area networks.

Random Access Techniques

Most wireless networks carry traffic in a bursty form. The dedicated resource allocation schemes, such as multiple-access techniques, prove to be inefficient for some types of systems under bursty traffic. With *random-access* schemes, channels are accessed at random. *Aloha-based* and *reservation-based* protocols are two well-known random-access techniques that require packets to be acknowledged. Distortions in the wireless channel of these schemes, however, may result in loss or delay of acknowledgments. In such cases, a packet retransmission is required, which in turn makes the process inefficient. One solution to this problem would be to use smarter link-layer techniques for the acknowledgment packets to increase their reliability. Common objectives of performing channel access are as follows.

- To minimize interference from other users, a transmitter listens before any transmission.
- To give fair access of the spectrum to all other users, a transmitter transmits for only a certain period of time.
- To minimize transmitter power, the same frequency could be reused in farther areas.

In the basic Aloha scheme, each transmitter sends data packets whenever it has data available to send. This naturally leads to a large number of collisions, and hence a number of data packets have to be retransmitted. Hence, the effective throughput of the Aloha channel is very low because the probability of packet collisions is high. The slotted Aloha scheme was developed to deal with the collision problem. In slotted Aloha, the time is divided into slots, and packet transmission is restricted to these time slots. Thus, the number of collisions is reduced significantly. The throughput with slotted Aloha is double that with basic Aloha. Spread-spectrum techniques are used in combination with Aloha to support a large number of users.

Collision detection and carrier sensing are very difficult in a wireless environment. Shadow-fading effects impair collision detection, as objects obstruct the direct signal path between users. This difficulty of detecting collisions in a wireless environment is often referred to as the *hidden-terminal problem*. Path loss and shadow-fading effects result in signals being hidden between users. Thus, collision-avoidance schemes are normally used in wireless networks, especially in wireless LANs. In a collision-avoidance scheme, the receiver sends a busy tone on receiving a packet. This busy tone is broadcast to all the nearby transmitters. A transmitter that receives a busy tone from any receiver refrains from transmitting. Once the busy tone ends, the transmitter waits for a random amount of time before sending packets. This random back-off scheme is used to prevent all transmitters from transmitting at the same time when the busy signal ends. The collision-avoidance schemes help reduce the collisions in Aloha channels and thus significantly improve the throughput of Aloha.

Reservation-based schemes assign channel to users on demand. The effective channel bandwidth is divided between the data channel and the reservation channel. Users reserve channel bandwidth on the reservation channel and send the data packets on the data channel. Users send small packets along the reservation channel, requesting access to the data channel. If a data channel is available, the request is accepted, and a message is sent to the user. Thus, an overhead in reservation-based schemes is the assignment of the data channel. But these data channels are assigned only on demand. For networks on which only small messages are exchanged, the overhead may be tolerable. Also, when the traffic in the network increases rapidly, the reservation channel may get congested with request messages.

The *packet-reservation multiple-access* (PRMA) scheme combines the benefits of the Aloha and the reservation-based schemes. In PRMA, time is slotted and organized into frames, with N time slots per frame. A host that has data to transmit competes for an available time slot in each frame. Once a host successfully transmits a packet in a time slot, the slot is reserved for the user in each subsequent frame until the user has no

more packets to transmit. When the user stops transmitting, the reservation is revoked, and the user has to compete for a time slot to send more packets. PRMA is used in multimedia applications.

4.4.3 Code-Division Multiple Access

In *code-division multiple access* (CDMA), users use both time and bandwidth simultaneously, modulated by spreading codes. Spreading codes can be either orthogonal (nonoverlapping) or semiorthogonal (overlapping). Orthogonal spreading codes make it possible to recover the signal at the receiver without any interference from other users. But orthogonal spreading codes place a hard limit on the number of supported users, such as FDMA and TDMA. Semiorthogonal spreading codes involve some interferences from other users at the receiver side. The fundamental superiority of CDMA over other channel-access method lies in the use of the *spread-spectrum technique*.

Spread-Spectrum Technique

The *spread-spectrum technique* involves spreading frequencies of a transmitted signal over a wider range. This technique reduces flat fading and intersymbol interference. The message signal is modulated by a *pseudonoise signal*, which encompasses a wider bandwidth. Hence, the resultant transmission signal obtains a much larger bandwidth.

Spread-spectrum techniques can be implemented in two ways. In the first one, *direct sequence*, the message signal is Exclusive-ORed, with the pseudonoise sequence thereby spreading the frequency range. The second technique, *frequency hopping*, involves using the pseudonoise sequence to transmit over a range of frequencies. Frequency hopping is formed based on the random pseudonoise sequence.

Rake Receiver

The *rake receiver*, shown in Figure 4.4, is used in a CDMA system where multipath effects are common. The binary data to be transmitted is first Exclusive-ORed with the transmitter's chipping code to spread the frequency range of the transmitted signal. The signal is then modulated for transmission over the wireless medium. The multipath effects in the wireless medium result in multiple copies of the signal, each with different delays of t_1 , t_2 , and t_3 and with different attenuations of α_1 , α_2 , and α_3 . The receiver demodulates the combined signal and then introduces a variable delay for each signal. Signals are then combined with different weighing factors— β_1 , β_2 , and β_3 —to give the resultant signal with reduced multipath effects.

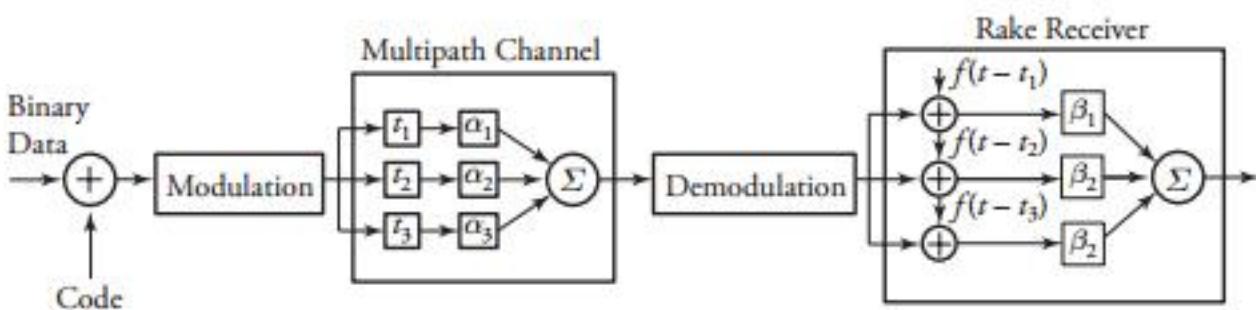


Figure 4.4 The rake receiver at the heart of CDMA

Semiorthogonal CDMA is the most complex type of multiple-access scheme. But in this technique, the mobile units located far from the receiver experience high interference from other users and hence poor performance. Some power-control schemes are used to mitigate this problem by equalization. The semiorthogonal schemes have the advantage that they do not place any hard limit on the number of users supported. But owing to the semiorthogonal codes, the interference from other users increases as the number of users becomes large. However, the interference can be reduced by using steering antennas, interference equalization, and multiuser detection techniques.

In CDMA, the available frequency bandwidth for each cell is divided in half: one for forward transmission between the base station to the mobile unit and the other part for reverse transmission from the mobile unit to the base station. The transmission technique used is called *direct-sequence spread spectrum* (DSSS). Orthogonal chipping codes are used to increase the data rates and to support multiple users. The transmitted signal also has an increased bandwidth. Using CDMA for cellular systems has several advantages:

- *Diversity in frequency.* In CDMA, the transmitted signal occupies a wider range of frequencies. Therefore, the transmitted signal is not greatly affected by noise and selective fading.
- *Multipath effects.* The orthogonal chipping codes used in CDMA have low cross-correlation and autocorrelation. Hence, multipath signals delayed by more than one chip interval do not interfere with the dominant signal.
- *Privacy.* Since DSSS techniques use pseudorandom sequences, privacy is ensured.
- *Scalability.* FDMA or TDMA systems support only a fixed number of users. CDMA can support a larger number of users with an acceptable amount of degradation in performance. The error rate increases gradually as the number of users becomes larger.

CDMA also has certain disadvantages. In CDMA, the spreading sequences of different users is not orthogonal, and there is some overlap, which results in some cross-correlation, resulting in self-jammering. Also, signals, being at a greater distance from the receiver, experience significant attenuation compared to signals close to the receiver. Thus, signal strength is weak for remote mobile units.

4.4.4 Space-Division Multiple Access

The three multiple-access techniques FDMA, TDMA, and CDMA are based on *isotropical antennas*. Recall that an isotropical antenna operates essentially in a uniform manner in all directions. Another method of multiple access in wireless systems is *space-division multiple access* (SDMA), which uses smart antennas. At one end of communication system, a directional antenna, typically known as a smart antenna, can focus directly on the other end of the system. This technique offers a number of advantages, such as reduction of transmission power, reduced amount of interference owing to reduced transmission power, and the strong signal received by the receiver, owing to the high-gain antenna.

4.4.5 Hybrid Multiple-Access Techniques

An effective multiple-access scheme needs to be carefully chosen on the basis of application requirements. Practical wireless systems normally use two or more multiple-access techniques. This strategy provides a reasonable growth plan and compatibility with existing systems. Multiple-access methods can also be combined to better serve certain applications. The two most commonly used hybrid schemes are FDMA/TDMA and FDMA/CDMA. Other forms of CDMA are so-called W-CDMA and TD-CDMA. W-CDMA provides higher data rates and uses the spectrum in an efficient manner. TD-CDMA combines W-CDMA and TDMA.

4.5 Error Detection and Correction

Error sources are present when data is transmitted over a medium. Even if all possible error-reducing measures are used during the transmission, an error invariably creeps in and begins to disrupt data transmission. Any computer or communication network must deliver accurate messages.

Error detection is applied mostly in the data-link layer but is also performed in other layers. In some cases, the transport layer includes some sort of error-detection scheme. When a packet arrives at the destination, the destination may extract an error-checking

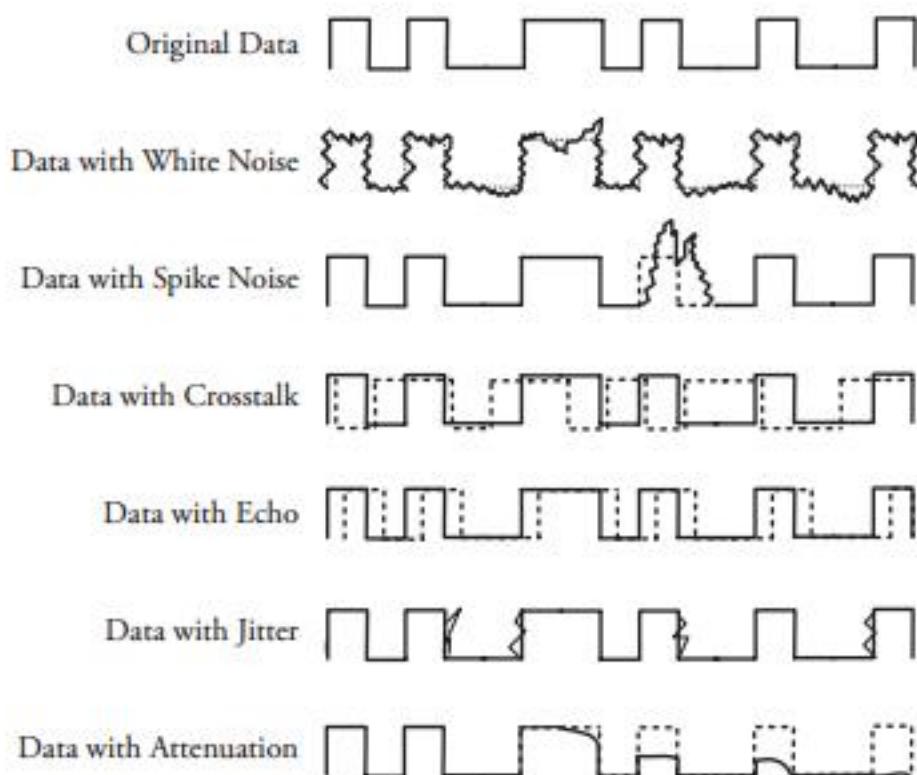


Figure 4.5 Common forms of data errors at the data-link level

code from the transport header and perform error detection. Sometimes, network-layer protocols apply an error-detection code in the network-layer header. In this case, the error detection is performed only on the IP header, not on the data field. At the application layer, some type of error check, such as detecting lost packets, may also be possible. But the most common place to have errors is still the data-link layer. Possible and common forms of errors at this level are described here and are shown in Figure 4.5.

- *White, or Gaussian, noise* is continuous and is dependent on the temperature of the medium. White noise might change the content of data, as seen in the figure. White noise can be removed by passing the noisy signal through a set of filters.
- *Spike noise* is not continuous but may completely obliterate the data, so that it cannot be recovered.
- *Cross talk* is a coupling action between two active links. Coupling can be electrical, as between two sets of twisted-pair wire, or electromagnetic, as when unwanted signals are picked up by an antenna.
- *Echo* is the reflecting impact of a transmitted signal. A signal can hit the end of a cable and bounce back through the wire, interfering with the original signal. This error occurs in bus-type LANs. A one-directional filter, known as an *echo canceler*, can be attached to a link to eliminate echo.

- *Jitter* is a timing irregularity that shows up at the rises and falls of a signal, causing errors. Jitter can result from electromagnetic interference or cross talk and can be reduced by proper system shielding.
- *Bit attenuation* is the loss of a bit's strength as it travels through a medium. This type of error can be eliminated with the use of amplifiers and repeaters for digital systems.

No link is immune to errors. Twisted-pair copper-based media are plagued by many types of interference and noise. Satellite, microwave, and radio networks are also prone to noise, interference and cross talk. Fiber-optic cable too may receive errors, although the probability is very low.

4.5.1 Error Detection Methods

Most networking equipment at the data-link layer inserts some type of error-detection code. When a frame arrives at the next hop in the transmission sequence, the receiving hop extracts the error-detection code and applies it to the frame. When an error is detected, the message is normally discarded. In this case, the sender of the erroneous message is notified, and the message is sent again. However, in real-time applications, it is not possible to resend messages. The most common approaches to error detection are

- Parity check
- Cyclic redundancy check (CRC)

The *parity check* method involves counting all the 1 bits in the data and adding one extra bit, called the *parity bit*. This makes the total number of 1 bits even (even parity) or odd (odd parity). The parity-check method is the simplest error-detection technique but is not effective. The *cyclic redundancy check* (CRC) method is one of the most elaborate and practical techniques but is more complex, adding 8 to 32 check bits of error-detection code to a block of data. Our emphasis is CRC.

At this point, we need to clarify that the *Internet checksum* is another error-detection method, though it is used at the network and transport layers and is discussed in Chapters 7 and 8.

4.5.2 Cyclic Redundancy Check (CRC) Algorithm

The *cyclic redundancy check* (CRC) method provides smart error checking and has been adopted in most computer and communication systems. Figure 4.6 shows error detection and correction on links, using CRC for a transmitter.

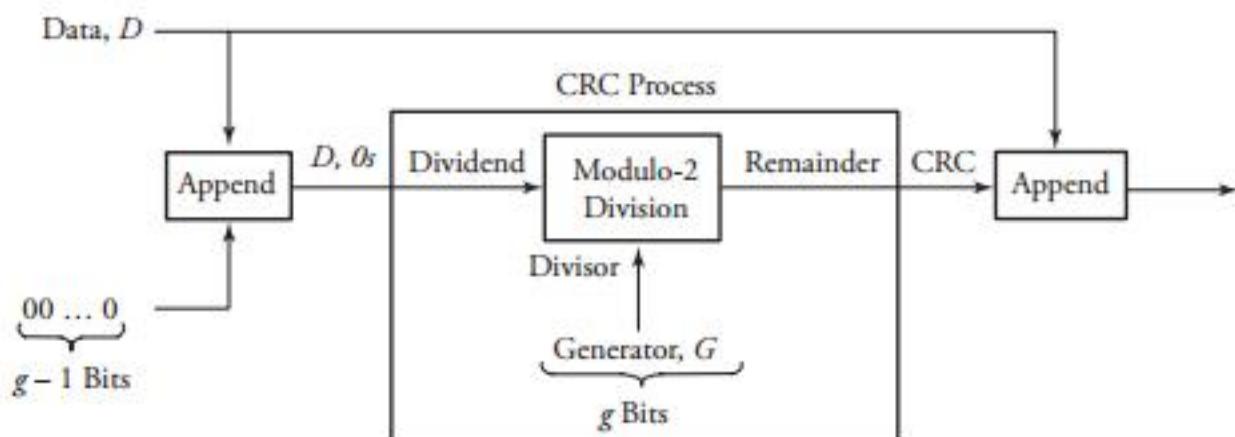


Figure 4.6 Error detection on links using the CRC method at a transmitter

In any system, a standard and common value between transmitters and receivers is adopted for error processing. This g -bit-long value, known as a checking *generator*, is denoted by G . At the transmitter, a partial or entire packet or frame is treated as a block of data, and each block is processed individually. The CRC algorithm at the transmitter part can be summarized as follows.

Begin CRC Algorithm at Transmitter

1. A string of $g - 1$ zero bits is appended to the incoming data, D . We call this new block $D,0s$.
2. $D,0s$ as a dividend is divided by the generator G acting as the divisor. The division is of type *modulo-2*.
3. The quotient of this division is discarded, but the remainder is called CRC.
4. The CRC value is appended to the data, producing D,CRC . ■

At the other end point of the communication system, the receiver receives the value of D,CRC and performs the following algorithm to detect errors, as shown in Figure 4.7.

Begin CRC Algorithm at Receiver

1. D,CRC , as a dividend, is divided by the same generator G acting as the divisor used by the transmitter. The division is of type *modulo-2*.
2. The quotient of this division is discarded, but if the remainder is 0, the receiver knows that the data has no errors; otherwise, the data is not accepted, as it contains one or more errors. ■

The modulo-2 division arithmetic is very simple. A modulo-2 division function is done without carries in *additions* or borrows in *subtractions*. Interestingly, the modulo-2 division function performs exactly like the Exclusive-OR logic. For example,

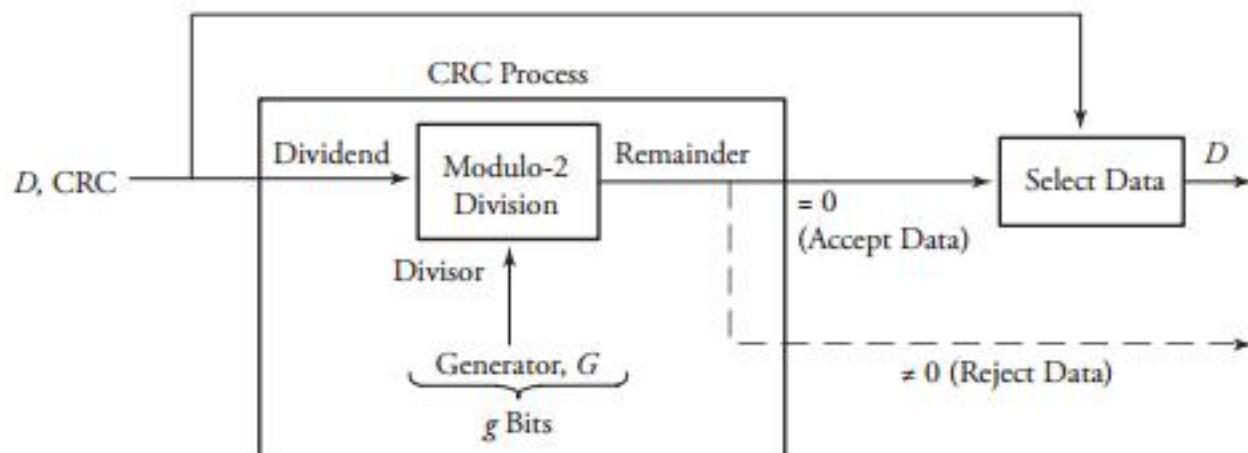


Figure 4.7 Error detection on links, using CRC at a receiver

with modulo-2 arithmetic, we get $1 + 1 = 0$ and $0 - 1 = 1$. Equivalently, with logic Exclusive-OR: $1 \oplus 1 = 0$, and $0 \oplus 1 = 1$.

Example. Assume that 1010111 as a block of data (D) is going to be transmitted using the CRC error-checking method. Suppose that the common value of the generator, G , is 10010. Produce the final value that the transmitter sends on the link (D, CRC), and show the detail of the error-detection process at the receiver.

Solution. At the transmitter, clearly, the number of 0s needed to be appended to D is four ($g - 1 = 4$), since the generator has 5 bits ($g = 5$). Figure 4.8 (a) shows the details of the CRC process at the transmitter side. Since $D = 1010111$, $D, 0s = 10101110000$, the dividend. The divisor is $G = 10010$. Using modulo-2 arithmetic, the quotient turns out to be 1011100, and the remainder is $\text{CRC} = 1000$. Therefore, the transmitter transmits $D, \text{CRC} = 1010111, 1000$. At the receiver, as shown in Figure 4.8 (b), $D, \text{CRC} = 1010111, 1000$ is treated as the dividend and is divided by the same divisor, $G = 10010$. Since the remainder in this case is 0, the receiver learns that there is no error in the data and can extract the data.

The CRC algorithms are fairly straightforward. Consider again the example. A fact from the modulo-2 arithmetic states that the remainder is always ($g - 1$) bits long; therefore, for this example, the remainder is 4 bits long. Hence, if we make sure that $g - 1$ additional 0s, or four 0s in the example, are at the end of the dividend, the receiver can perform an identical arithmetic in terms of size of dividend. If the transmitter produces a remainder, let's say 1,000, the addition of this value to the same data can indeed result in 0 remainder, provided that the data is the same data used in the transmitter. Therefore, if there is any error during transmission, the remainder may not become 0 at the receiver, giving a notion of error.

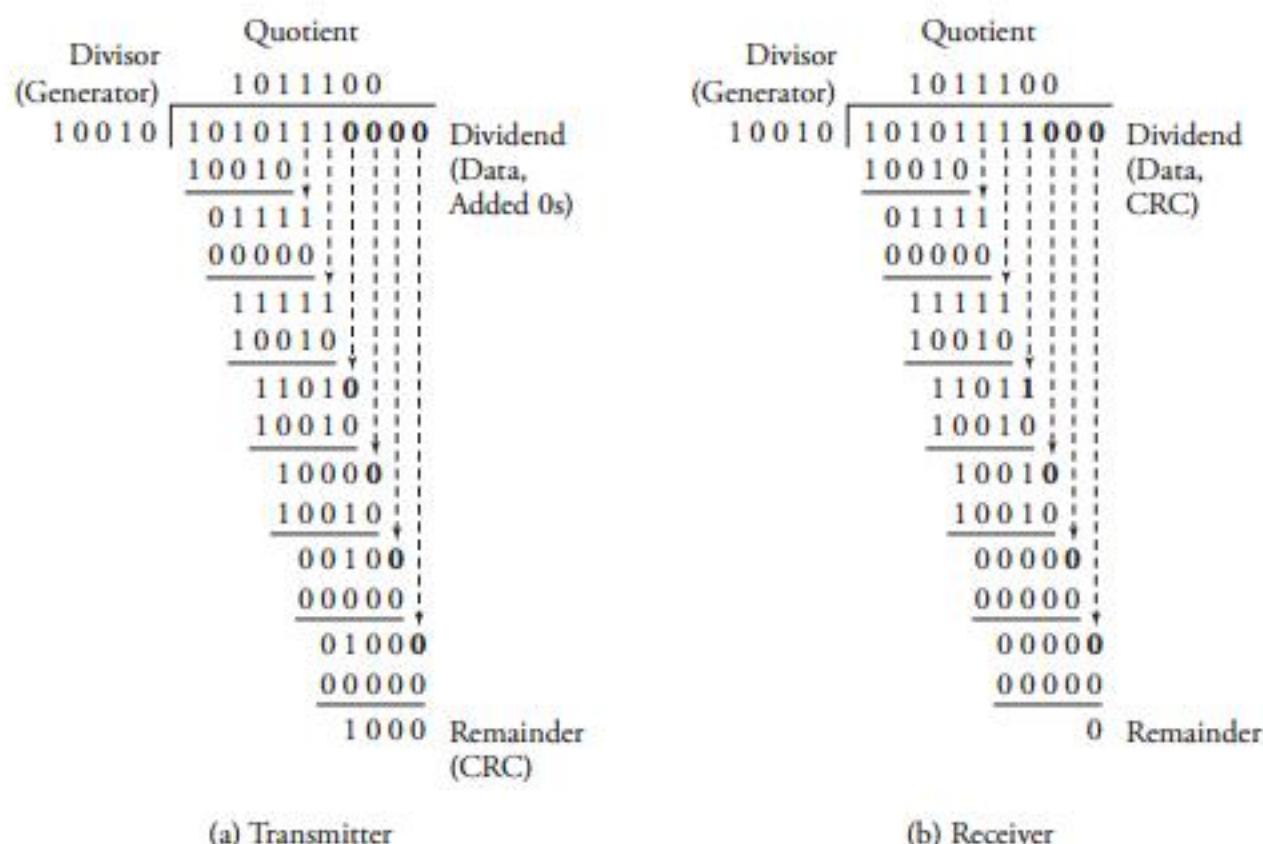


Figure 4.8 Modulo-2 division for the CRC process shown for a transmitter and a receiver

Equivalent Polynomial Interpretation

The preceding analogy can be restated such that the CRC error-detection method treats the data to be transmitted as a polynomial. In general, consider the bit string $a_{n-1}a_{n-2}a_{n-3}\dots a_0$, which forms a generic polynomial:

$$a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + a_{n-3}x^{n-3} + \dots + a_0x^0,$$

where a_i can be 0 or 1. In other words, when a data bit is 1, the corresponding polynomial term is included. For example, the bit string $D = 1010111$ produces the string $a_6a_5a_4a_3a_2a_1a_0 = 1010111$, which is interpreted as

$$x^6 + x^4 + x^2 + x^1 + x^0.$$

The *generator* value, acting as the divisor, is known as the *generating polynomial* in this case. Some well-known and widespread industry-approved generating polynomials—

common divisor between transmitters and receivers—used to create the cyclic checksum remainder are:

CRC-8 for ATM: $x^8 + x^2 + x + 1$

CRC-12: $x^{12} + x^{11} + x^3 + x^2 + x + 1$

CRC-16: $x^{16} + x^{15} + x^2 + 1$

CRC-CCITT: $x^{16} + x^{15} + x^5 + 1$

CRC-32 used in IEEE 802:

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

Note that the polynomial interpretation is simply a convenient method of explaining the CRC algorithm. In practice, the transmitter and the receiver perform divisions in bits, as described earlier.

Effectiveness of CRC

The CRC method is pretty goof-proof. However, an analysis is needed to answer whether the receiver is always able to detect a damaged frame. Consider the generating polynomial $x^{g-1} + x^{g-2} + \dots + 1$ with g terms. Apparently, $g - 1$ is the highest power of the generating polynomial. Let n be the length of burst error occurring in a received message. If the size of the error burst is $n < g$, error detection is 100 percent. For all other cases, the terms between x^{g-1} and 1 define which bits are erroneous. Since there are $g - 2$ such terms, there are 2^{g-2} possible combinations of erroneous bits. Considering that all combinations can occur with equal probability, there is a chance of $\frac{1}{2^{g-2}}$ that a combination exactly matches the terms of the polynomial. This is the probability of a damaged bit becoming undetected. The probability of catching such error bursts through CRC for all cases is

$$p = \begin{cases} 1 & \text{if } n < g \\ 1 - \left(\frac{1}{2}\right)^{(g-2)} & \text{if } n = g \\ 1 - \left(\frac{1}{2}\right)^{(g-1)} & \text{if } n > g \end{cases} \quad (4.10)$$

Example. Consider a computer communication standard based on CRC-CCITT defined earlier. For this standard, the highest power of the polynomial is $g - 1 = 16$. If the error burst n is less than $g = 17$ bits in length, CRC detects it. Assuming that $n = g = 17$:

$$p = 1 - \left(\frac{1}{2}\right)^{(17-2)} = 0.999969,$$

which is close to 1.

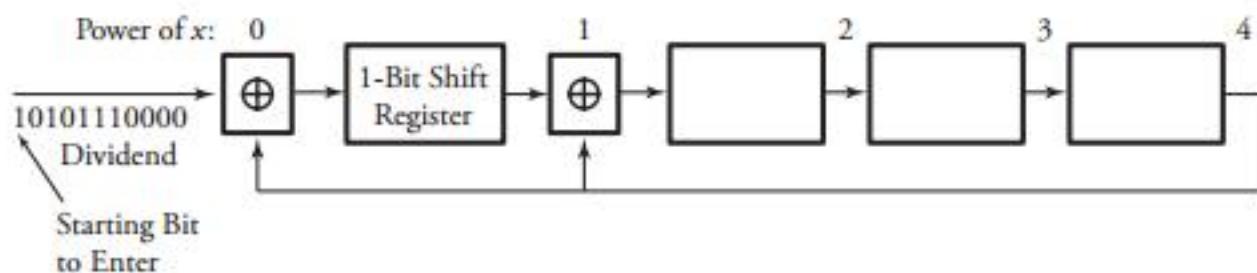


Figure 4.9 CRC process: The most significant bit enters first.

Implementation of the CRC Process Unit

Hardware with a combination of software performs the process of division very quickly. An example of the basic hardware used to perform the CRC calculation is shown in Figure 4.9. The hardware includes a simple register that implements the CRC process-generating polynomial $x^4 + x + 1$. Except for the first term (x^4), an Exclusive-OR is included for each power of existing term, as shown. Note that the notion of the generator value 10010 is now appearing in the form of hardware shown by a combination of 1-bit shift registers and Exclusive-OR gates. Therefore, we can see from the figure where there is a term in the generating polynomial.

Initially, the registers contain 0s. All data bits of 1010111,0000, beginning from the most-significant bit, arrive from the left, and a 1-bit shift register shifts the bits to the right every time a new bit is entered. The rightmost bit in a register feeds back around at select points. At these points, the value of this feedback bit is Exclusive-ORed, with the bits shifting left in the register. Before a bit shifts right, if there is an Exclusive-OR to shift through, the rightmost bit currently stored in the shift register wraps around and is Exclusive-ORed with the moving bit. Once all data bits are fed through, the register's contents must be exactly the same as the remainder indicated in Figure 4.8 (a).

4.6 Link-Level Flow Control

Communication systems must use *flow-control techniques* on their transmission links to guarantee that a transmitter does not overwhelm a receiver with data. Several protocols guarantee the control of link flows. Two widely used flow-control protocols are *stop and wait* and *sliding window*.

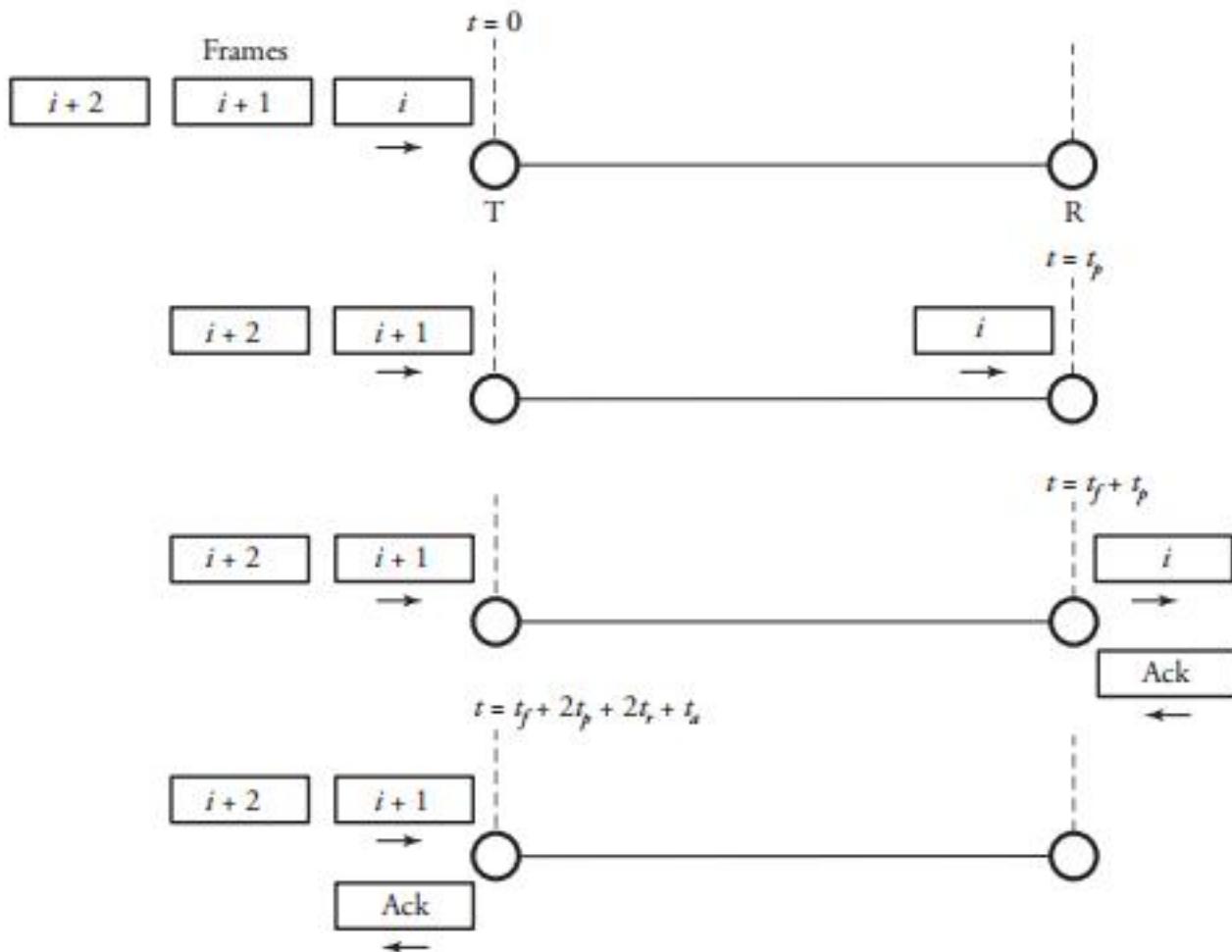


Figure 4.10 A simple timing chart of a stop-and-wait flow control of data links

4.6.1 Stop-and-Wait Flow Control

The *stop-and-wait* protocol is the simplest and the least expensive technique for link-overflow control. The idea behind this protocol is that the transmitter waits for an acknowledgement after transmitting one frame (see Figure 4.10). The essence of this protocol is that if the acknowledgement is not received by the transmitter after a certain agreed period of time, the transmitter retransmits the original frame.

In this figure, we assume two consecutive frames i and $i + 1$. Frame i is ready to enter the link and is transmitted at $t = 0$. Let t_f be the time required to enter all the bits of a frame and t_p the propagation time of the frame between the transmitter (T) and the receiver (R). It takes as long as $t = t_f + t_p$ to transmit a frame. At the arrival of a frame i , the receiver processes the frame for as long as t_r and generates an acknowledgment. For the same reason, the acknowledgment packet takes $t = t_a + t_p$

to be received by the transmitter if t_a is assumed to be the time required to enter all the bits of an acknowledgment frame, and it takes t_r for processing it at the receiver. Therefore, the total time to transmit a frame, including acknowledgment processes,

$$t = t_f + 2t_p + 2t_r + t_a. \quad (4.11)$$

Note here that with this technique, the receiver can stop or slow down the flow of data by withholding or delaying acknowledgment. Practically, t_r and t_a are negligible compared to other components of the equation. Thus, this equation can be approximated as

$$t \approx t_f + 2t_p. \quad (4.12)$$

In this equation, $t_f = \ell/r$, where ℓ is the length of frame in bits; r is the data rate; and $t_p = d/v$, where d is the length of transmission line, and v is speed of transmission. For wireless and wired transmissions, $v = 3 \times 10^8$ m/s where except for fiber-optic links, $v \approx 2.2 \times 10^8$ m/s, owing to the fact that light travels zigzag over links, and thus overall speed is lower. Therefore, the link efficiency is defined as

$$E_\ell = \frac{t_f}{t} = \frac{1}{1 + 2 \left(\frac{t_p}{t_f} \right)}. \quad (4.13)$$

4.6.2 Sliding-Window Flow Control

The shortcoming of the stop-and-wait protocol is that only one frame at a time is allowed for transmission. This flow control can be significantly improved by letting multiple frames travel on a transmission link. However, allowing a sequence of frames to be in transit at the same time requires a more sophisticated protocol for the control of data overflow on links. The well-known *sliding-window* protocol is one technique that efficiently controls the flow of frames.

Figure 4.11 shows an example of sliding-window flow control. With this protocol, a transmitter (T) and a receiver (R) agree to form identical-size sequences of frames. Let the size of a sequence be w . Thus, a transmitter allocates buffer space for w frames, and the receiver can accept up to w frames. In this figure, $w = 5$. The transmitter can then send up to $w = 5$ frames without waiting for any acknowledgment frames. Each frame in a sequence is labeled by a unique sequence number. For every k frames forming a sequence, the transmitter attaches a sequence-number field to each frame. Therefore, as many as 2^k sequence numbers exist.

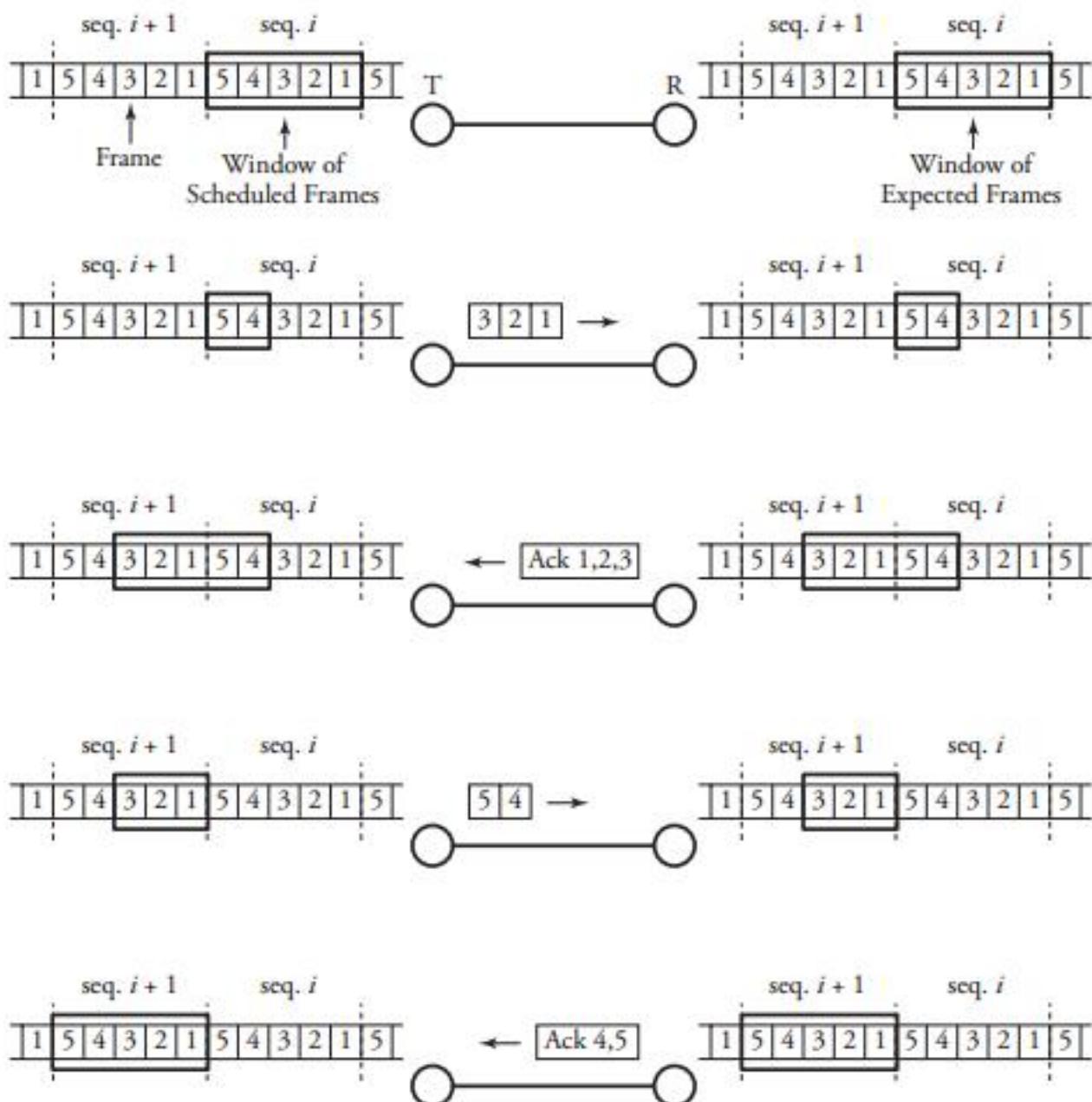


Figure 4.11 Timing chart of a sliding-window flow control for two sequences of frames

First, a transmitter opens a window of size $w = 5$, as the figure shows. The receiver also opens a window of size w to indicate how many frames it can expect. In the first attempt, let's say that frames 1, 2, and 3 as part of sequence i are transmitted. The transmitter then shrinks its window to $w = 2$ but keeps the copy of these frames in its buffer just in case any of the frames is not received by the receiver. At the receipt of these three frames, the receiver shrinks its expected window to $w = 2$ and acknowledges the receipt of frames by sending an acknowledgment ACK1,2,3 frame to the transmitter.

At the release of ACK1,2,3, the receiver changes its expected window size back to $w = 5$. This acknowledgment also carries information about the sequence number of the next frame expected and informs that the receiver is prepared to receive the next w frames. At the receipt of ACK1,2,3, the transmitter maximizes its window back to $w = 5$, discards the copies of frames 1, 2, and 3, and continues the procedure. In this protocol, the window is imagined to be sliding on coming frames. Similarly, we can derive an expression for this protocol as we did for the stop-and-wait protocol. For the transmission of a frame as the integral portion of w frames sequence, the total time, including all the acknowledgment processes, is obtained using Equation (4.11), with averaging over w and inclusion of wt_f as

$$t = \frac{1}{w}(wt_f + 2t_p + 2t_r + t_a). \quad (4.14)$$

Practically, t_r and t_a are negligible; thus, this equation can be approximated by

$$t \approx t_f + 2 \left(\frac{t_p}{w} \right). \quad (4.15)$$

Therefore, link efficiency is expressed as

$$E_\ell = \frac{t_f}{t} = \frac{w}{w + 2 \left(\frac{t_p}{t_f} \right)}. \quad (4.16)$$

Link efficiency provides a network designer with a good understanding of the link utilization for network management purposes.

4.7 Summary

We started this chapter by discussing wired and wireless transmission media. Data can travel on guided links, such as optical fibers, and unguided links, such as certain types of wireless links. Link capacity is further partitioned into *channels*. Wireless channels have several weaknesses, such as shadow fading, path loss, and interference. Methods of accessing link channels deal with how to mediate access to a shared link so that all users eventually have a chance to transmit their data. We examined frequency-division, time-division, code-division, and space-division multiple-access methods; in most cases, time-division multiple-access methods offer several benefits for channel access in local area networks.

We also looked at methods that determine whether transferred bits are in fact correct or whether they possibly were corrupted in transit. With the *cyclic redundancy check*

(CRC) method, some frames arriving at the destination node contain errors and thus have to be discarded.

Two link-control schemes are *stop and wait* and *sliding window*. In the stop-and-wait method, a sender waits for an acknowledgment after transmitting one frame. This flow-control method is significantly improved in the *sliding window* method, which lets multiple frames travel on a transmission link.

In the next chapter, we use our knowledge of data links from Chapters 2, 3, and 4 to form small local area networks.

4.8 Exercises

1. In order to transmit a 500-page book with an average of 1,000 characters per page between places 5,000 km apart, we assume that each character uses 8 bits, that all signals travel at the speed of light, and that no link-control protocol is used.
 - (a) How much time is required if a digital voice circuit operating at the speed of 64 kb/s is used?
 - (b) How much time is required if a 620 Mb/s fiber-optic transmission system is used?
 - (c) Repeat parts (a) and (b) for a library with 2 million volumes of books.
2. Assume that a wireless system with 200 terminals uses TDMA for its channel access. The packet lengths are T in average and are considered short compared with the TDMA long channel length. Compare the efficiency of two strategies: *polling* and CSMA.
3. Design a CRC process unit for the following two standard generators of computer networking:
 - (a) CRC-12
 - (b) CRC-16
4. For the example presented in the CRC section, we had 1010111 as a block of data (D), and the common value of generator, G , 10010, as the divisor.
 - (a) Show the dividend and the divisor in polynomial forms.
 - (b) Divide the dividend and the divisor in polynomial forms.
 - (c) Compare the results of part (b) to its binary form obtained in example.
5. For data $D, \text{CRC} = 1010111, 1000$, presented in the example of the CRC section, and the common value of generator, G is 10010 as the divisor. Sketch a picture of

Figure 4.9 as many as needed and every time you shift in a bit, show the content of each register. Prove that the final contents of the registers show the value of CRC.

6. Assume that 101011010101111 is a block of data (D) to be transmitted using the CRC error-checking method. Suppose that the common value of generator, G , is 111010. Using modulo-2 arithmetic.
 - (a) Produce the final value that the transmitter sends on the link (D, CRC).
 - (b) Show the detail of the error-detection process at the receiver.
7. Consider a coaxial transmission link that uses the stop-and-wait protocol requiring a propagation time to transmission time ratio of 10. Data is transmitted at rate 10 Mb/s, using 80-bit frames.
 - (a) Calculate the efficiency of this link.
 - (b) Find the length of this link.
 - (c) Find the propagation time.
 - (d) Sketch a plot of link efficiency when the ratio of propagation time to transmission time is reduced to 8, 6, 4, and 2.
8. Consider a 2 Mb/s satellite transmission link through which 800-bit frames are transmitted. The propagation time is 200 ms.
 - (a) Find the link efficiency, using the stop-and-wait protocol.
 - (b) Find the link efficiency, using the sliding-window protocol if the window size is $w = 6$.
9. Consider the bidirectional control of links with the sliding-window method applied between two routers R2 and R3 (window size $w = 5$) and stop-and-wait control between routers R3 and R4. Assume that the distance between R2 and R3 is 1,800 km and is 800 km between R3 and R4. Data frames with average size of 5,000 bits flow from R2 to R3 at 1 Gb/s rate. Acknowledgment frames are small enough to be ignored in the calculations. All links generate 1 $\mu\text{s}/\text{km}$ propagation delay.
 - (a) Determine a condition on the data rate at the output port of R3 toward R4 so that R3 remains congestion-free.
 - (b) Find the link efficiency for the R2–R3 link.
 - (c) Find the link efficiency for the R3–R4 link.

CHAPTER 5

Local Area Networks and Networks of LANs

A *local area network* (LAN) is a small interconnection infrastructure that typically uses a shared transmission medium. Because of such factors as the volume of traffic, the level of security, and cost, the network structure in a local area network can be significantly different from that for a wide area network. This chapter focuses on the fundamentals of local area networks and describes the *internetworking* concept at the LAN level. major topics are as follows:

- *Basic LAN topology*
- *LAN protocols*
- *MAC and IP addresses*
- *Classification of MAC protocols*
- *Contention-access MAC*
- *Round-robin-access MAC*
- *Network of LANs*

First, we explore some simple topologies of local area networks and see how a LAN is formed. We then extend the discussion of protocols presented in Chapter 2 to LAN protocols, focusing on *medium access control* (MAC) and addressing. We explore two well-known methods of link-access methods: *contention* and *round-robin*.

Another important topic is *internetworking*. Some pointers toward internetworking LANs with repeaters and bridges are provided. The focus of this chapter is primarily on layer 2 of the protocol stack reference model.

5.1 LANs and Basic Topologies

A LAN is used for communications in a small community in which resources, such as printers, software, and servers, are shared. Each device connected to a LAN has a unique address. Two or more LANs of the same type can also be connected to forward data frames among multiple users of other local area networks. In LANs, packets are additional headers appended for local routing. These new-looking packets are known as *frames*. Users in a local area network can be interconnected in several ways. The fundamental network topology in LANs can be categorized into *bus*, *ring*, and *star*, as shown in Figure 5.1.

In the *bus* topology, all users are connected to a common transmission medium referred to as a bus. The users are connected to a common bus via a duplex link that allows both uplink and downlink operations, as seen in the figure. The transmission from a user is propagated on the bus in both directions, and all users in the path receive its frame. However, only the destination user copies the frame into its computer; all other users discard the frame.

The *ring* topology comprises of layer 2 devices called *repeaters* (see Section 3.3.1). Repeaters are interconnected to form a closed loop, and each user is connected to one repeater, shown in the figure by a smaller circle. When a user transmits a frame, its associated repeater forwards the frame to the ring. The ring is normally unidirectional, so a frame can flow in one direction. During the circulation of the frame in the ring, the destination user copies the frame onto its buffer. Once copied by the destination, the frame continues its circulation until the sender receives it and removes it from the system. To avoid collision, only one user can transmit at a given time.

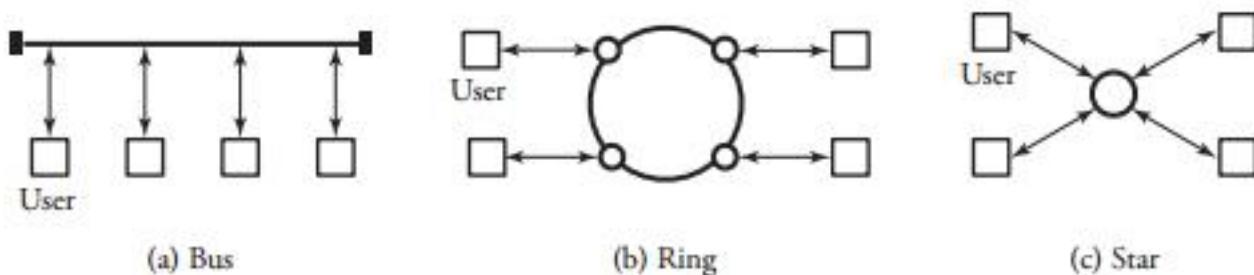


Figure 5.1 Three fundamental LAN configurations to connect users

In the *star* topology, all users are directly connected to a central user through two unidirectional links: one for uplink and the other for downlink. The central user functions in either broadcast mode or frame-switch mode. In *broadcast mode*, the central user is called a *hub* (see Section 3.3.1). When it receives the frame from a user on the uplink, the hub retransmits the frame to all users on the downlink. Broadcast mode allows only one user to transmit at a time. In *frame-switch mode*, the central user buffers the received frame and retransmits the frame only to the destination.

5.2 LAN Protocols

Protocols designed for layers 3 and above are independent of the underlying network topology. Hence, protocols designed for LANs are normally concerned with the data-link layer and the physical layer. Organizations working on LAN standards comply with the specifications of IEEE 802 reference model. The physical layer of a LAN implements such functions as signal transmission and reception, encoding and decoding, and generation and removal of synchronization information. The physical layer also specifies the type of medium used for transmission and the network topology.

Figure 5.2 shows the position of the two LAN sublayers in the overall structure of the protocol stack. The IEEE 802 standard subdivides the *data-link layer* of the protocol model into the *logical-link control* (LLC) layer and the *media access control* (MAC) layer. The LLC layer implements flow and error control apart from providing an interface to the network layer. The MAC layer primarily controls the access to transmission medium and is responsible for framing. LLC has the option of choosing from various types of MAC layers.

5.2.1 Logical-Link Layer (LLC)

Data to be transmitted from the higher layers is passed down to the *logical-link layer*, which determines the mechanism for addressing users across the medium. The LLC also

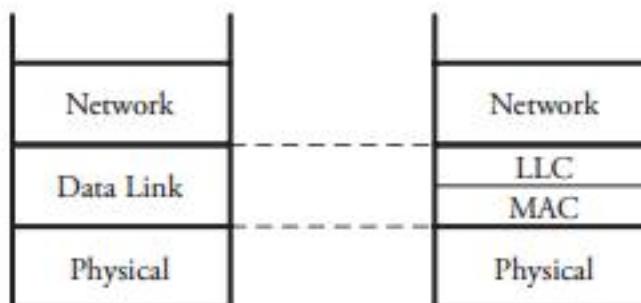


Figure 5.2 LAN sublayer protocols in the overall structure of the protocol stack

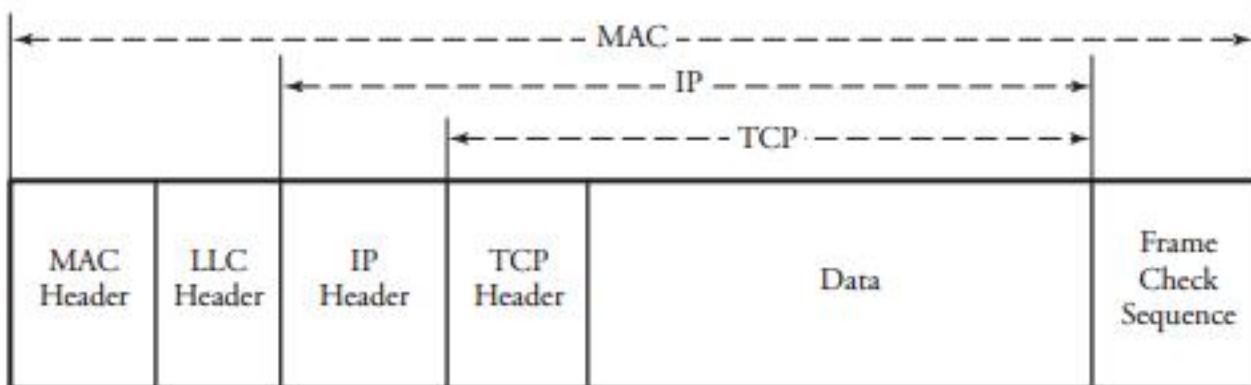


Figure 5.3 Generic MAC frame format

controls the exchange of data between each two users. The LLC appends its header to form the LLC protocol data unit, which is then sent to the MAC layer, which appends the header and the frame check sequence to create the MAC frame.

5.2.2 Medium Access Control (MAC)

A LAN is required to provide sharing access to the transmission medium. To ensure efficient access to a medium, users have to comply with some rules. The MAC protocol manages access to the medium. Figure 5.3 shows a generic MAC frame format within frame formatting for the MAC protocol. The fields of the frame format are as follows.

- *MAC header* gives the MAC control information, such as the MAC address of the destination and the priority level.
- *LLC header* contains the data from the logical-link layer.
- *IP header* specifies the IP header of the original packet.
- *TCP header* specifies the TCP header of the original packet.
- *Frame check sequence* is used for error checking.

The next section provides a detailed discussion on MAC and its interaction with IP addresses.

5.3 MAC and IP Addresses

Each node has an IP address. Each node's adapter to its attached link has a link-layer address, which is in fact the MAC *address*. A MAC address is as wide as 6 bytes and is normally shown in hexadecimal notation, such as 00-40-33-25-85-BB. The MAC address is unique for each device and is permanently stored in the adapter's read-only

memory. Consequently, networking manufacturers need to purchase MAC addresses for their products. Unlike an IP address, a MAC address is not hierarchical. The advantage of MAC addressing is that a device may not need to have an IP address in order to communicate with the surrounding devices in its own LAN.

Each node adapter that receives a frame checks whether the MAC address of the frame matches its own MAC address. If the addresses match, the adapter extracts the inner packet and passes it up the protocol stack. In summary, each destination address specified in an IP header is the logical address and is different from the physical or the link-layer address. For a packet from a source host to be delivered successfully to its destination, the host needs to identify both the IP address and the link-layer address. The source uses the *address resolution protocol* (ARP) to find the link-layer address of a destination.

5.3.1 Address Resolution Protocol (ARP)

The *Address Resolution Protocol* (ARP) is designed to convert IP addresses to MAC addresses or vice versa. Suppose that a user wants to transmit a packet to its destination. If it does not know the link-layer address of the destination, the sender broadcasts an ARP packet requesting the link-layer address given the IP address. The destination is denoted by its IP address in the ARP request. Only the destination replies with its link-layer address. The sender then stores this address in the local ARP table for its subsequent use. Each networking device, such as a host or a router, has an interface, or *adapter*. Each adapter has a table that keeps the MAC address of each device within the network it is attached to. MAC addresses are listed in the adapter's ARP table.

Example. Figure 5.4 shows that LAN1 has several users. A user with IP address 133.176.8.55 and adapter MAC address AB-49-9B-25-B1-66 wants to send a frame within its LAN to user 133.176.8.56 with adapter MAC address 11-40-33-55-A3-57. In this case, original packets are converted to frames in its adapter with the destination address AB-49-9B-25-B1-66. Now suppose that the same sender wants to send frames outside its LAN to a user in LAN3 with IP address 198.34.7.25 connected through a four-port router. But assume that the sender does not know the MAC address of the destination. In this case, the sender must broadcast an ARP packet to all users, but only the destination user replies to the ARP query, providing its adapter MAC address of 33-BA-76-55-A3-BD. This new address is used for the router to forward the message of the sending user to the destination.

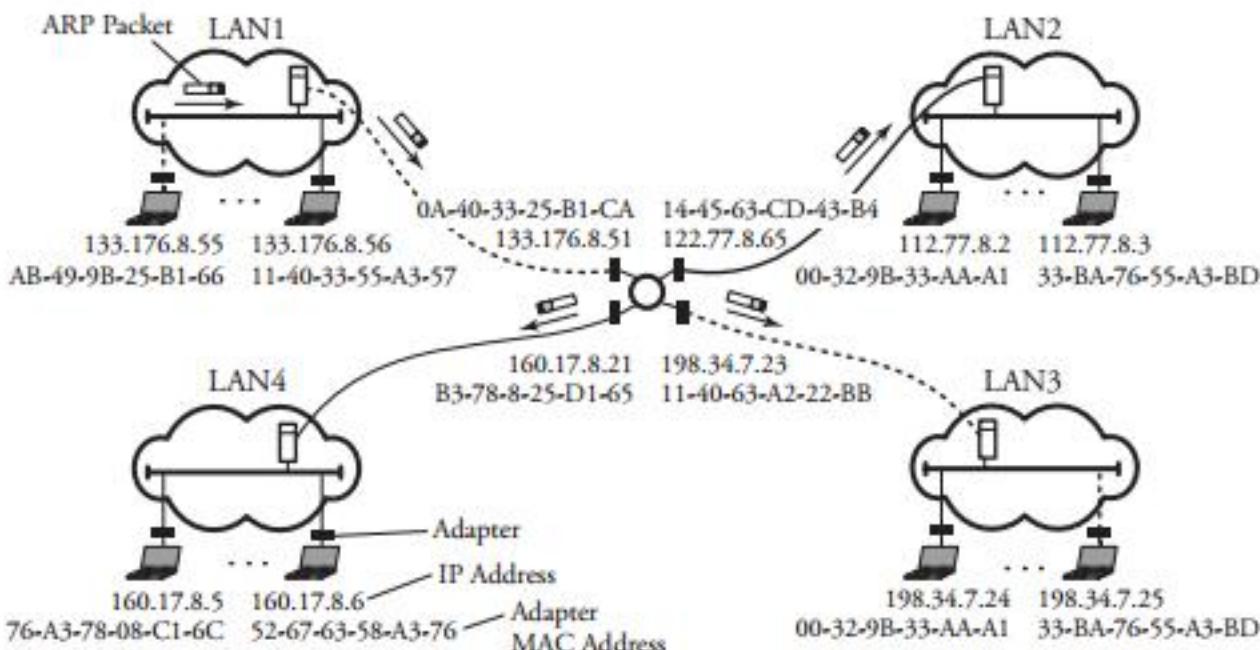


Figure 5.4 ARP packets and MAC and port adapters

5.3.2 Reverse Address Resolution Protocol (RARP)

Reverse Address Resolution Protocol (RARP) too is a protocol to find a certain address. But in this case, RARP is used when the link-layer address of the destination is known but not its IP address. The host broadcasts the link-layer address of the destination and requests the IP address. The destination responds by sending the IP address.

5.4 Classification of MAC Protocols

MAC protocols can be broadly classified as *centralized* or *distributed* based on the type of network architecture. In the centralized scheme, a central controller controls the operation of the users, and users communicate only with the controller. Users adhere to the transmission schedule specified by the controller. In a distributed network, users talk to each other and dynamically determine the transmission schedule. Centralized schemes are used in applications that require a simple access scheme, QoS, and guaranteed capacity. However, centralized schemes are vulnerable to a single point of failure. Distributed schemes have no single point of overall failure. Yet, coordinating access among devices is a complex process for distributed schemes, and the network cannot provide a QoS guarantee.

MAC protocols can also be characterized as *synchronous* or *asynchronous*. In the synchronous scheme, such as time-division multiplexing and frequency-division mul-

tiplexing, capacity is preassigned to each connection. The synchronous scheme is rarely used in LANs, as users' transmission requirements vary dynamically. Rather, it is optimal to assign capacity to users asynchronously, based on demand.

In terms of access to the shared medium, MAC protocols can be broadly classified into two types:

- *Contention access*, by which each user needing to transmit data must contend to access the medium
- *Round-robin access*, by which each user is given a chance to transmit data in a round-robin fashion

The details of these two protocols are presented in Sections 5.5 and 5.6.

A third category of access method is *reservation-access* MAC. This application-specific method is used mainly for streaming traffic. With this method, as in synchronous time-division multiplexing, time slots are used to access the medium. A user can reserve time slots for transmission in advance. The control mechanism for the reservations can be either centralized or decentralized.

5.5 Contention-Access MAC

Contention-access MAC is stochastic, more suitable for bursty traffic, whereby the waste of shared-medium bandwidth cannot be tolerated. Unlike the round-robin and reservation schemes, no control is involved in the contention scheme. Rather, each user needing to transmit frames must contend to access the medium. This scheme is simple to implement and is an effective solution for light or moderate traffic.

When a user logs on to a LAN, a channel for the transmission of a frame can be demanded at any random instance, potentially resulting in frame collision. This serious issue can be resolved in a number of ways, as follows:

- *Permanent assignment of one channel to each user*. This method can clearly waste the system bandwidth.
- *Checking users regularly*. The system can poll each user on a regular basis to see whether it has anything to transmit. This method could result in a long delay for larger networks.
- *Random access of a channel*. The system can provide a mechanism whereby a user can access at any time. This method is efficient but faces the collision issue if two or more frames are transmitted at the same time.

Several random-access methods are provided for computer networks. Two commonly used ones are *Aloha* method (see Section 4.4.2) and *Carrier Sense Multiple Access* (CSMA). The CSMA scheme is explained in the next section.

5.5.1 Carrier Sense Multiple Access (CSMA)

Carrier Sense Multiple Access (CSMA) is a protocol that lets only one user at a time transmit, on a first come, first served basis. A user needing to transmit data first listens to the medium and senses for a carrier on the medium to determine whether other users are transmitting: the *carrier-sense* phase. If the medium is busy, the user has to wait until the medium is idle. The amount of time a user must wait depends on the particular type of the protocol. If no other users transmit data, the user proceeds to transmit its data onto the medium. However, when the medium is busy, a user can follow one of the following three approaches:

1. *Nonpersistent CSMA*. The user waits for a random amount of time after a collision before sensing the channel again. The random delay is used to reduce the collision. However, this scheme uses the transmission channel inefficiently: Even though the transmission is completed, the user rechecks the medium only after expiration of the random delay. The random wait time is normally $512 g$ bit times, where g is a number drawn randomly.
2. *1-persistent CSMA*. This scheme overcomes the disadvantage of the nonpersistent CSMA by continuing to sense the channel while it is busy. As soon as the medium is idle, the user transmits immediately. In this scheme, a collision occurs if more than one user is waiting to transmit.
3. *p-persistent CSMA*. This scheme is a compromise between the nonpersistent and the 1-persistent methods. When the medium is idle, the user can transmit with a probability p . If the medium is busy, the user can transmit for a time equal to the maximum propagation delay. Deciding on an appropriate value of p is crucial for efficient operation of the scheme.

If two or more users simultaneously try to transmit, a collision of frames occurs, and all data is corrupted. In such a case, all corresponding users stop transmitting. Thus, after a collision, all the users involved will start contention, each after a randomly chosen amount of time. After finishing the transmission of data, a user waits for an acknowledgment from the destination user. If the acknowledgment does not arrive, the user retransmits the data.

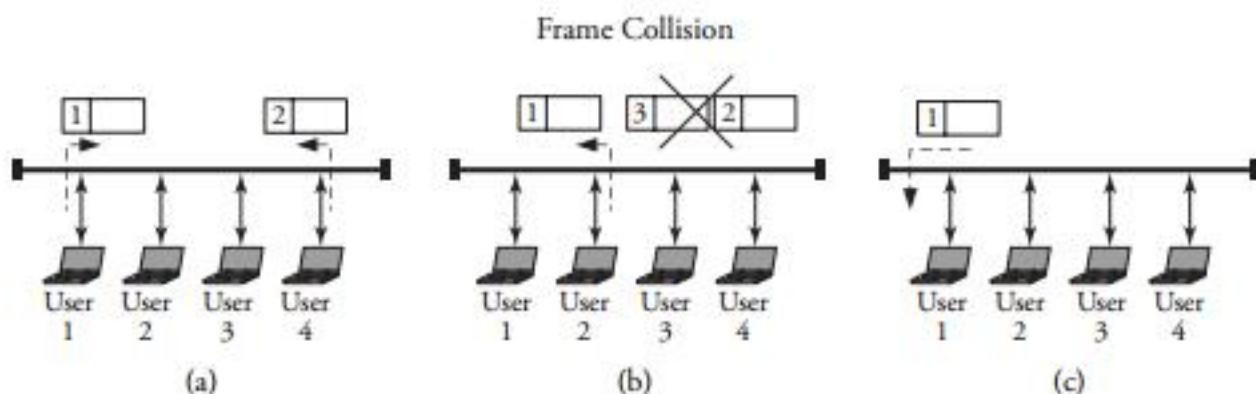


Figure 5.5 The movement and collision of frames in a contention-access MAC network

The main disadvantage of CSMA when a collision occurs is that other users cannot use the medium until all corrupted frames finish transmission. This problem increases in the case of long frames. However, this issue can be resolved with the use of CSMA/CD, whereby a user listens to the channel while transmitting data. In case of a collision, the transmission is halted, and a jamming signal is transmitted to inform the other users that a collision has occurred. The user enters into *back-off mode* and transmits after the back-off duration if the medium is idle. In Figure 5.5, a frame that leaves user 4 destined for user 2, collides with another frame that leaves user 1 destined for user 3. Immediately after the collision, user 2 finds the medium idle and transmits its frame destined to user 1.

Consider n users to be connected onto a common cable so that when one user transmits a frame while others are silent, all other users sense that frame. When a user starts the transmission, no others start before the user has propagated a signal throughout the cable. This way, the user can finish its frame without collision. The CSMA/CD method can be set up to detect collisions ahead of time. It is quite possible for a user to listen to the cable while transmitting; obviously, if two or more users start to transmit simultaneously, they find a collision in process and cease transmission. That is why this process is called CSMA collision detection (CSMA/CD).

CSMA/CD requires the frame length to be long enough to permit collision detection before the completion of transmission. The maximum time to detect a collision is not greater than twice the end-to-end propagation delay. The process of CSMA/CD is viewed in terms of slots and minislots. Setting minislots is required for a signal to propagate from one end of the cable to the other. Minislots are used in a contention mode. If all users are synchronized in minislots and one user transmits during an empty slot, all the other users pick up the transmitted frame until the frame is transmitted

completely. However, if more than one user transmits during that slot, each transmitting user senses the situation and ceases transmitting.

Analysis of Frame Delay

Consider once again the bus LAN shown in Figure 5.5. Let ℓ be the average distance between any two users. The maximum ℓ is the distance between user 1 and user 4, and the minimum ℓ is the distance between user 1 and user 2. Let c be the speed of propagation. The average propagation delay of a frame between any two users can be determined by

$$t_p = \frac{\ell}{c}. \quad (5.1)$$

Now, let f be the average frame size in bits and r be the data rate on the channel over the medium. The frame transmission time, t_r , can be calculated by

$$t_r = \frac{f}{r}. \quad (5.2)$$

Here, the *utilization* of the LAN bus can be obtained by

$$u = \frac{t_r}{t_r + t_p}. \quad (5.3)$$

Analysis of Contention

If a total of n users are attached to the medium and n_a of them are active, the probability of a user restraining itself to resend its frame during a time slot is

$$p = \frac{1}{n_a}. \quad (5.4)$$

In general, an empty time slot remains empty, is taken by a user, or undergoes a collision. Apparently, the probability that a user attempts to transmit frames in an empty time slot is

$$p_c = \binom{n_a}{1} p^1 (1-p)^{n_a-1} = n_a p (1-p)^{n_a-1}. \quad (5.5)$$

By combining the two Equations (5.4) and (5.5), we obtain

$$p_c = \left(\frac{n_a - 1}{n_a} \right)^{n_a-1}. \quad (5.6)$$

This probability value merges to $\frac{1}{e}$ when n_a approaches a very large number. A different situation is that a frame tries i times in i empty slots to transmit its frame and is unsuccessful owing to collision, but it successfully sends its frame on time $i + 1$. The probability that this situation happens is obviously modeled by a geometric random variable, explained in Appendix C, and is obtained by

$$p_i = p_c(1 - p_c)^i. \quad (5.7)$$

Interestingly, the average number of contentions can be computed by knowing this behavioral model using the expected value explained in Appendix C:

$$\begin{aligned} E[C] &= \sum_{i=1}^{\infty} i p_i = \sum_{i=1}^{\infty} i p_c (1 - p_c)^i \\ &= \frac{1 - p_c}{p_c}. \end{aligned} \quad (5.8)$$

Analysis of Throughput

Consider Figure 5.6, and assume that the frame arrivals on the network have a Poisson distribution (see Appendix C for the details of the Poisson model) with an average arrival rate λ and a frame duration of T seconds. Based on this model, the probability that there are k frames in a given period $[0, t]$ is obtained from

$$P_X(k) = \frac{\lambda^k e^{-\lambda}}{k!}. \quad (5.9)$$

We start with frame 2 and let t_p be the total propagation delay between any pair of users. Thus, the probability that a frame is transmitted successfully, P_t , is the probability that no additional frame is transmitted during t_p and is expressed by Equation (5.9) when $k = 0$ as

$$P_t = P_X(0) = e^{-\lambda t_p}. \quad (5.10)$$

The delay time caused by the last colliding frame is modeled by a random variable, Y . The probability that this last frame is transmitted at or before time y is the probability that no other frames are transmitted in the interval $(y, t_p]$ and is obtained from

$$P[Y \leq y] = e^{-\lambda(t_p - y)}. \quad (5.11)$$

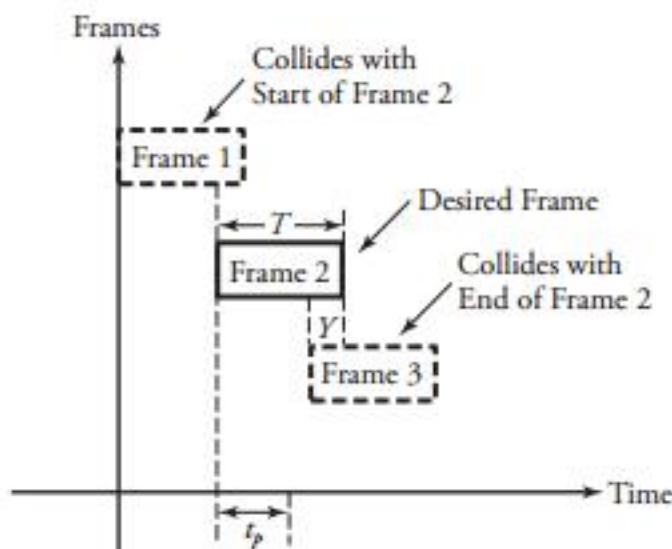


Figure 5.6 A timing illustration of CSMA/CD

The preceding probability is clearly calculated only for $0 < y < t_p$ and is known as the *cumulative distribution function* (CDF) of the random variable y . If this distribution is known, the *probability density function* (PDF), or $f_Y(y)$, can be determined (see Appendix C); thus, the average of time y can be estimated by the expected value of y :

$$E[Y] = \int_0^{t_p} y f_Y(y) dy = t_p - \frac{1 - e^{-\lambda t_p}}{\lambda}. \quad (5.12)$$

As seen in Figure 5.6, the average *busy time* of a channel, t_B , has three components: frame duration (T), propagation delay (t_p), and the relative delay of the last colliding frame ($E[Y]$):

$$t_B = T + t_p + E[Y] = T + 2t_p - \frac{1 - e^{-\lambda t_p}}{\lambda}. \quad (5.13)$$

From the property of the Poisson random variable, we calculate the average idle time as

$$t_I = \frac{1}{\lambda}. \quad (5.14)$$

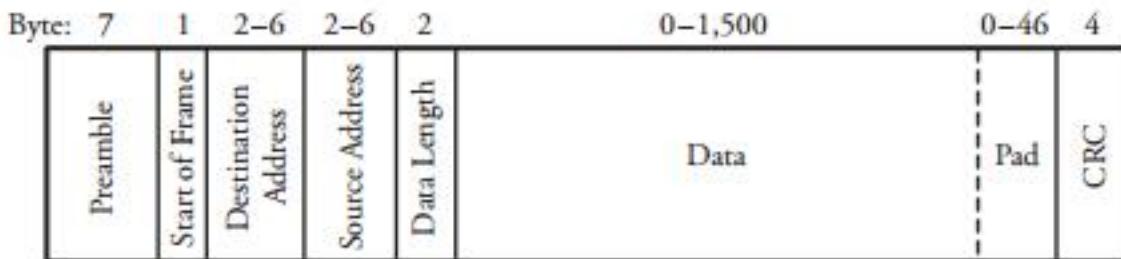


Figure 5.7 Details of Ethernet IEEE 802.3 LAN frame

The overall throughput of a CSMA/CD channel is defined as the number of frames per time slot and is obtained from

$$\begin{aligned}
 R &= \frac{P_t}{t_B + t_I} \\
 &= \frac{e^{-\lambda t_p}}{\left(T + 2t_p - \frac{1-e^{-\lambda t_p}}{\lambda} \right) + \frac{1}{\lambda}} \\
 &= \frac{\lambda e^{-\lambda t_p}}{\lambda(T + 2t_p) + e^{-\lambda t_p}}. \tag{5.15}
 \end{aligned}$$

Throughput R can be normalized to throughput per frame time, or R_n . Practically, R_n is easier to use for the estimation of the system throughput.

5.5.2 Ethernet LAN: IEEE 802.3 Standard

The IEEE 802.3 standards committee developed a widely used LAN standard called *Ethernet*, which covers both the MAC layer and the physical layer. The IEEE 802.3 standard uses CSMA for controlling media access and the *1-persistent* algorithm explained earlier, although the lost time owing to collisions is very small. Also, IEEE 802.3 uses a back-off scheme known as *binary exponential backoff*. The use of random backoff minimizes subsequent collisions. This back-off scheme requires a random delay to be doubled after each retransmission. The user drops the frame after 16 retries. The combination of the 1-persistent scheme and binary exponential backoff results in an efficient scheme. A brief description of the frame fields follows and is shown in Figure 5.7.

- *Preamble* is 7 bytes and consists of a pattern of alternating 0s and 1s. This field is used to provide bit synchronization.

- *Start of frame* consists of a 10101011 pattern and indicates the start of the frame to the receiver.
- *Destination address* specifies the destination MAC address.
- *Source address* specifies the source MAC address.
- *Length/Type* specifies the frame size, in bytes. The maximum Ethernet frame size is 1,518 bytes.
- *LLC data* is data from the LLC layer.
- *Pad* is used to increase the frame length to the value required for collision detection to work.
- *Frame check sequence* is 32-bit CRC for error checking (see Section 4.5.2).

The Ethernet versions have different data rates. Version 1000BaseSX, carrying 1 Gb/s, and 10GBase-T, carrying 10 Gb/s, hold the most promise for the future of high-speed LAN development.

5.6 Round-Robin-Access MAC

Unlike contention-access MAC, the *round-robin-access* scheme is deterministic, and there is no random allocation of a time slot to a frame. Although this scheme is not as popular as the contention-access LANs, some practical applications are associated with this scheme. Round-robin-access MAC is effective when most users have large amounts of data to transmit, such as in stream traffic.

Each user is given a chance to transmit data in a round-robin fashion. Each user may transmit data; if it has no data to transmit, the user passes its turn to the next user. Round-robin access can be implemented in a couple of ways. One method is known as the *token-ring access*.

5.6.1 Token-Ring Access Protocol

The *token-ring* configuration (see Figure 5.8) is based on the round-robin-access MAC. Token-ring passing is a method used for accessing the medium. The ring comprises repeaters interconnected with unidirectional links to form a closed loop. Each user is connected to a repeater. Repeaters perform data insertion, data reception, and data removal. A *token* is a small frame that is injected into the ring and circulated through the ring. Normally, one user is responsible for this task. Once a new token is circulated through the ring, the user needing to transmit captures the token and then starts transmitting data.

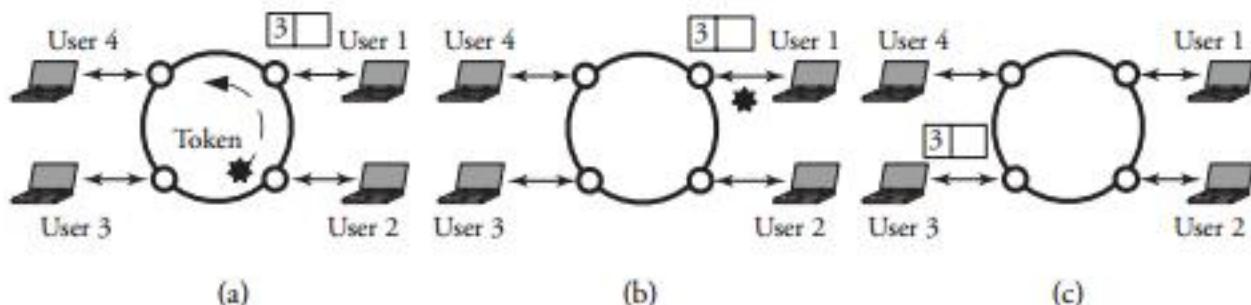


Figure 5.8 The movement of a frame in a ring-access MAC network

When a user transmits a frame, its repeater forwards the frame to the ring. The frame is circulated completely through the ring. During the circulation of the frame in the ring, each repeater copies the destination address of the frame and examines it. At each stop, if a repeater is the destination, it copies the entire frame. The data is removed by the sender after a complete circulation in the ring. The transmitting user also releases the token to the ring when the following two conditions are met: (1) the transmitting user has finished transmitting the frame and (2) the leading edge of the frame has finished one circulation through the ring and has reached the transmitting user. The advantages of this approach are

- Elimination of the need for acknowledgment
- Convenience of multicasting frames

A repeater plays an important role in the operation of the ring. The repeater goes to the *listen* state when passing the data through the ring and then goes to the *transmit* state when transmitting and receiving data. The repeater can also be in the *bypass* state when its associated user is down or offline. To provide fairness in the system, an upper limit is set on the amount of data each user can transmit at a time. The control mechanism for this scheme can be either centralized or decentralized.

The token-ring scheme is simple, promotes fairness, and provides priority and guaranteed bandwidth. The major flaw with this scheme is maintenance of the token. If the token is lost, the operation of the entire ring is halted. On the other hand, the duplication of token results in frame collisions. For successful ring operations, one user is selected as a monitor to replace a lost token and to ensure that only one token is circulated in the ring.

Example. Assume that user 1 in Figure 5.8 needs to transmit a frame to user 3. First, user 1 captures the token and transmits the frame, which circulates through the ring. Since the destination is user 3, user 4 examines the frame and then passes it to the next

user. When the frame reaches user 3, user 3 copies the frame. The frame continues the circulation through the ring until user 1 removes the frame and releases a new token.

Example. Now assume that user 1 in Figure 5.8 wants to multicast a frame to users 2 and 3. In this case, user 1 waits until it captures the token and then starts transmission. user 3 copies the frame followed by user 2. Once the frame reaches user 1, it releases the token to the ring and removes the frame from the ring.

5.6.2 Token-Ring: IEEE 802.5 Standard

The IEEE 802.5 standard defines the token-ring topology. IEEE 802.5 has also introduced the dedicated token ring, which makes use of a star topology. Users are directly connected to a central hub (concentrator) through full-duplex point-to-point links. This scheme eliminates the use of tokens.

5.7 Network of LANs

Increasing the number of interconnected devices and the volume of traffic requires splitting a single large LAN into multiple smaller LANs. Doing so dramatically improves network performance and security but also introduces a new challenge: how to interconnect multiple LANs. LANs may be of different types or may have devices that are not compatible with one another. New protocols need to be developed that allow all partitions of a LAN to communicate with one another.

Multiple LANs can be connected to form a college campus network. The campus backbone serves as a channel between a department and the rest of the campus network and facilitates the connection of the department to the Internet via a gateway router. The campus backbone is an interconnection of routers and switches. Servers used by an entire organization are usually located in a data bank and organization dispatching center.

Devices known as *protocol converters* are used to interconnect multiple LANs. Depending on the level of interconnection required, layer 1, layer 2, layer 3, and layer 4 protocol converters are available. Two networks can be connected using layer 1 devices referred to as *hubs* and *repeaters*. Layer 2 protocol converters have information about the layer 2 protocols on both interconnected LANs and can translate one to the other. At layer 2, *bridges* and *switches* can carry out the task as layer 2 devices. At layers 3 and 4, *routers* and *gateways*, respectively, are used.

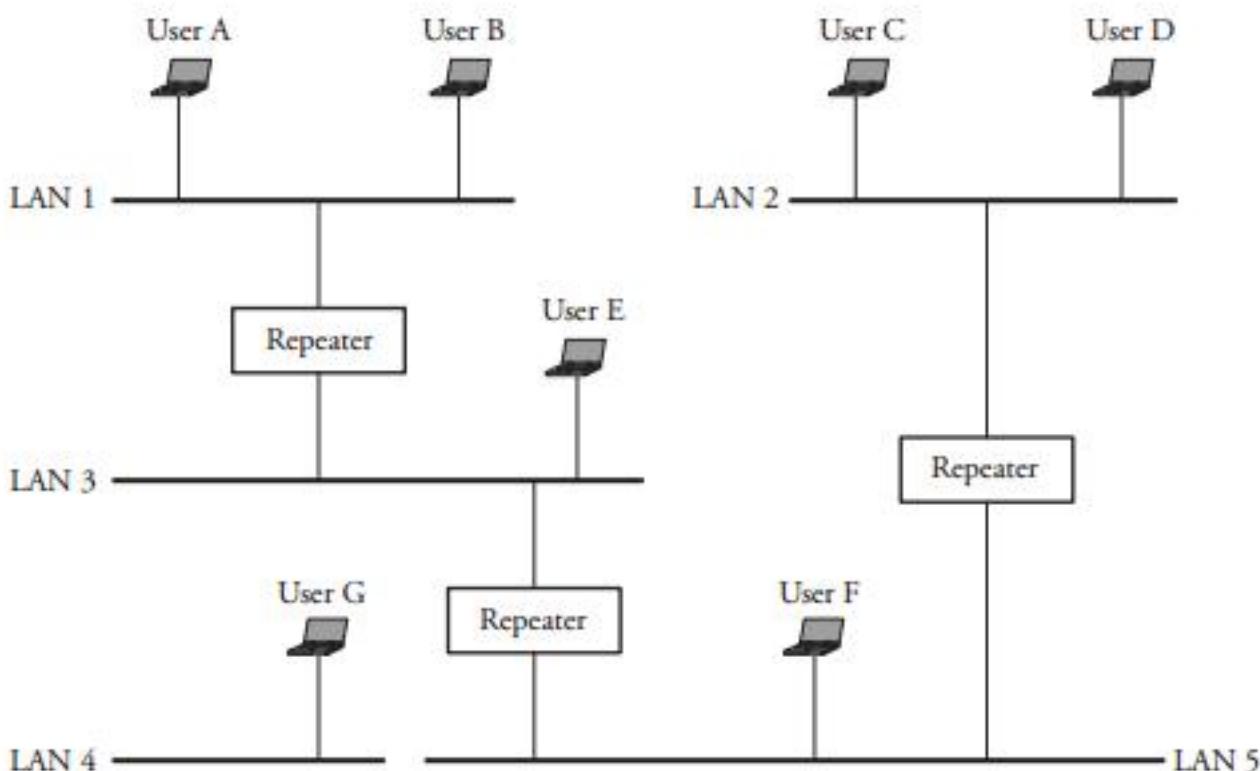


Figure 5.9 Seven layer 1 users connected through repeaters

5.7.1 Using Repeaters, Hubs, and Bridges

In Chapter 3, we explained the operations of repeaters, hubs, and bridges. Figure 5.9 shows a situation in which bus LANs are interconnected through repeaters. Users A–G are connected to multiple Ethernet LANs. Users of these networks are aware of the existence of repeaters and function as a single large LAN. Any user reads all flowing frames sent by other users but accepts those frames that are specifically addressed to it. Thus, collisions may be possible throughout the network if two or more users try to transmit at the same time.

Figure 5.10 depicts a network connection using hubs. A hub is similar to a repeater but copies frames and forwards them to all connected users. As in the case of a repeater, collisions may occur if two or more users try to transmit at the same time. Hubs and repeaters have the following limitations.

- Hubs forward frames to all users. This method of networking results in reduced LAN performance, owing to excess traffic.
- A large number of users are included in the collision domain.
- Security cannot be implemented, since a frame is forwarded to all users.

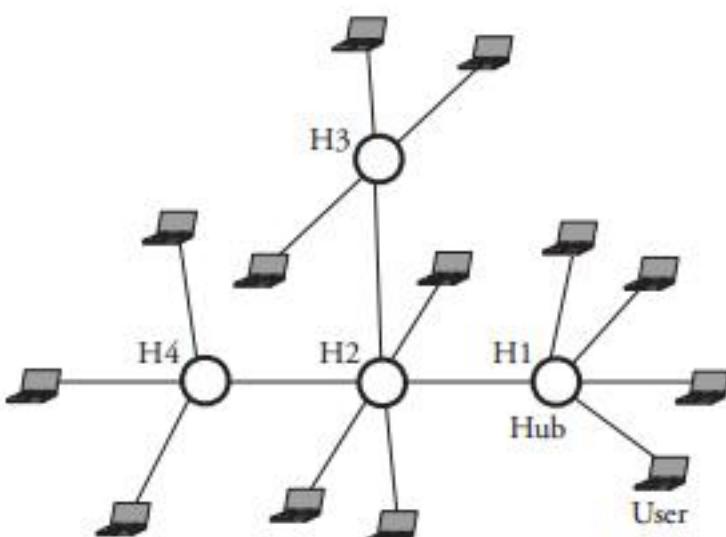


Figure 5.10 Using hubs in layer 1

Bridges

Clearly, using a bridge for networking Ethernet LANs can reduce the possibility of collision, as bridges split a LAN system into different collision domains. Because they can selectively retransmit the frame, bridges also offer a greater level of security than repeaters can. Bridges also facilitate communication across multiple LANs and bridges. Sometimes, a bridge that connects two LANs with nonidentical bit rates must have a buffer. A buffer in a bridge holds frames that arrive from a faster LAN directed to a slower LAN. This introduces transmission delay and has adverse effects on the network, causing the flow-control protocols to time out.

Figure 5.11 shows multiple bus LANs being connected by bridges. Suppose that user 1 wants to transmit a frame to user 5. First, bridge B1 examines the destination address and determines whether the forwarded frame is to be delivered to any of the users on LAN 2. If user 3 is not the destination within its connected LANs, the frame can be either dropped or forwarded on LAN2. Making such decisions is a bridge routing capability and depends on how well the routing table of the bridge is structured. Thus, the bridge decides whether to accept or reject a frame at any time.

If it decides to forward a frame on LAN 2, bridge B1 also performs error detection to ensure that the frame is not corrupted. Next, the bridge checks whether LAN 2 and LAN 5 have the same frame format. If they do, the bridge forwards the frame as is. If the frame format is different, it is reformatted to match the frame format of LAN 5. Since a bridge in such scenarios is generally connected to an Ethernet LAN, the bridge has to conform to CSMA/CD while transmitting the frame. When the frame reaches

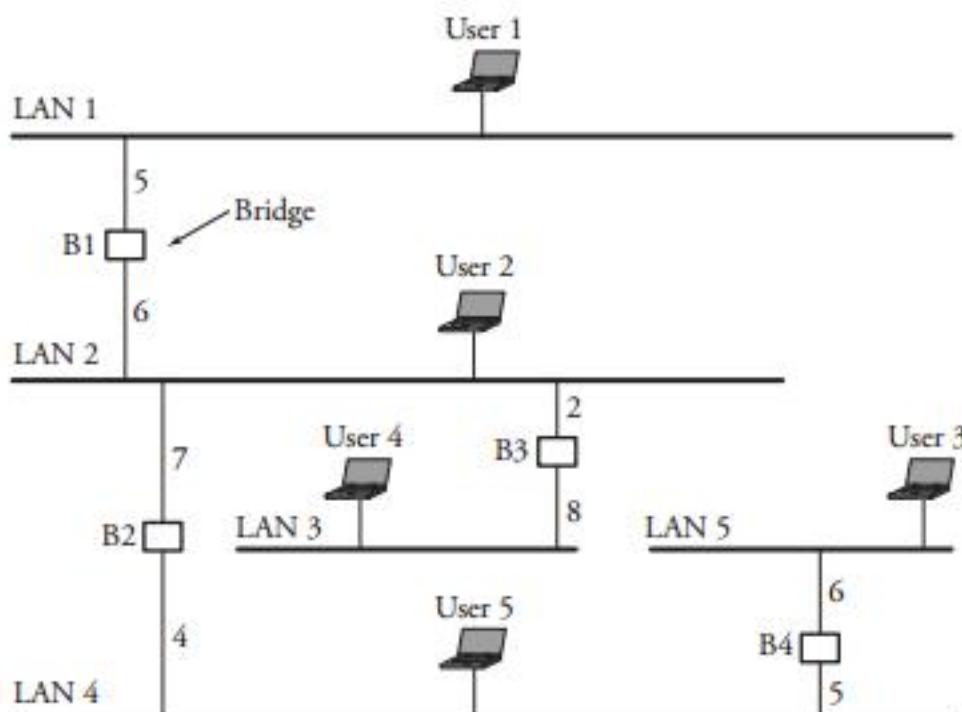


Figure 5.11 Connecting LANs through bridges

bridges B2 and B3, the same procedure as the one completed in B1 takes place. As a result, B3 rejects the frame, and B2 accepts the frame. The frame is now forwarded on LAN 2. The frame ultimately reaches the destination at user 3 after passing safely over LAN 2, B3, and LAN 3.

A bridge that connects two LANs, such as an Ethernet LAN and a token-ring LAN, has to reformat frames before any transmission. A frame arriving from a token-ring LAN is reformatted to match the Ethernet LAN. As the Ethernet frame format does not contain the priority field, the priority information specified by the token-ring frame is lost. In contrast, a frame arriving from an Ethernet network is assigned a default priority before transmission to the token-ring LAN. Figure 5.12 shows an example in which a LAN using bridges and hubs interconnects a token-ring LAN with several other LANs.

A bridge does not have a global view of its outside network but rather has knowledge only of immediate neighbors. Bridge routing is a process of deciding where to forward received frames. Bridges have this information stored in a table called the *routing table*. Each bridge has multiple subrouting tables for corresponding to all its existing surrounding connected LANs. The routing table consists of the destination address and the destination LAN.

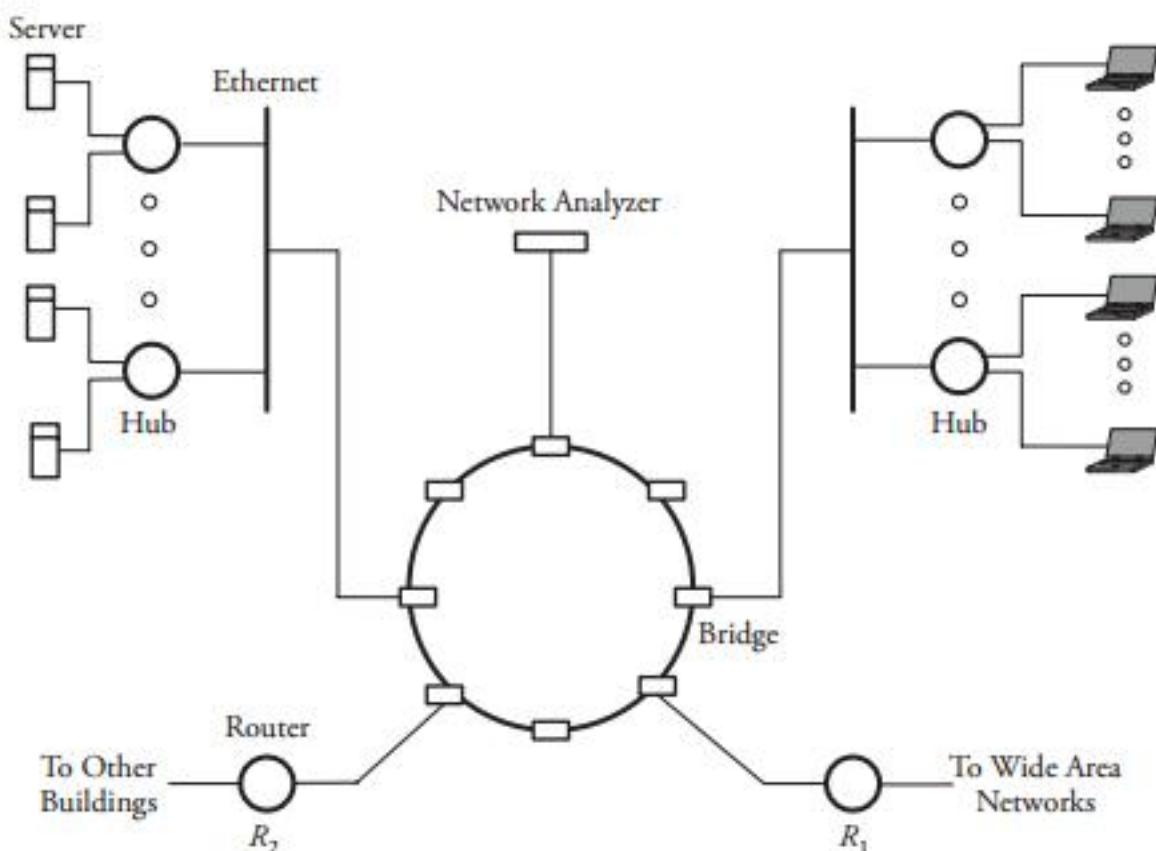


Figure 5.12 Using bridges and hubs

Example. Table 5.1 provides a bridge routing table for B2 and B3 in Figure 5.11. Table 5.2 gives more detail of a routing table for bridge B1. B1 has one routing table for LAN 1 and another routing table for LAN 2. For example, if a frame arrives on LAN 1, the bridge parses the frame for the destination address and looks up the routing table for source LAN 1. If the destination is user 1, the LAN does not need to forward the frame. However, for destinations at user 2, user 3, and user 5, the bridge forwards the frame to LAN 2. For the destination at user 5, the bridge forwards the frame to LAN 4. LAN 1 has five users, LAN 2 has four computers, and LAN 3 has LAN 2 users. Port 1 of the bridge connects to LAN 1, Port 2 of the bridge connects to LAN 2, and port 3 of the bridge connects to LAN 3.

In a static network, connections are fixed, so the routing table entries can be programmed into the bridge: *fixed routing*. In case of any changes to the network, the table entries need to be reprogrammed. This solution is not scalable for large, dynamic networks having frequent user addition and removal. Hence, such networks use an *automatic update* of the routing tables.

Table 5.1 Routing table for two bridges (B1 and B3) of Figure 5.11

Dest.	Next LAN from						
LAN 1	LAN 1	LAN 2	LAN 2	LAN 1	LAN 1	LAN 2	LAN 2
User 1	—	User 1	LAN 1	User 1	—	User 1	LAN 2
User 2	LAN 2	User 2	—	User 2	—	User 2	LAN 2
User 3	LAN 2	User 3	—	User 3	LAN 4	User 3	—
User 4	LAN 2	User 4	—	User 4	—	User 4	—
User 5	L2	User 5	—	User 5	LAN 4	User 5	—

Table 5.2 Bridge routing table for bridge B2 in Figure 5.11

Destination MAC Address	Next LAN
00-40-33-25-85-BB	LAN 1
00-40-33-25-85-BC	LAN 1
00-61-97-44-45-5B	LAN 2
00-C0-96-25-45-C7	LAN 2

In Figure 5.13, for example, if user 1 on LAN 1 sends a frame to user 5 on LAN 1, any corresponding bridge, such as B1 or B5 can figure out that both user 1 and user 5 belong to LAN 1 and forward the frame within LAN 1. Bridge B1 or B2 has the MAC addresses of both user 1 and user 5. Similarly, if user 10 is transmitting a frame to user 11, bridge B5 records the MAC addresses of users 10 and 11 and LAN 3 in its routing table. Each bridge must parse the destination address of an incoming frame in its routing table to determine the association between the destination MAC address and the MAC addresses of the devices connected to its LAN. The bridge scans the routing table to determine whether an association exists and forwards the frame if an entry is in the routing table.

Bridges that update their routing tables are called *transparent bridges* and are typically equipped with the IEEE 802.1d standard. These bridges act as plug-and-play devices and have the capability to build their own routing tables instantaneously. A transparent bridge also has the intelligence to learn about any change in the topology

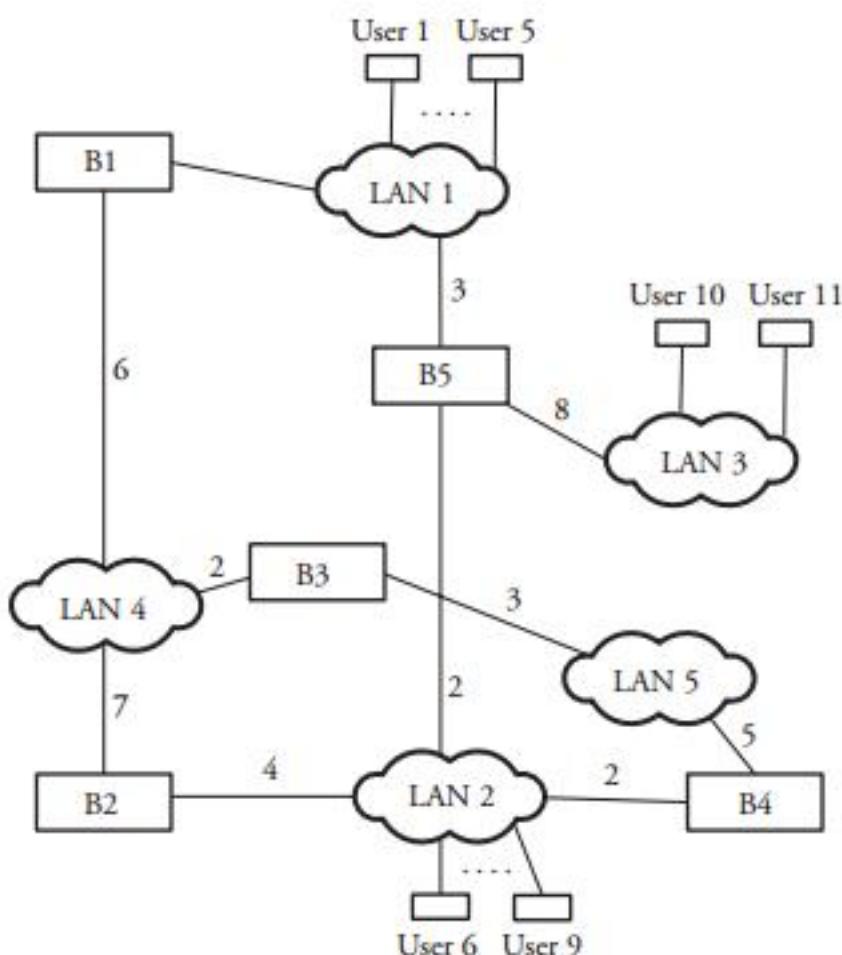


Figure 5.13 Bridging in local area networks

and to update its routing table. This type of bridge can dynamically build the routing table, based on the information from arriving frames. By parsing the source address of a received frame, a bridge can determine how it can access the local area network of the arriving frame. Based on this information, the bridge updates its routing table.

Example. Suppose that frames from user 1 move from LAN 1 to LAN 2 in Figure 5.11. If user 1 transmits a frame to user 2 located in LAN 2, bridge B4 examines its routing table and discovers that the direction on which it received the frame from user 1 is not the same as what it has in its routing table. Hence, bridge B4 updates its routing table.

A bridge initializes its routing tables by using the flooding algorithm. When none of the users in the network have routing table entries, a frame is flooded across the network to all users. If a frame arrives on a bridge that does not have the routing table

entry for the destination, the frame is flooded across all users. As more and more frames are flooded through the network, all bridges will eventually have routing table entries.

Example. Interconnection of multiple LANs via bridges can potentially result in the frames circulating indefinitely. One such situation is illustrated in Figure 5.13, where multiple routes are possible between user 1 and user 9. User 1 can reach user 9 via route LAN 1-B1-LAN 4-B2-LAN 2. User 1 can also reach user 9 via LAN 1-B5-LAN 2. With flooding algorithm, a frame released from user 1 being transmitted to user 9 via route LAN 1-B1-LAN 4-B2-LAN 2 can be transmitted back to user 1 via bridge B5, resulting in an infinite loop and hence network congestion.

Spanning-Tree Algorithm

The *spanning-tree algorithm* is used to overcome the problem of infinite loops in networks with bridges. A spanning tree generates a subset of bridges to be used in the network to avoid loops. The algorithm is as follows

Begin Spanning-Tree Algorithm

1. Each link from a bridge to a LAN is assigned a cost. This link cost is inversely proportional to the link's bit rate. A higher bit rate implies a lower cost.
2. Any bridge with the lowest ID is selected as root. A spanning tree is constructed originating from the root bridge. To build the spanning tree, all bridges send a special frame, called a *bridge protocol data unit* (BPDU), comprising a bridge ID and the aggregate cost, from a sender to a receiving user.
3. A receiving bridge compares the sender's bridge ID with its own ID. If the sender's bridge ID is lower, the BPDU is not forwarded. If the sender's bridge ID is higher, the BPDU is stored and forwarded to other users after incrementing the cost. Thus, the bridge determines that it is not the root bridge and stops sending BPDU advertising for the lowest bridge ID. Over a period of time, all bridges, excluding the bridge with the lowest ID, stop sending BPDUs. When the bridge receives no other BPDUs, it declares itself the root bridge.
4. Based on the comparison of all the stored BPDUs, each of the involving bridges determines the least-cost path to the root bridge with an identified port number. This port is called the root port, and any bridge communicates with the root bridge through the root port.
5. Every LAN determines a designated bridge through which it forwards frames. To determine the designated bridge, each bridge sends BPDUs to all LANs to which it is connected. Bridges connected to a particular LAN compare the respective costs to reach the root bridge. The bridge with the lowest cost is designated the root bridge. In case of a tie, the lowest bridge ID determines the designated bridge. ■

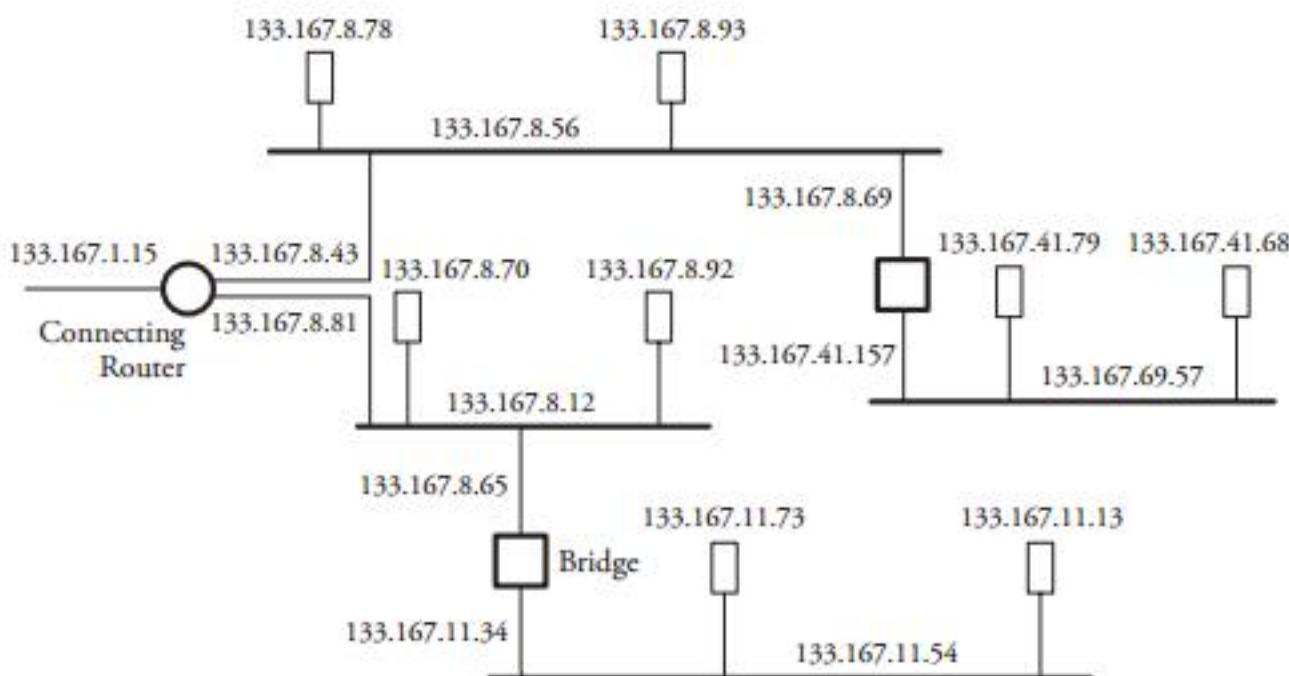


Figure 5.14 A LAN system and the assignment of IP addresses

Example. Figure 5.14 shows that every component of a LAN system is assigned an IP address. Every piece of a LAN cable, including the ones that connect a bridge to a bus, takes an IP address.

5.7.2 Layers 2 and 3 Switches

As the complexity of the network system grows, layer 2 devices are not adequate to meet the needs of networks. Users on LANs connected by layer 2 switches have a common MAC broadcast address. Hence, a frame with a broadcast MAC address is forwarded to all users on the network. In a large network, this is considered a large overhead and may result in network congestion. Another issue with layer 2 switches is that to avoid closed loops, there can be only one path between two users. This poses a significant limitation on the performance of large networks. The limitations are overcome by splitting the network into subnets.

Routers, known as layer 3 switches, implement the switching and forwarding functions at the network layer of the protocol stack. The routers are capable of handling heavy traffic loads. Routers are also used to connect multiple subnets in LANs. Routers are sophisticated devices that permit access to multiple paths among users. A router uses software to forward packets or frames. However, the use of software significantly reduces the speed of forwarding frames. But high-speed LANs and high-performance

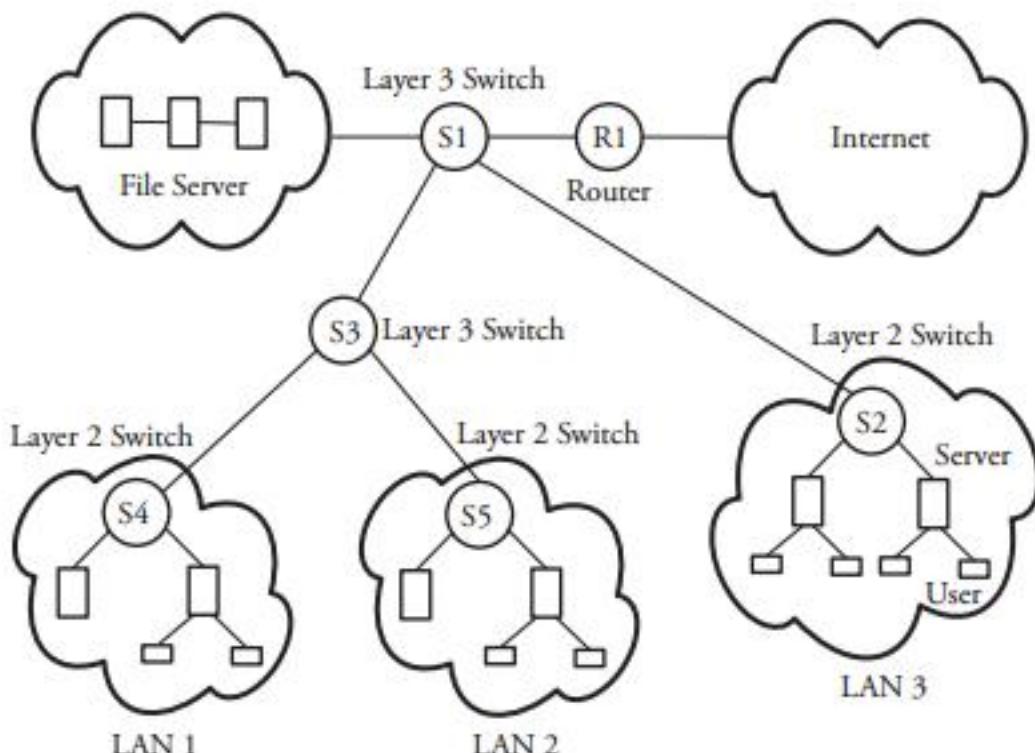


Figure 5.15 A network with layers 2 and 3 switches

layer 2 switches can operate on millions of frames per second, which mandates layer 3 devices to match the load.

Figure 5.15 shows a typical network scenario in a large organization. The network is split up into subnets, each having a number of desktop systems connected to a layer 2 switch. A layer 3 switch acts as the backbone and connects layer 2 switches through higher-speed links. Servers are connected to either the layer 2 or the layer 3 switch. A software-based router provides the WAN connection.

5.8 Summary

A local area network is a communication network that interconnects a variety of data communications devices to serve within a small community. The primary feature of a LAN is its ability to share data and software to provide common services, such as file serving, print serving, support for electronic mail, and process control and monitoring in office and industrial environments. We explored the three basic topologies of LANs—bus, ring, and star—using fundamental knowledge on devices (Chapter 3) and on links (Chapter 4). The simplicity of bus LANs is an advantage when a cable can connect users through its taps. The star topology is a variation on the bus topology,

whereby a hub provides connection capability. Star topology has the advantage of easier installation than bus topology.

For a user to place data onto a local area network, the network must be equipped with a MAC protocol to find ways of sharing its medium. The *Address Resolution Protocol* (ARP) is designed to convert a IP address to a MAC address or vice versa. Two well-known MAC methods are *contention access*, such as CSMA/CD for bus and star topologies, and *round-robin access*, such as token passing or token ring. CSMA/CD operates on a first-come first-served basis and clearly is one of the most popular access protocols. However, its frame collision during high-usage periods is indeed a bottleneck. Round-robin access is more appropriate than CSMA, as the round-robin method can regulate traffic under heavy loads.

We also covered important topics of *internetworking* and the use of repeaters, hubs, and bridges with *switched* LANs. A bridge operates at layer 2 and typically separates users into two collision domains. Routers, known as layer 3 switches, are also used to connect multiple subnets in larger LANs. Routers are typically designed to handle heavy traffic loads.

In Chapter 6, we explore wireless networks. We consider them from two angles: cellular structures and local area networks.

5.9 Exercises

1. Consider the transfer of a file containing 1 million characters from one computer to another. Transfer consists of a sequence of cycles. For one cycle, $a = (\text{time for data packet} + \text{propagation}) + (\text{time for ack packet} + \text{propagation})$. The throughput refers to the number of sequences required to transfer 1 million characters. Each character in its digital form requires 8 bits. The two computers are $D = 1$ km apart, and each generates a data rate of $b = 1$ Mb/s, with a packet size $s = 256$ bits, which includes 80 bits of overhead. The propagation speed on the bus is 200 m/ μ sec. Find the total elapsed time using throughput and C for the following two cases.
 - (a) A bus topology, with each frame acknowledged with an 88-bit frame before the next frame is sent.
 - (b) A ring topology having a total circular length of $2D$, with the two computers D distance apart. Acknowledgment is achieved by allowing a frame to circulate past the destination user back to the source user. The ring has $N = 100$ repeaters, each of which introduces a delay of 1 bit time.

2. We want to design a coaxial LAN for 12 offices arranged on three similar floors, each floor having two rows with 2 offices and the rows separated by a hallway. Each office is $5 \text{ m} \times 5 \text{ m}$ with a height of 3 m. The LAN center is in the center of the ground floor beneath the three office floors. Assume that each office requires two IP telephone lines and retrieves two Web pages per minute at the average rate of 22 K per page.
 - (a) Estimate the distance from each office to the LAN center.
 - (b) Estimate the required available bit rate for the LAN.
3. Consider a 100 m bus LAN with a number of equally spaced computers with a data rate of 100 Mb/s.
 - (a) Assume a propagation speed of $200 \text{ m}/\mu\text{s}$. What is the mean time to send a frame of 1,000 bits to another computer, measured from the beginning of transmission to the end of reception?
 - (b) Assume a mean distance between pairs of computers to be 0.375 km, an approximation based on the following observation: For a computer on one end, the average distance is 0.5 km. For a computer in the center, the average distance is 0.25 km. With this assumption, the time to send is transmission time plus propagation time.
 - (c) If two computers with a mean distance of 0.37 km start transmitting at the same time, their frames interfere with each other. If each transmitting computer monitors the bus during transmission, how long does it take before it notices a collision? Show your answer in terms of both time and bit time.
4. Consider a 100 Mb/s 100BaseT Ethernet LAN with four attached users, as shown in Figure 5.5. In a nonpersistent CSMA/CD algorithm, a user normally waits $512g$ bit times after a collision before sending its frame, where g is drawn randomly. Assume that a 96 bit times of waiting period are needed for clearing the link from the jammed signal in the 100BaseT Ethernet LAN. Assume that only user 1 and user 4 are active and that the propagation delay between them is 180 bit times. Suppose that these two users try to send frames to each other and that their frames collide at the half-way LAN link. User 1 then chooses $g = 2$, whereas user 4 picks $g = 1$, and both retransmit.
 - (a) How long does it take for user 1 to start its retransmission?
 - (b) How long does it take for user 4 to start its retransmission?
 - (c) How long does it take the frame from user 4 to reach user 1?

5. For a CSMA/CD system, consider the throughput derived from Equation (5.15). Set two parameters in this equation: $\alpha = t_p/T$ and $\beta = \lambda T$.
 - (a) Rewrite the throughput in terms of α and β , and denote it by U_n .
 - (b) Comment why R_n is in terms of frames/time slot and why β can be called "offered load" in this case.
 - (c) Using a computer, sketch four plots all in one chart for U_n in terms of β , given $\alpha = \{0.001, 0.01, 0.1, 1.0\}$, and comment on the behavior of the normalized throughput, U_n .
6. For a local area network using CSMA/CD, assume a 12 percent frame error rate owing to noise and errors resulting from collisions. Discuss how the throughput U could be affected.
7. A 10 Gb/s Ethernet LAN with ten users attached to it uses CSMA/CD. The bus is about 10 meters, and users' frames are restricted to a maximum size 1,500 bytes. Based on the statistics, four users in average are active at the same time.
 - (a) Find the frame propagation and transmission times.
 - (b) Find the average utilization of the bus.
 - (c) Find the probability that a user attempts to transmit frames in an empty time slot.
 - (d) Find the probability that a user attempts seven different times in seven different empty slots to transmit its frame and is not successful, owing to collision, but is successful on the eighth attempt.
 - (e) Find the average number of contentions.
8. Using the CSMA details discussed in this chapter, sketch a block diagram that represents the implementation of this algorithm for the following cases:
 - (a) Nonpersistent CSMA
 - (b) p -persistent CSMA
9. Design (show only the network) a LAN system for a small five-story building. One floor is dedicated to two mail servers and separated three database servers. Each of remaining floor has four computers with broadband access. Your design should meet the following restrictions and conditions: three-input hubs, one bridge, and unlimited Ethernet buses. The incoming broadband Internet access must be connected to a six-repeater ring, no bus LAN is allowed outside of a floor, and a traffic analyzer must be attached to the network.

CHAPTER 6

Wireless Networks and Mobile IP

In Chapter 4, we reviewed the fundamentals of wireless communication media at the link layer. In this chapter, we explore the concept of wireless networks. Wireless networks are designed for all home and business networking applications and are used in both LANs and WANs. The major topics of the chapter are

- *Infrastructure of wireless networks*
- *Wireless LANs*
- *Protocol layers of wireless LANs*
- *Cellular networks*
- *Mobile IP*
- *Wireless mesh networks (WMNs)*

We start with an overview of wireless communication systems at all levels, from satellite to LANs. We review the protocols and topologies of LANs in the wireless environment, as well as the mobility issue in wireless networks. We then focus on cellular networks, one of the main backbones of our wireless networking infrastructure. At the end of the chapter, we introduce *mobile IP* and *wireless mesh networks*, including WiFi and WiMAX technologies.

6.1 Infrastructure of Wireless Networks

Figure 6.1 shows wireless communication systems at all levels. Satellite systems can provide a widespread global coverage for voice, data, and video applications. A satellite orbits the earth while interfacing between a receiver/transmitter pair. The interfacing facilitates the transmission of data to long-distance destinations. Satellite systems are classified according to the orbit distance from the earth:

- *Low-earth orbit.* These satellites orbit the earth at distances between 500 km and 2,000 km and can provide global roaming.
- *Medium-earth orbit.* These satellites orbit the earth at distances of about 10,000 km.

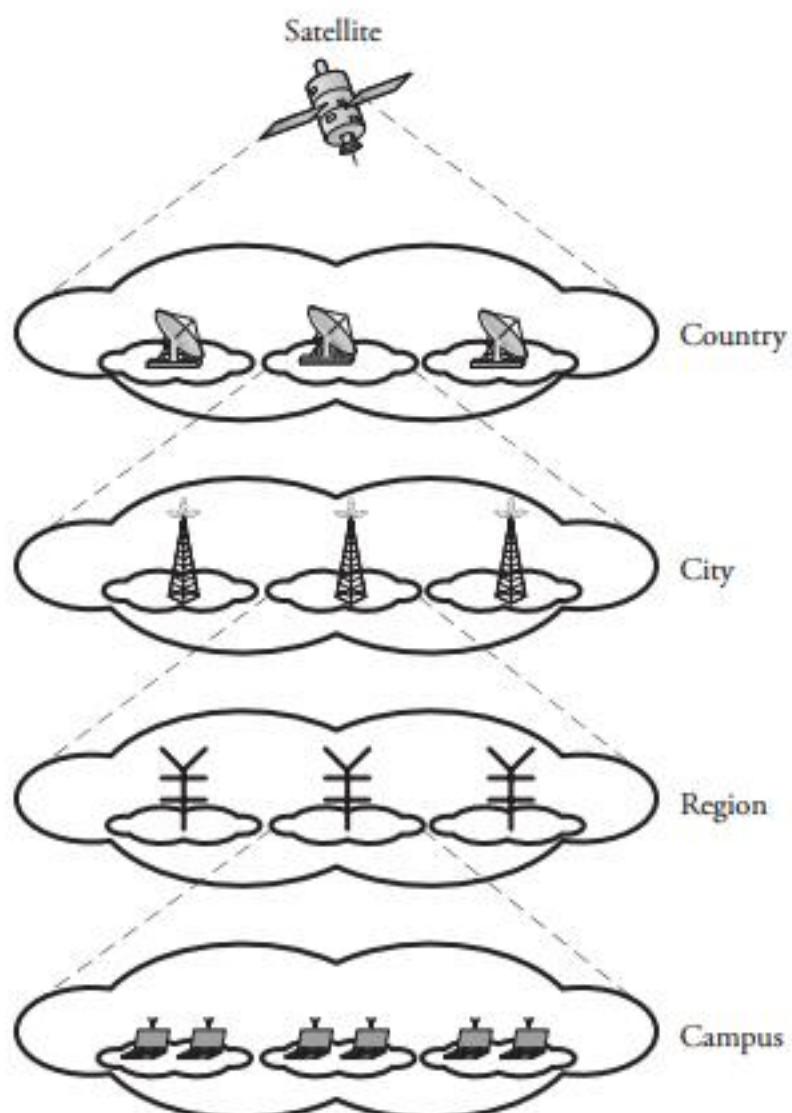


Figure 6.1 Wireless communication systems, from satellite to LAN

- *Geosynchronous orbit.* These satellites orbit the earth at distances of about 35,800 km. They have a large coverage area and can handle large amounts of data.

Similar to wired networks, wireless networks are hierarchical. However, wireless networks are characterized by limited resources from the perspective of frequency range and available bandwidth. The available bandwidth often varies with time. Wireless networks must be able to adapt to the changing network topology. Wireless network topologies are classified into three main groups:

- Hierarchical
- Star
- Peer to peer (P2P)

The hierarchical architecture acts as a spanning tree and normally covers a large geographical area. The lowest layer of the hierarchy normally represents indoor systems that cover very small areas. The next layer in the hierarchy consists of cellular systems. This hierarchy can be extended to global coverage. The hierarchical topology is ideally suited for large networks. In a star topology, as explained in Chapter 5, nodes are connected to a central hub. Thus, the entire traffic from nodes flows through this central hub. Star networks are used in cellular and paging systems.

Peer-to-peer networks are characterized by pairs of nodes and are normally used in military applications. In these networks, nodes are self-configuring, and various tasks are evenly distributed among nodes. Peer-to-peer networks are multihop networks; a node may use multiple hops to communicate with another node. These networks normally have multiple paths between each two nodes to route around link failures.

6.2 Wireless LAN Technologies

Wireless technology helps wired data networks join wireless components. Local area networks can be constructed in wireless fashion mainly so that wireless users moving within a certain organization, such as a university campus, can access a backbone network.

The basic topology in wireless LANs is shown in Figure 6.2 (a). Each user in the wireless network communicates directly with all others, without a backbone network. An improvement to this scheme involves the use of *access points*, or transceivers that can also serve as an interface between the wired and the wireless LANs. Figure 6.2 (b) shows a typical setup with an access point called *wireless switch/hub*. In this scheme, all wireless users transmit to an access point to communicate with users on the wired or

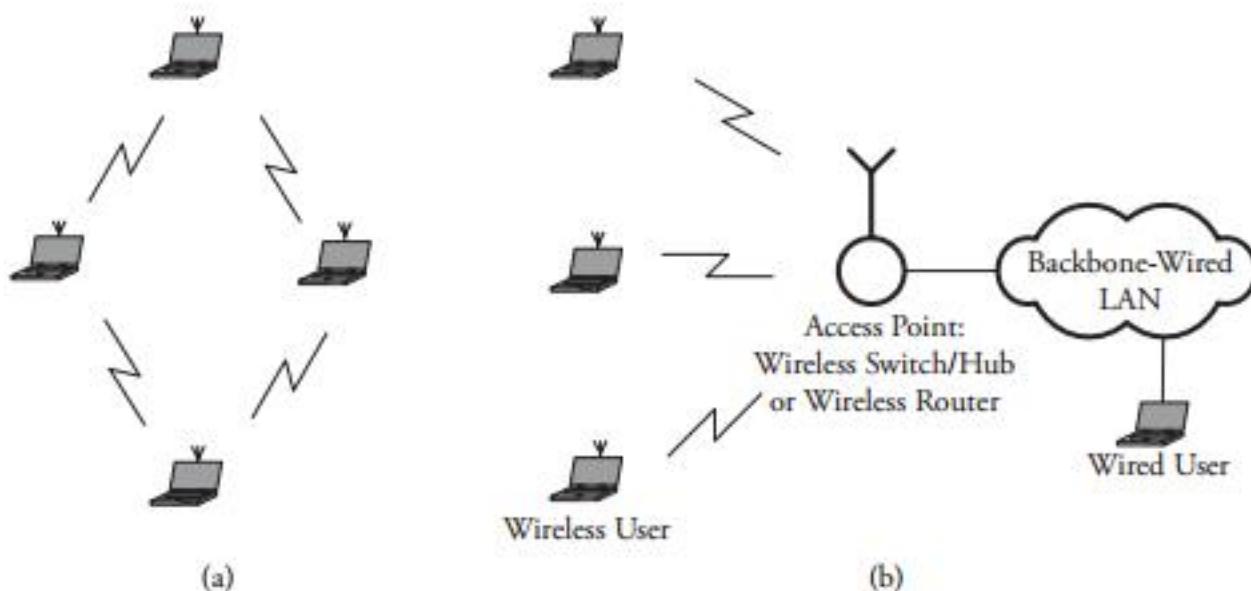


Figure 6.2 Basic wireless LANs: (a) basic topology; (b) typical setup with access point

wireless LAN. The range of user mobility in a wireless LAN can be extended by using several access points. A mobile user searches for a new access point when the signal level from a nearby access point increases beyond that of its current one. Wireless LAN technologies can be classified into four types: *infrared*, *spread-spectrum*, *narrowband RF*, and *home RF* and *Bluetooth*.

6.2.1 Infrared LANs

Each signal-covering cell in an *infrared LAN* is limited to one room. The coverage is small, since the infrared rays cannot penetrate through walls and other opaque obstacles. Infrared communication technology is used in several home devices, such as television remote controls. Three alternative transmission techniques are used for infrared data transmission: *directed beam*, *omnidirectional configuration*, and *diffused configuration*.

The *directed beam* involves point-to-point connections. The range of communications is limited by the transmitted power and the direction of focus. With proper focusing, ranges up to a kilometer can be achieved. This technology can be used in token-ring LANs and interconnections between buildings. The *omnidirectional configuration* consists of a single base station that is normally used on ceilings. The base station sends an omnidirectional signal, which can be picked up by all transceivers. The transceivers in turn use a directional beam focused directly at the base-station unit. In the *diffused-configuration* method, the infrared transmitters direct the transmitted signal

to a diffused reflecting ceiling. The signal is reflected in all directions from this ceiling. The receivers can then pick up the transmitted signal.

The use of infrared has several advantages. For example, the bandwidth for infrared communication is large and can therefore achieve high data rates. Also, because infrared rays are reflected by lightly colored objects, it is possible to cover the entire area of the room with reflections from objects. Since infrared cannot penetrate through walls and other opaque obstacles, it becomes very difficult for any adversary to carry out a passive attack or to eavesdrop. Hence, communication with infrared technology is more secure. Also, separate infrared networks can be used in adjacent rooms without any interference effects. Finally, equipment for infrared communication is much cheaper than microwave communication. The one major disadvantage of infrared technology is that background radiation from sunlight and indoor lighting can cause interference at the infrared receivers.

6.2.2 Spread-Spectrum LANs

Spread-spectrum LANs operate in industrial, scientific, and medical applications, making use of multiple adjacent cells, each having a different center frequency within a single band to avoid any interference. Within each of these cells, a star or peer-to-peer topology can be deployed. If a star topology is used, a hub as the network center is mounted on the ceiling. This hub, serving as an interface between the wired and wireless LANs, can be connected to other wired LANs. All users in the wireless LAN transmit and receive signals from the hub. Thus, the traffic flowing among users moves through the central hub. Each cell can also deploy a peer-to-peer topology. The spread-spectrum techniques use three different frequency bands: 902–928 MHz, 2.4 GHz–2.4835 GHz, and 5.725 GHz–5.825 GHz. Higher-frequency ranges offer greater bandwidth capability. However, the higher-frequency equipment is more expensive.

6.2.3 Narrowband RF LANs

Narrowband radio frequency (RF) LANs use a very narrow bandwidth. Narrowband RF LANs can be either licensed or unlicensed. In licensed narrowband RF, a licensed authority assigns the radio frequency band. Most geographic areas are limited to a few licenses. Adjacent cells use different frequency bands. The transmissions are encrypted to prevent attacks. The licensed narrowband LANs guarantee communication without any interference. The unlicensed narrowband RF LANs use the unlicensed spectrum and peer-to-peer LAN topology.

6.2.4 Home RF and Bluetooth

The *home RF* is a wireless networking standard that operates in the 2 GHz frequency band. Home RF is used to interconnect the various home electronic devices, such as desktops, laptops, and appliances. Home RF supports data rates of about 2 Mb/s and both voice and data and has a range of about 50 meters. *Bluetooth* is a technology to replace the cables necessary for short-range communication within 10 meters, such as between monitors and CPU, printer and personal computers, and so on. Bluetooth technology also eliminates the need for cables in laptops and printers. Bluetooth operates at 2.4 GHz frequency band and supports data rates of 700 Kb/s.

6.3 IEEE 802.11 Wireless Standard

Each wireless LAN user in Figure 6.2 (b) has a wireless LAN adapter for communication over the wireless medium. This adapter is responsible for authentication, confidentiality, and data delivery. To send data to a user in the wired LAN, a user in the wireless LAN first sends the data packet to the access point. The access point recognizes the wireless user through a unique ID called the *service-set identification* (SSID). SSID is like a password-protection system that enables any wireless client to join the wireless LAN. Once the wireless user is authenticated, the access point forwards data packets to the desired wired user through the switch or hub.

Access points build a table of association that contains the MAC addresses of all users in the wireless network. The access point uses this information for forwarding data packets in the wireless network. Figure 6.3 shows a setup whereby the LANs in two buildings are interconnected by *wireless bridges*. A wireless bridge is basically the same as a regular bridge but is equipped with a wireless transceiver. The most common medium for wireless networks is radio waves at a frequency of 2.4 GHz band. Wireless bridges are also used to interconnect LANs in different buildings. The access range of wireless LANs can be extended by deploying a greater number of access points.

Figure 6.4 shows multiple access points being used to extend the connectivity range of the wireless network. The area of coverage of each access point can be overlapped to adjacent ones to provide seamless user mobility without interruption. Radio signal levels in a wireless LAN must be maintained at an optimum value. Normally, a site survey must be conducted for these requirements. Site surveys can include both indoor and outdoor sites. The surveys are normally needed for power requirements, placement of access points, RF coverage range, and available bandwidth.

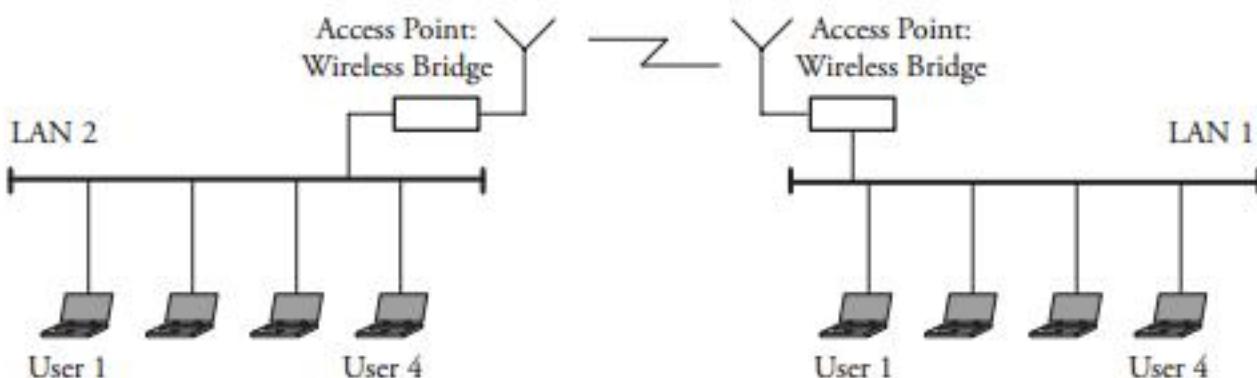


Figure 6.3 Connecting two LANs through wireless bridges

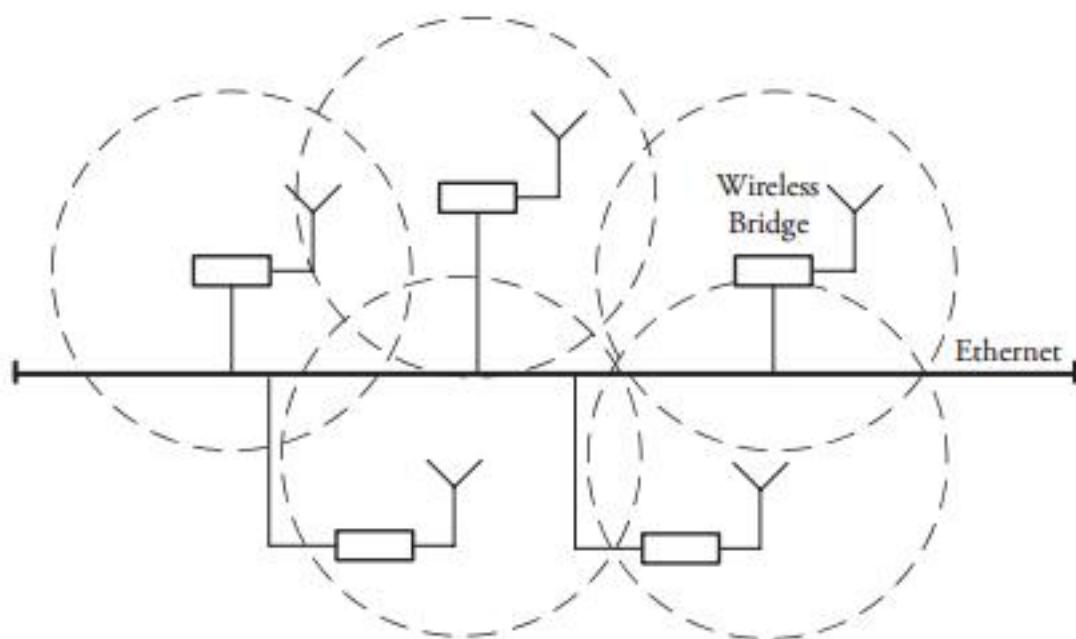


Figure 6.4 Use of multiple access points to extend the range of wireless access

The transmission media used in high-speed LANs are twisted pair and fiber-optic cables. The use of wireless media presents a few advantages, such as user mobility and reduced cost of transmission media. User mobility enables users to access the network resources from any point in the geographic area. Wireless LANs must be reliable and secure in order to be widely deployed. The standards for wireless LANs include 802.11 and its family: 802.11a, 802.11b, and 802.11g. Standard 802.11 typically uses the *Carrier Sense Multiple Access with collision avoidance* (CSMA/CA) method (see Chapter 5). With this method, each user listens for traffic coming from other users and transmits data if the channel is idle. If the channel is busy, the user waits until the channel becomes idle. The user then transmits data after a random *back-off time*. This is done to prevent

all users from transmitting at the same time when the channel becomes idle. The details of the 802.11 standards are explained further in the next two subsections.

The IEEE 802.11 wireless LAN standard defines services for physical, MAC layer, and MAC management protocols. The physical layer is responsible for transmitting raw data over RF or infrared media. The MAC layer resolves access control issues and ensures privacy of transmitted data and reliability of data services. The management protocols ensure authentication and data delivery.

6.3.1 802.11 Physical Layer

IEEE 802.11 operates in the 2.4 GHz band and supports data rates of 1 Mb/s to 2 Mb/s. IEEE 802.11a operates in the 5 GHz band and supports data rates of up to 54 Mb/s. IEEE 802.11b operates in the 2.4 GHz band and supports data rates of 5.5 Mb/s to 11 Mb/s. IEEE 802.11g operates at 2.4 GHz and supports even higher data rates.

The IEEE 802.11 physical layer is of four types.

1. *Direct-sequence spread spectrum* (DSSS) uses seven channels, each supporting data rates of 1 Mb/s to 2 Mb/s. The operating frequency range is 2.4 GHz ISM band. DSSS uses three nonoverlapping channels in the 2.4 GHz ISM band. The 2.4 GHz frequency band used by 802.11 results in interference by certain home appliances, such as microwave ovens and cordless telephones, which operate in the same band.
2. *Frequency-hopping spread spectrum* (FHSS) uses a pseudonoise sequence and signal hopping from one channel to another. This technique makes use of 79 channels. FHSS operates in the 2.4 GHz ISM band and supports data rates of 1 Mb/s to 2 Mb/s.
3. *Infrared* with an operating range of about 20 meters operates on a broadcast communication paradigm. A *pulse position modulation* (PPM) scheme is used.
4. *Orthogonal frequency division multiplexing* (OFDM), explained in Chapter 4, is a multicarrier modulation scheme whereby the carrier spacing is carefully selected so that each subcarrier is orthogonal to the other subcarriers. Two signals are *orthogonal* if they are multiplied together and their integral over an interval is 0. Orthogonality can be achieved by letting the carrier spacing be equal to the reciprocal of the useful symbol period. As the subcarriers are orthogonal, the spectrum of each carrier has a null at the center frequency of each of the other carriers in the system. This results in no interference between the carriers, allowing them to be spaced as close as possible.

IEEE 802.11a uses OFDM, which uses 12 orthogonal channels in the 5 GHz range. This reduces the interference from other home appliances, unlike the case with 802.11b. The two standards 802.11a and 802.11b can operate next to each other without any interference: 802.11a equipment is more expensive and consumes more power, as it uses OFDM. The frequency channels are nonoverlapping. *IEEE 802.11a* operates in the 5 GHz band. The achievable Mb/s data rates are 6, 9, 12, 18, 24, 36, 48, and 54. Convolution coding is used for forward error correction.

IEEE 802.11b uses DSSS but supports data rates of up to 11 Mb/s. The modulation scheme employed is called *complementary code keying* (CCK). The operating frequency range is 2.4 GHz and hence can interfere with some home appliances. *IEEE 802.11g* achieves very high data rates compared to 802.11b and uses the 2.4 GHz frequency band. A combination of encoding schemes is being used in 802.11g. An 802.11g client can operate with an 802.11b access point; similarly, an 802.11b client can operate with an 802.11g access point.

6.3.2 802.11 MAC Layer

IEEE 802.11 provides several key functionalities: reliable data delivery, media access control, and security features. *Reliable data delivery* is a key feature available in the MAC layer of IEEE 802.11. The imperfections of the wireless medium, such as noise, interference, and multipath effects, may lead to frame loss. IEEE 802.11 uses acknowledgment (ACK) to ensure reliable data delivery. When a source sends a data frame, the destination responds with an ACK to acknowledge receipt of the frame. If the source does not receive an ACK for a certain period of time, it times out the process and retransmits the frame.

The *request-to-send/clear-to-send* (RTS/CTS) scheme also is used to further enhance reliability. When it has data to send, a source sends an RTS signal in the form of a frame to the destination. The destination sends a CTS signal if it is ready to receive data. The source sends the data frame after receiving the CTS signal from the destination. The destination then responds with an ACK to indicate successful receipt of data. This four-way handshake leads to a greater reliability of data delivery. When a source sends an RTS frame, users within the reception range of the source avoid sending any frames during this interval, to reduce the risk of collisions. For the same reason, when the destination sends a CTS frame, users in the reception range of the destination refrain from sending any frames during this interval.

Another key functionality is *media access control* (MAC). Media-access algorithms are of two types: *distributed access* and *centralized access*. In distributed-access protocols,

media access control is distributed among all the nodes. Nodes use a carrier-sense mechanism to sense the channel and then transmit. Distributed-access protocols are used in ad hoc networks with highly bursty traffic. In centralized-access protocols, the media-access issues are resolved by a central authority. Central-access protocols are used in some wireless LANs that have a base-station backbone structure and in applications that involve sensitive data. The IEEE 802.11 MAC algorithm provides both distributed- and centralized-access features. Centralized access is built on top of distributed access and is optional.

The MAC layer consists of two sublayers: the *distributed-coordination function* (DCF) algorithm and the *point-coordination function* algorithm (PCF).

Distributed Coordination Function (DCF) Algorithm

The DCF algorithm uses contention resolution, and its sublayer implements the CSMA scheme for media access control and contention resolution. As explained in Chapter 5, a CSMA sender listens for traffic on the medium. If it senses that the medium is idle, it transmits; otherwise, if the medium is busy, the sender defers transmission until the medium becomes idle. DCF has no provisions for collision detection, which is difficult in wireless networks because of the hidden-node problem. To overcome this problem, the interframe space (IFS) technique is used. IFS is a delay whose length is based on frame priority. IFS has three timing levels. The steps in the algorithm follow.

Begin DCF Algorithm for Wireless 802.11 MAC

1. The sender senses the medium for any ongoing traffic.
2. If the medium is idle, the sender waits for a time interval equal to IFS. Then the sender senses the medium again. If the medium is still idle, the sender transmits the frame immediately.
If the medium is busy, the sender continues to sense the medium until the medium becomes idle.
3. Once the medium becomes idle, the sender delays its activity by a time interval equal to IFS and senses the medium again.
4. If the medium is still idle, the sender backs off for an exponential time interval and senses the medium again after that interval.
If the medium continues to be idle, the sender transmits immediately.
If the medium becomes busy, the sender stops the back-off timing and restarts the process once the medium becomes idle. ■

The IFS time-interval technique is based on the priority of the data. The three timing intervals used for IFS are:

1. *Short IFS (SIFS)*. This timing interval is used if immediate response is required. A sender using a SIFS has highest priority. SIFS is used to send ACK frames. A user receiving a frame directed to only itself responds with an ACK after a time interval equal to SIFS. The SIFS time interval compensates for the lack of a collision-detection system in wireless networks.
2. *Point IFS coordination function (PIFS)*. This timing interval is used by the central authority or controller in the PCF scheme.
3. *Distributed IFS coordination function (DIFS)*. This timing interval is used for the normal asynchronous frames. Senders waiting for the DIFS interval have the least priority.

Point Coordination Function

The *point-coordination function* (PCF) provides a contention-free service. PCF is an optional feature in IEEE 802.11 and is built on top of the DCF layer to provide centralized media access. PCF includes a *polling feature* implemented at the centralized polling master (point coordinator). The point coordinator uses the PIFS interval to issue polls. Since this interval is greater than the DIFS interval, the coordinator effectively restrains all asynchronous traffic when issuing a poll. The protocol defines an interval called the *superframe interval*, which consists of a contention-free period that the point coordinator used to issue polls. The other interval is the contention period used for stations to send normal asynchronous data. The next superframe interval begins only after the medium becomes idle. The point coordinator has to wait during this period to gain access.

MAC Frame

The frame format for the 802.11 MAC is shown in Figure 6.5 and is described as follows.

- *The frame control (FC) field* provides information on the type of frame: control frame, data frame, or management frame.
- *Duration/connection ID (D/I)* refers to the time allotted for the successful transmission of the frame.
- *The addresses* field denotes the 6-byte source and destination address fields.

Byte:

	2	2	6	6	6	2	6	4
FC	D/I	Address	Address	Address	SC	Address	Frame Body	CRC

Figure 6.5 IEEE 802.11 MAC frame

- The sequence control (SC) field consists of 4 bits reserved for fragmentation and reassembly and 12 bits for a sequence number of frames between a particular transmitter and receiver.
- The frame body field contains a MAC service data unit or control information.
- The cyclic redundancy check (CRC) field is used for error detection.

The three frame types in IEEE 802.11 are *control frames*, *data-carrying frames*, and *management frames*. Control frames ensure reliable data delivery. The types of control frames are

- Power save-poll (PS-Poll). A sender sends this request frame to the access point. The sender requests from the access point a frame that had been buffered by the access-point because the sender was in power-saving mode.
- Request to send (RTS). The sender sends an RTS frame to the destination before the data is sent. This is the first frame sent in the four-way handshake implemented in IEEE 802.11 for reliable data delivery.
- Clear to send (CTS). The destination sends a CTS frame to indicate that it is ready to accept data frames.
- ACK frame. The destination uses this frame to indicate to the sender a successful frame receipt.
- Contention-free end (CFE). The PCF uses this frame to signal the end of the contention-free period.
- CFE/End + CFE/ACK. PCF uses this frame to acknowledge the CFE end frame.

The data-carrying frames are of the following types:

- Data. This is the regular data frame and can be used in both the contention and contention-free periods.

- *Data/CFE-ACK*. This is used for carrying data in the contention-free period and is used to acknowledge received data.
- *Data/CFE-Poll*. PFC uses this frame to deliver data to destinations and to request data frames from users.
- *Data/CFE ACK/CFE-Poll*. This frame combines the functionalities of the previous three frames into one frame.

Management frames are used to monitor and manage communication among various users in the IEEE 802.11 LAN through access points.

6.3.3 WiFi Technology and 802.11

Wireless fidelity (WiFi)—a term trademarked by the WiFi Alliance—technology is a set of standards for wireless local area networks (WLANs). WiFi allows mobile devices, such as laptop computers, digital cameras, and personal digital assistants (PDAs), to connect to local area networks. WiFi is also intended for Internet access and wireless voice over IP (VoIP) phones. Computers can also have built-in WiFi, allowing offices and homes to be networked without expensive wiring.

Figure 6.6 shows three WiFi networks connected to the Internet through *wireless routers with gateway/bridge*. A wireless router with gateway/bridge is a router that can provide radio communications to certain wireless access points, as well as routing to wired Internet devices. Such router/gateways can also communicate with one another. For example, in Figure 6.6, routers A and B are communicating directly to establish

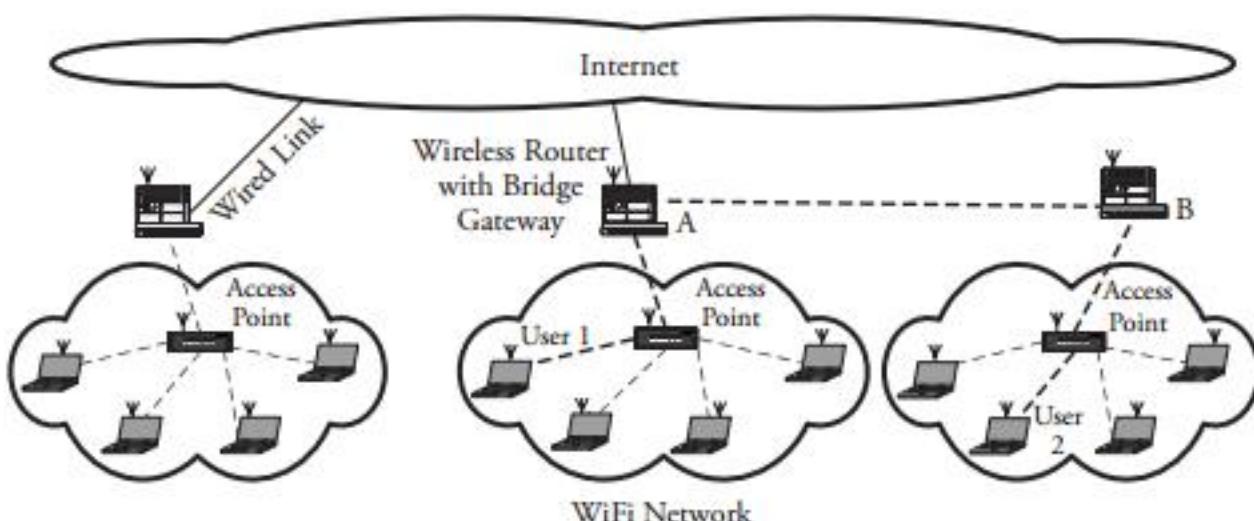


Figure 6.6 Connectivity of wireless users to WiFi access points to reach the Internet and direct wireless connections

connection for WiFi users 1 and 2; these two users could also be connected through the Internet. However, because an Ethernet uses contention access in WiFi, all users wanting to pass data through an access point contend for its attention on a random basis. This can cause nodes distant from the access point to be interrupted by closer nodes, resulting in reducing their throughput.

The connection is made by radio link signals. A *hotspot* is defined as an access point in a geographical region covered by WiFi. The range of an access point built into a typical WiFi home router is 50 meters indoors and 90 meters outdoors. WiFi is based on the IEEE 802.11 standard. The most widespread version of WiFi is based on IEEE 802.11b/g operating over 11 channels (5 MHz each), centered on Channel 1 at 2,412 MHz all the way to Channel 11 at 2,462 MHz. In the United States, maximum transmitter power is 1 watt, and maximum effective radiated power is 4 watts.

Several routing protocols are used to set up WiFi devices. One of these protocols is the *Optimized Link State Routing* (OLSR) protocol developed for mobile networks. OLSR operates as a table-driven and proactive protocol (see Chapter 19 for details). Thus, it regularly exchanges topology information with other nodes of the network. Nodes are selected as multipoint relays by some neighboring nodes. They exchange this information periodically in their control messages. With WiFi, most networks rely heavily on open source software or even publish their setup under an open source license.

WiFi allows LANs to be deployed without cabling, thereby lowering the cost of network deployment and expansion. WiFi may be used in places where cables cannot be laid. However, the use of the 2.4 GHz WiFi band does not require a license in most of the world, provided that one stays below the local regulatory limits and accepts interference from other sources, including interference that causes devices to no longer function.

However, the 802.11b and 802.11g standards over the 2.4 GHz spectrum used for WiFi are crowded with other equipment, such as Bluetooth devices, microwave ovens, and cordless phones. This may cause degradation in performance, preventing the use of open access points by others. In addition, the power consumption is fairly high compared to that for other standards, making battery life and heat a concern.

6.4 Cellular Networks

Cellular networks use a networked array of transceiver *base stations*, each located in a *cell* to cover the networking services in a certain area. Each cell is assigned a small frequency band and is served by a base station. Neighboring cells are assigned different frequencies

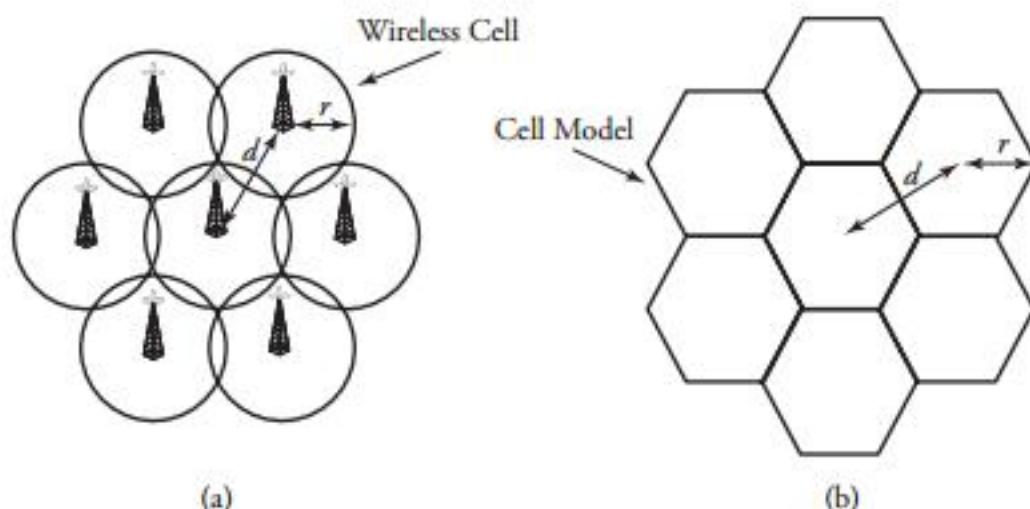


Figure 6.7 Cellular partitions: (a) real areas; (b) modeled areas

to avoid interference. However, the transmitted power is low, and frequencies can be reused over cells separated by large distances. The hexagonal pattern of a cell is chosen so that the distance d between the centers of any two adjacent cells becomes the same. Distance d is given by

$$d = \sqrt{3}r, \quad (6.1)$$

where r is the cell radius. A typical practical value for the cell radius is 3 km to 5 km. First-generation cellular networks were analog and used FDMA for channel access. Second-generation cellular networks were digital, using CDMA channel-access techniques. With the advent of the third- and later-generation cellular networks, multimedia and voice applications are supported over wireless networks. A covered area of a cellular network is visualized and approximated by a hexagonal cell served by its own antenna at the center of the hexagon, as shown in Figure 6.7.

6.4.1 Connectivity

Figure 6.8 shows the connectivity of two mobile users in cellular systems. A cell base station at the center of a cell consists of an *antenna*, a *controller*, and a *transceiver*. The base station is connected to the *mobile switching center* (MSC). An MSC serves several base stations and is responsible for *connecting calls* between mobile units. In Figure 6.8, an MSC is connected to two base stations, A and B. An MSC is also connected to the public telephone system to enable communication between a fixed subscriber and mobile subscriber. An MSC also manages mobility and accounting for user billing.

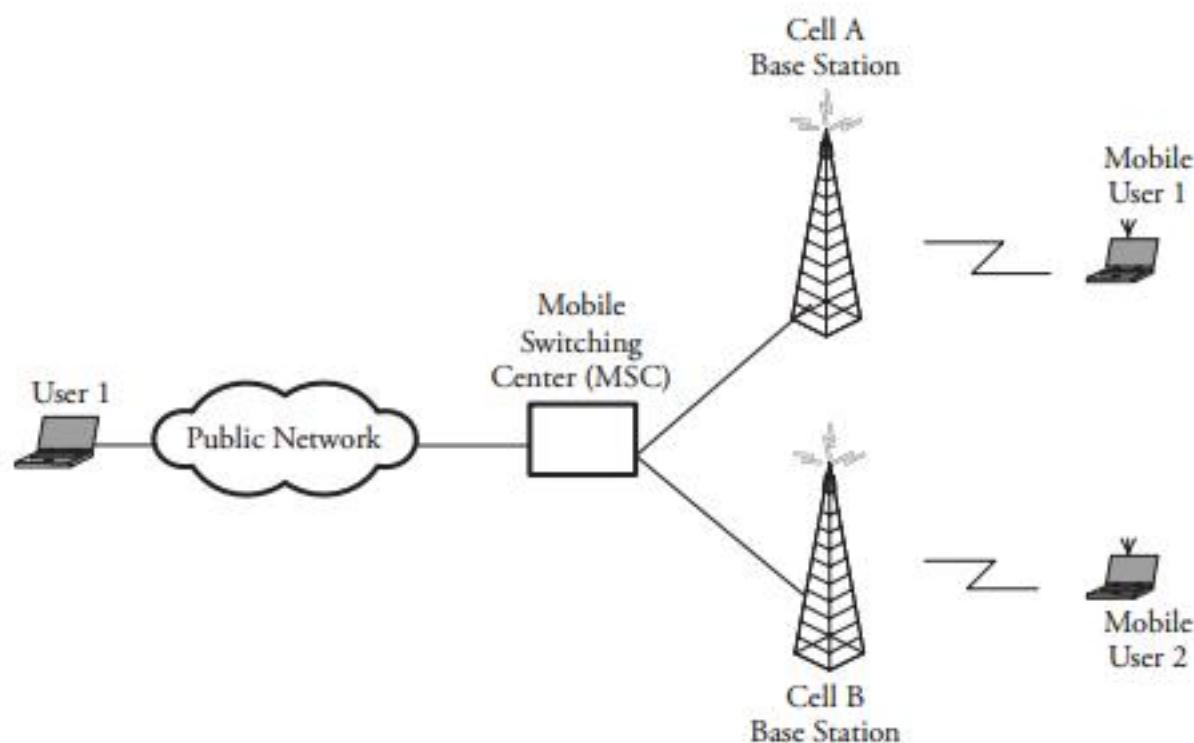


Figure 6.8 Connectivity of mobile users in cellular systems.

The steps involved in establishing a call between two mobile users in a cellular network are as follows:

1. *Mobile unit setup.* When the mobile unit is switched on, it searches for the strongest control channel. The mobile user is assigned a base station with which it operates. A handshake of messages takes place between the associated MSC and the mobile user through the base station. The MSC registers and authenticates the user through the base station. If the user moves to a new cell, this step repeats in the new cell.
2. *Originated call.* When a mobile originates a call, the called number is sent to the base station, from where it is forwarded to the MSC.
3. *Paging.* MSC pages specific base stations, based on the called number. The base stations in turn send a paging message on their set-up channel to locate the called user, as shown in Figure 6.9.
4. *Call accepting.* When the base station pages all users in its cell, the called user recognizes its number and responds. The base station then notifies the MSC, which sets up a connection between the called and calling base-station units.

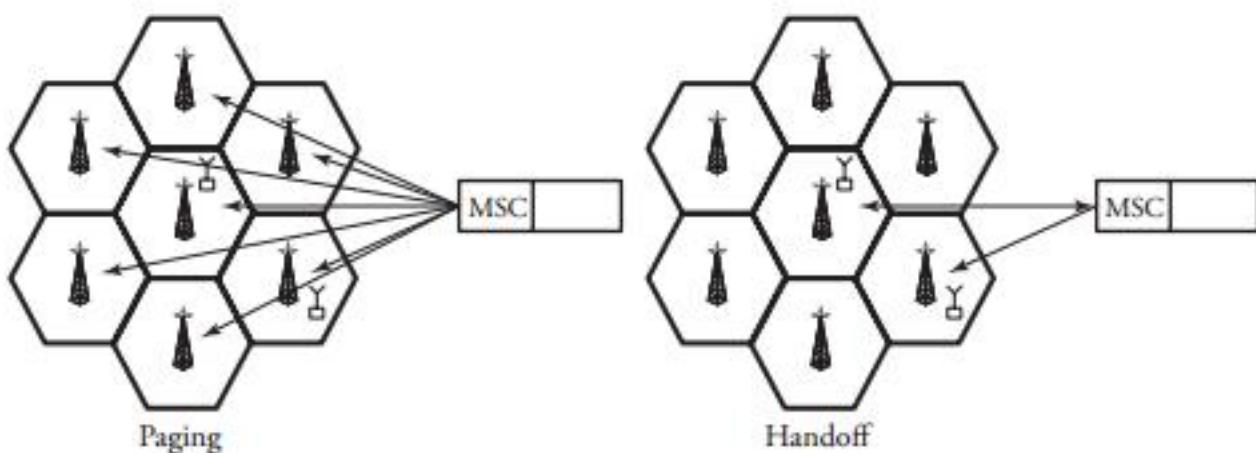


Figure 6.9 Basic operation of cellular systems

5. *Ongoing call*. Once the connection is established, exchange of data and voice occur between the two communicating mobile units through the base stations and the MSC.
6. *Handoff*. A handoff occurs when a mobile unit moves from one cell to another. The traffic channel switches to the new base station, using the MSC, as shown in Figure 6.9. This switch appears seamless to the user, without any interruption of the traffic.
7. *Call blocking*. When a mobile user originates a call, a busy tone is returned to the user if all the traffic channels to the base station are busy.
8. *Call termination*. When one of the users in a mobile conversation hang up, the MSC is informed of the call termination, and the traffic channels are deallocated at both base stations.
9. *Call drop*. When a base station cannot maintain a minimum signal level during a call, the call is dropped. Weak signals may occur because of interference or channel distortions.

The preceding operations use two types of channels for communication between the mobile and a base station, depending on the application of an operation. These two types of channels are *control channel* and *traffic channel*. Control channels are used for call setup and maintenance. This channel carries control and signaling information. Traffic channels carry the data between the users. Calls between fixed and mobile subscribers are also possible. An MSC can connect to the public telephone system and enable the connection between a mobile user and a fixed subscriber.

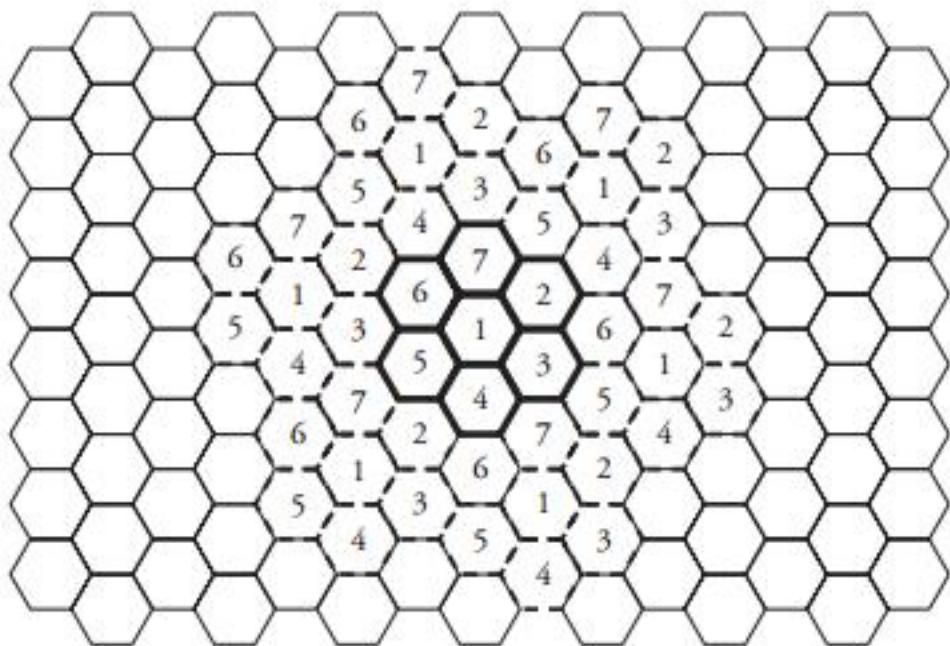


Figure 6.10 Cell clusters and frequency reuse among seven-cell clusters

6.4.2 Frequency Reuse

The basic idea of *frequency reuse* is that if a channel of a certain frequency covers an area, the same frequency can be reused to cover another area. The transmission power of the antenna in a cell is limited to avoid energy from escaping into neighboring cells. We define a *reuse cluster of cells* as N cells in which no frequencies are identical. Two *cochannel cells* are then referred to two cells in which a frequency in one cell is reused in the other one. Figure 6.10 shows a frequency-reuse pattern in a cellular network. In this example, each cluster has seven cells, and those cells with the same numbers are cochannel cells.

Let F be the total number of frequencies allotted to a cluster with N cells. Assuming that all cluster cells share an equal number of frequencies, the number of channels (frequencies) each cell can accommodate is

$$c = \frac{F}{N}. \quad (6.2)$$

A cluster can be replicated many times. If k is the number of times a cell is replicated, the total number of channels in the cellular system—so-called *system capacity*—is derived by

$$C = kF = kcN. \quad (6.3)$$

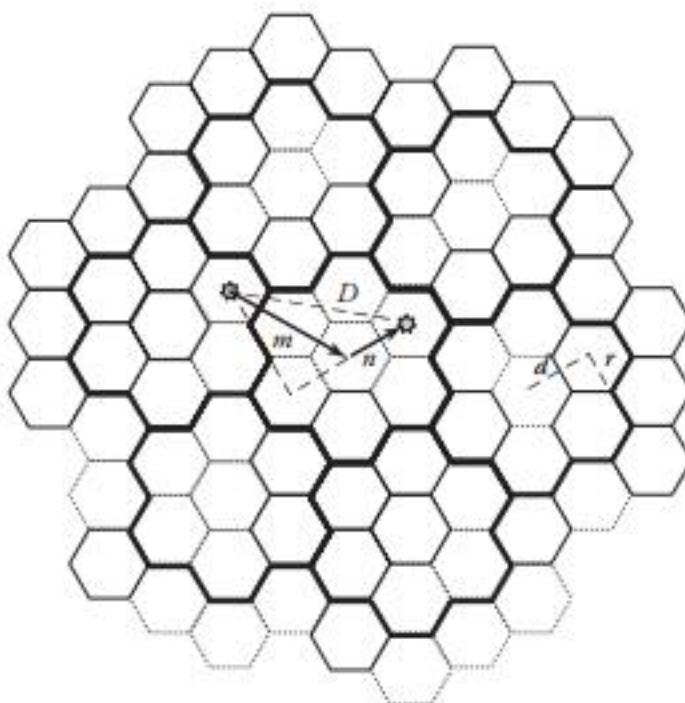


Figure 6.11 Nearest cochannel cells

In order to determine the location of a cochannel cell, we start from the center of a cell and move m cells or md km to any direction, then turn 60 degrees counterclockwise, and, finally, move n cells or nd km until we reach a cell with the same frequency. As shown in Figure 6.11, we have a case with $m = 3$ and $n = 1$. With a simple geometric manipulation of this situation, we can calculate the distance between the center of the nearest neighboring cochannel cell, D , shown in Figure 6.11:

$$\begin{aligned} D &= \sqrt{(nd \cos 30)^2 + (md + nd \sin 30)^2} \\ &= d\sqrt{m^2 + n^2 + mn}. \end{aligned} \quad (6.4)$$

As the distance between the centers of any two adjacent cells is $d = \sqrt{3}r$ according to Equation (6.1), we can rewrite Equation (6.4) as

$$D = r\sqrt{3(m^2 + n^2 + mn)}. \quad (6.5)$$

If we connect all the centers of all hexagons for “1” cochannel cells—the same arguments can be made for 2, 3, . . . cochannel cells—we make a larger hexagon, covering N cells with radius D . Knowing that the area of a hexagon is approximately

$2.598 \times (\text{square of its radius})$, we can derive the ratio of the areas of the r -radius hexagon, A_r , and D -radius hexagon A_D as

$$\frac{A_r}{A_D} = \frac{2.598r^2}{2.598D^2}. \quad (6.6)$$

Combining Equations (6.5) and (6.6), we calculate the area ratio as

$$\frac{A_r}{A_D} = \frac{1}{3(m^2 + n^2 + mn)}. \quad (6.7)$$

From the geometry, we can easily verify that the D -radius hexagon can enclose N cells plus $1/3$ of the cells from each six overlapping peripheral D -radius hexagons. Consequently, the total number of cells covered in a D -radius hexagon is $N + 6(1/3)N = 3N$. Then, $\frac{A_r}{A_D} = \frac{1}{3N}$, and Equation (6.7) can be simplified to

$$N = m^2 + n^2 + mn. \quad (6.8)$$

This important result gives an expression for the size of the cluster in terms of m and n . As the number of users in a cellular network increases, frequencies allotted to each cell may not be sufficient to properly serve all its users. Several techniques can be used to overcome these problems, as follows:

- *Adding new channels.* As networks grow in size and cells expand, channels can be added to accommodate a large number of users.
- *Frequency borrowing.* When a cell becomes congested, it can borrow frequencies from adjacent cells to handle the increased load.
- *Cell splitting.* In areas with a high volume of usage, cells are split into smaller cells. The power level of the transceiver is reduced to enable frequency reuse. But smaller cells increase the number of handoffs that take place as the user moves from one cell to another more often.
- *Cell sectoring:* A cell can be partitioned into sectors, with each sector containing its own subchannel.
- *Microcells.* In congested areas, such as city streets and highways, microcells can be formed. In a smaller cell, the power levels of the base station and mobile units are low.

Example. Consider a cellular network with 64 cells and a cell radius of $r = 2$ km. Let F be 336 traffic radio channels and $N = 7$. Find the area of each cell and the total channel capacity.

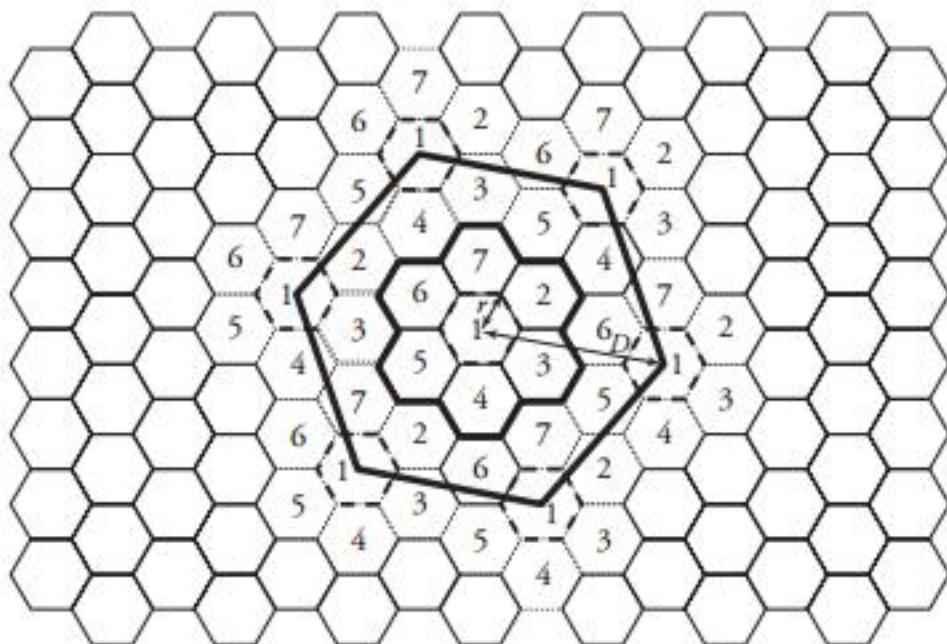


Figure 6.12 Forming a D -radius cluster by connecting all cell 1 centers

Solution. Each hexagon has an area of $1.5\sqrt{3}r^2 = 10.39 \text{ km}^2$. The total area covered by the hexagonal cells is $10.39 \times 64 = 664.69^2$. For $N = 7$, the number of channels per cell is $336/7 = 48$. Therefore, the total channel capacity is equal to $48 \times 64 = 3,072$ channels.

6.4.3 Local and Regional Handoffs

When it moves to a new cell, a wireless terminal requests a *handoff* for a new channel in the new cell. The increase in traffic volume and demand, as well as seamless, high-performance handoff in wireless systems are expected. A successful handoff operation requires certain criteria to be achieved. When a wireless terminal moves from one base-station cell to another, handoff protocols reroute the existing active connections in the new cell. The challenges in wireless networks are to minimize the packet loss and to provide efficient use of network resources while maintaining quality-of-service (QoS) guarantees. Robustness and stability must also be taken into consideration for handoff protocol design. Robustness and stability are especially important when interference or fading in the radio channel results in a request-handoff operation by a wireless terminal.

Cellular networks typically have three types of handoffs, as shown in Figure 6.13: *channel*, *cell*, and *regional*. *Channel handoff* involves transferring a call between channels in a cell. The wireless terminal first initializes a request for a channel change to

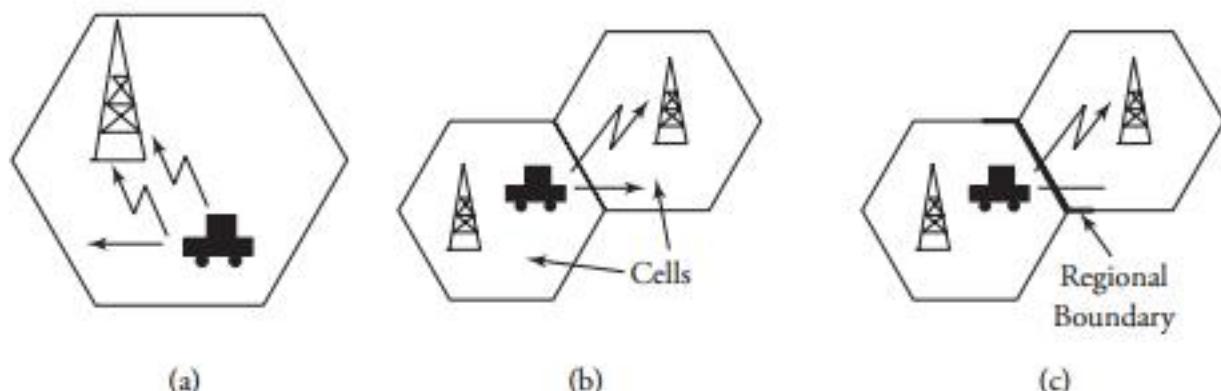


Figure 6.13 Cellular handoff types: (a) channel handoff, (b) cell handoff, and (c) regional handoff

the base station in a cell, if necessary. If no channels are idle in the cell, the request is rejected, and the wireless terminal has to keep the old channel.

Cell handoff occurs between two adjacent cells. When a wireless terminal moves from one cell to another, the handoff request to the new cell is initialized. If no channels are available in the new cell, the handoff call has to be rejected or terminated.

Regional handoff occurs when the mobile user moves from one region to another. From a theoretical standpoint, we can model a handoff process between any two regions, using stochastic models. Consider several hexagonal-shaped regions that consist of a group of hexagonal-shaped cells, as illustrated in Figure 6.14. Since the standard of handoff processes is still being developed, we have to assume that only the boundary cells in a region as labeled in Figure 6.14 are involved in the regional handoff model. Similarly, when all channels in the new region are in use, all handoff requests have to be rejected.

6.4.4 Mobility Management

Mobility management consists of three functions: *location management with user tracking*, *user authentication*, and *call routing*. Location management involves keeping track of the physical location of users and directing calls to correct locations. Before a user's call is routed to the desired location, the user must be authenticated. Routing involves setting up a path for the data directed to the user and updating this path as the user location changes. In cellular networks, *mobile switching centers*, in conjunction with base stations, coordinate the routing and location-management functions.

The requirement for handoff is dependent on the speed of the mobile terminals and the distance between a mobile user and its cell boundaries. The alternation of states for a mobile unit—whether it is still or mobile while carrying a call in progress—also has

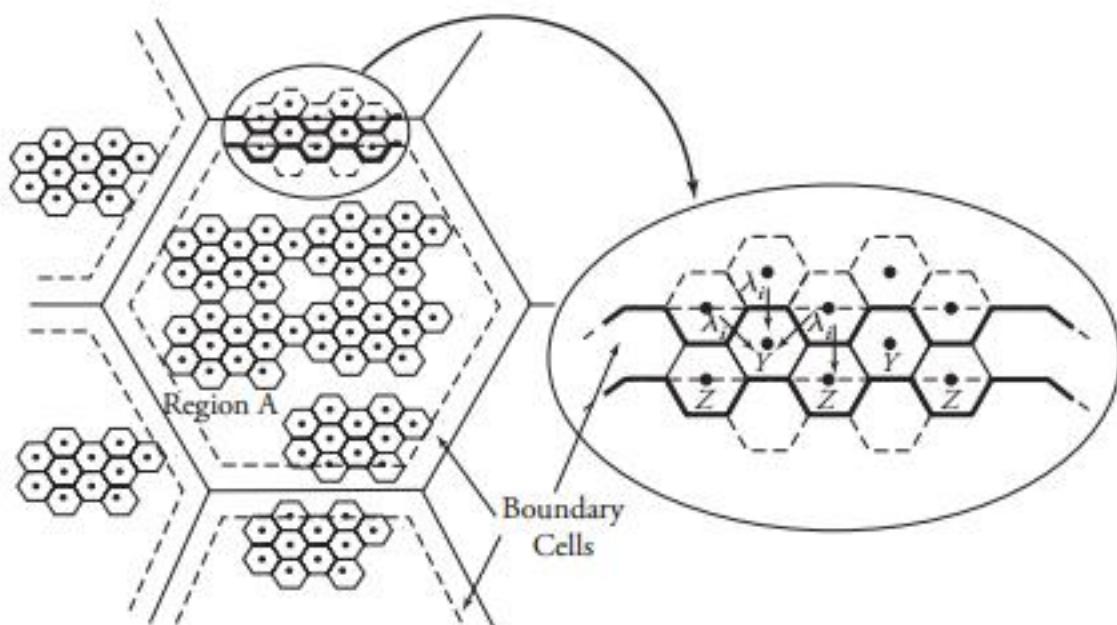


Figure 6.14 Cellular networks and a regional handoff

to be considered for handoff analysis. In reality, a handoff is needed in two situations: (1) the signal strength within the cell site is low or (2) when a vehicle reaches the cell boundary. We assume that the handoff model is free of signal-strength obstructions. Our second assumption prior to reaching a cell boundary is that a vehicle with a call in-progress alternates between still (stop state) and moving (go-state).

Stop-and-Go Model

The alternation of stop-and-go states can be modeled, using a simple state machine. In state 0 (stop), a vehicle is at rest but has a call in progress. In state 1 (go), the vehicle moves with an average speed of k mph and has a call in progress. Let $\alpha_{i,j}$ be the rate at which a system with i states moves from state i to state j . For example, $\alpha_{0,1}$ is the rate at which state 0 leaves for state 1, and $\alpha_{1,0}$ is the rate at which state 1 leaves for state 0. In our case, it is clear that $\alpha_{0,0} = -\alpha_{0,1}$ and that $\alpha_{1,1} = -\alpha_{1,0}$.

The time in the stop state is an exponential random variable with mean $1/\alpha_{0,1}$ (see Appendix C). The time in the go state also is an exponential random variable, with mean $1/\alpha_{1,0}$. Let $P_i(t)$ be the probability that a vehicle having a call in progress is in state i at time t . According to the *Chapman-Kolmogorov* theory on continuous-time Markov chain (explained in Section C.5.1), we can derive

$$P'_j(t) = \sum_i \alpha_{i,j} P_i(t), \quad (6.9)$$

where $P'_j(t)$ is the time differential of relative state j probability. Applying Equation (6.9) for a system with two states, 0 and 1, we have

$$\begin{aligned} P'_0(t) &= \alpha_{0,0}P_0(t) + \alpha_{1,0}P_1(t) \\ &= -\alpha_{0,1}P_0(t) + \alpha_{1,0}P_1(t) \end{aligned} \quad (6.10)$$

$$\begin{aligned} P'_1(t) &= \alpha_{0,1}P_0(t) + \alpha_{1,1}P_1(t) \\ &= \alpha_{0,1}P_0(t) - \alpha_{1,0}P_1(t), \end{aligned} \quad (6.11)$$

where $P_0(t)$ and $P_1(t)$ are the probability that a vehicle having a call in progress is in states 0 and 1, respectively, at time t . Knowing that the sum of probabilities is always 1, we get $P_0(t) + P_1(t) = 1$.

This equation and Equation (6.10) can be combined to form a first-order differential equation:

$$P'_0(t) + (\alpha_{0,1} + \alpha_{1,0})P_0(t) = \alpha_{1,0}, \quad P_0(0) = P_0. \quad (6.12)$$

The total solution to the differential equation consists of a homogeneous solution, $P_{0h}(t)$, plus a particular solution, $P_{0p}(t)$. Knowing that $P'_{0h}(t) + (\alpha_{0,1} + \alpha_{1,0})P_{0h}(t) = 0$, where $P_{0h}(0) = P_0(0)$, we can obtain the general solution for Equation (6.12) by

$$\begin{aligned} P_0(t) &= P_{0p}(t) + P_{0h}(t) \\ &= \frac{\alpha_{1,0}}{\alpha_{0,1} + \alpha_{1,0}} + \left(P_0(0) - \frac{\alpha_{1,0}}{\alpha_{0,1} + \alpha_{1,0}} \right) e^{-(\alpha_{0,1} + \alpha_{1,0})t}, \end{aligned} \quad (6.13)$$

where $P_0(t)$ is the probability that a vehicle with a call in progress is in state 0 at time t . Similarly, we obtain the general solution for Equation (6.11) by

$$\begin{aligned} P_1(t) &= P_{1p}(t) + P_{1h}(t) \\ &= \frac{\alpha_{0,1}}{\alpha_{0,1} + \alpha_{1,0}} + \left(P_1(0) - \frac{\alpha_{0,1}}{\alpha_{0,1} + \alpha_{1,0}} \right) e^{-(\alpha_{0,1} + \alpha_{1,0})t}, \end{aligned} \quad (6.14)$$

where $P_1(t)$ is the probability that a vehicle with a call in progress is in state 1 at time t .

There are four cases for a vehicle to change states.

1. A vehicle is resting permanently but has a call in progress; thus, $P_0(0) = 1$ and $P_1(0) = 0$.
2. A vehicle is moving at an average speed k until it reaches a cell boundary; thus, $P_0(0) = 0$ and $P_1(0) = 1$.

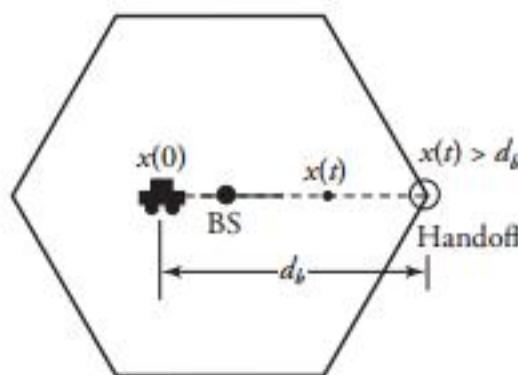


Figure 6.15 Cellular-handoff model with mobility

3. A vehicle stops at the initial state and moves on a congested path until reaching a cell boundary, so $P_0(0) = 1$ and $P_1(0) = 0$.
4. A vehicle moves and stops on a congested path until reaching a cell boundary; thus, $P_0(0) = 0$ and $P_1(0) = 1$.

Now, consider the mobilized model shown in Figure 6.15. To show the probability of a call to reach a cell boundary with an average speed k m/h with stop and go or the probability of requiring a handoff, let s be the average speed of the vehicle, where $s = 0$ in state 0 and $s = k$ in state 1. Let $x(t)$ be the vehicle's position at time t , assuming $x(0) = 0$, and let d_b be the distance a vehicle takes to reach a cell boundary. Suppose that t is a random variable representing a channel holding time, or the time a vehicle takes to reach a cell boundary. Let $P_i(t, d_b)$ be the probability that a vehicle in state i with a call in progress is at the cell boundary at time t , where $i \in \{0, 1\}$. Consequently, the probability that a vehicle reaches a cell boundary with speed s undergoing stop-and-go states is

$$P[x(t) \geq d_b] = P_0(d_b/s, d_b) \Big|_{s=0} + P_1(d_b/s, d_b) \Big|_{s=k}, \quad (6.15)$$

where

$$P_0(d_b/s, d_b) \Big|_{s=0} = \frac{\alpha_{1,0}}{\alpha_{0,1} + \alpha_{1,0}} \quad (6.16)$$

and

$$\begin{aligned} P_1(d_b/s, d_b) \Big|_{s=k} &= \frac{\alpha_{0,1}}{\alpha_{0,1} + \alpha_{1,0}} \\ &+ \left(P_1(0) - \frac{\alpha_{0,1}}{\alpha_{0,1} + \alpha_{1,0}} \right) e^{-(\alpha_{0,1} + \alpha_{1,0})d_b/k}. \end{aligned} \quad (6.17)$$

In case 1, since a vehicle is resting all the time, with an average speed of 0 m/h, the probability of reaching a cell boundary is clearly 0 percent. In contrast, for a vehicle moving with an average speed (k) (case 2), the chance of reaching a cell boundary is always 100 percent. Thus, when a vehicle is either at rest or moving, the probability of requiring a handoff is independent of d_b .

6.4.5 Generations of Cellular Systems

First-generation cellular systems were mostly analog. Channels were allotted to a single user, and each user had dedicated access to the channel. This led to underutilization of resources. Second-generation systems were digital and supported higher data rates, providing digital traffic channels and digitized voice before transmission over the channel. Data digitization made it simple to implement an encryption scheme. The digital traffic also made it possible to deploy better error detection and correction. Finally, multiple users shared a channel by using multiple-access schemes, such as TDMA or CDMA.

Third and later generations of wireless networks provide high data rates and support multimedia communications, in addition to voice communications. The main objectives for these cellular networks are to achieve quality voice communications, higher data rates for stationary and mobile users, support for a wide variety of mobile devices, and adapting to new services and technology usable in a wide variety of environments, such as offices, cities, and airplanes. Design issues involved the design of CDMA-based systems are *channel-usage bandwidth limitation* (5 MHz), *chip rate*, and multirate capability to provide different data rates on different channels for each user. The multirate scheme can scale effectively to support multiple applications from each user.

6.4.6 CDMA-Based Mobile Wireless

The most commonly used second-generation CDMA scheme is IS-95, which consists of two components: a *forward link* and a *reverse link*. The forward link consists of 64 CDMA channels, each operating at a bandwidth of 1.288 MHz. The channels are of four types:

1. *Pilot channel* (channel 0). This channel helps a mobile user to obtain timing information and enables the mobile unit to track signal-strength levels to initiate handoffs.
2. *Synchronization channel* (channel 32). This channel operates at 1,200 b/s and helps a mobile user to obtain information, system time and protocol version, from the cellular system.

3. *Paging channels* (channels 1–7). These channels are used for monitoring paging requests.
4. *Traffic channels* (channels 8–31 and 33–63). The forward link supports up to 55 traffic channels supporting data rates of up to 9,600 b/s.

If the data rate is low, bits are replicated to increase the rate to 19.2 Kb/s. This process, called *symbol repetition*, is followed by a scrambling process for increasing privacy and reducing the interference between users. The next step is a process to control the power output of the transmitting antenna. The data is then processed, using *direct-sequence spread spectrum* (DSSS) (see Chapter 4). This process spreads the transmitted signal from 19.2 Kb/s to 1.288 Mb/s. The *Walsh matrix* is used to generate the pseudorandom sequence. The signal is then modulated with the QPSK modulation scheme being transmitted onto the medium.

The *IS-95 reverse link* consists of up to 94 CDMA channels, each operating at 1.228 MHz. The link supports up to 32 access channels and 62 traffic channels. The access channels are used for call setup, location update, and paging. The reverse-link transmission process is very similar to forward-link transmission. The convolution encoder has a rate of 0.333. Thus, the data rate is tripled, to $3 \times 9.6 = 28.8$ Kb/s, followed by the data-scrambling process achieved by block interleaving. In the reverse link, the Walsh matrix is used to increase the data rate to 307.2 Kb/s as the data from the block interleaver is divided into 6-bit units to improve reception at the receiver. The data-burst randomizer is used in conjunction with the long code mask to reduce interference from other users. In the case of the reverse link, the long code mask is unique for each mobile unit. The resultant data is modulated, using the orthogonal QPSK modulator (OQPSK). The OQPSK scheme is used because the spreading codes need not be orthogonal for the reverse link.

Example. Consider a voice signal. It is first digitized at 8,500 b/s. Then error detection is added, which increases the number of bits and hence the data rate to around 9,600 b/s. During the idle periods of a conversation, the data rate can be lowered to around 1,200 b/s. The digitized voice signal is now transmitted in blocks of 20 ms interval. Forward error correction is provided by using a convolution encoder with rate of 0.5. This increases the data rate to $2 \times 9.6 = 19.2$ Kb/s.

6.5 Mobile IP

The *mobile IP* scheme is a protocol responsible for handling the mobility of users attached to the Internet. *Mobile computing* allows computing devices, such as computers, to move while functioning routinely. In a mobile IP network, it is essential for both

mobile and wired users to interoperate seamlessly. In most mobile IP cases, TCP cannot be used, as the congestion-control scheme would greatly reduce the throughput and the inherent delays and error bursts may result in a large number of retransmissions. Some changes have to be made to TCP to use it for internetworking wired and wireless networks. The major challenges with mobile IP are

- *Mobility.* A quality connection is desired for a user while it is mobile with different speeds.
- *Registration.* A mobile user's address must be identified and registered in different areas.
- *Interoperability.* A mobile user must interact with other stationary and mobile users.
- *Connection reliability.* TCP connections must survive in mobility cases.
- *Security.* A connection must be secured, especially since a wireless connection is less immune to intrusions.

A mobile user requires the same level of reliability for a TCP connection as he/she receives in a wired connection. Note that, typical Internet congestion-control schemes cannot be used in wireless networks, because the packet drop is caused mainly by poor link quality and channel distortions rather than by congestion. The channel imperfections make it difficult to implement a quality-of-service model other than the best-effort model. The varying data rates and delays make it challenging to implement high-speed and real-time applications, such as voice and video, over wireless networks.

6.5.1 Addresses and Agents

In a mobile IP network, a mobile host (mobile user) is allowed to hold two addresses simultaneously. One of the two addresses is permanent and the other is temporary. The permanent address of a host is the conventional IP address. Note that similar to regular networks, there are still MAC (at the link-layer) addresses in wireless networks that identify physical endpoints of links. A mobile host must have a permanent IP address in its *home network*. This address is called the *home address*. A home address is an IP address and is assigned to a mobile host for an extended period of time. The home address remains unchanged even if the host moves out of its home area. In such cases, a host needs to be registered by the home *mobile switching center* (MSC). This MSC is a router and it is called the *home agent*.

When a mobile host leaves its home network and enters a *foreign network*, the host must also be registered by the new network and obtain a temporary address. Changing

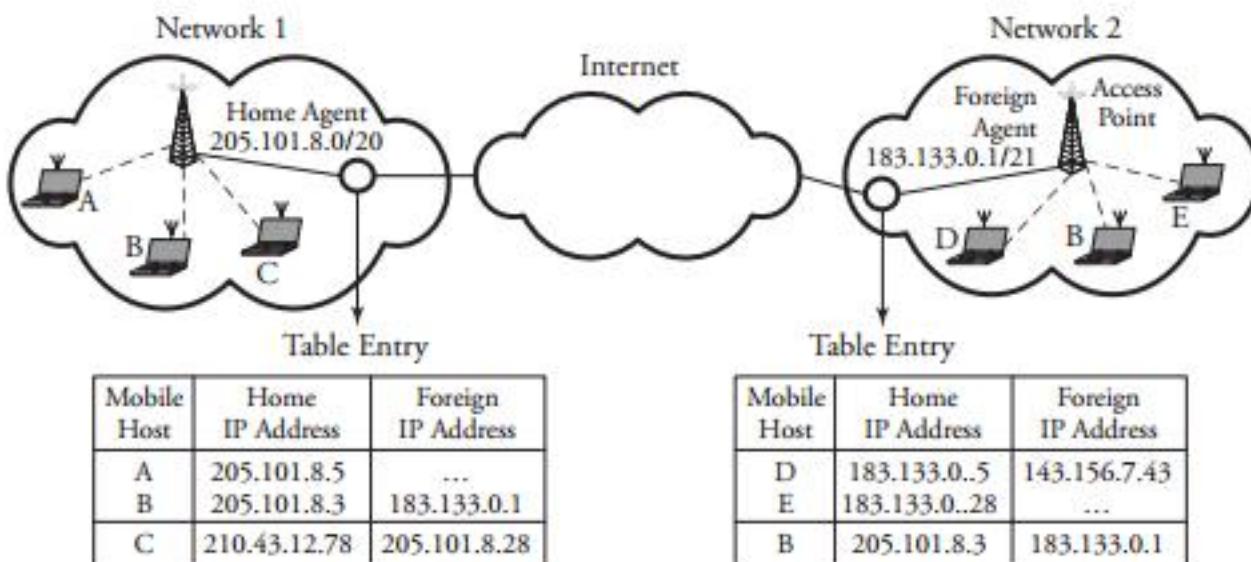


Figure 6.16 A mobile host moves from its home network to a foreign network

the network typically happens when a mobile host roams in a certain city or changes the city. Once a mobile host leaves its home network for a foreign network, it is assigned a *foreign address* reflecting the mobile host's current point of attachment when away from its home network. In such a case, its messages from the Internet corresponding servers are still sent to the mobile's home address. Similarly, a *foreign agent* is a router in the mobile host's foreign network that informs a host's home agent of its current foreign address. The home agent always forwards messages to the mobile host's current location. Figure 6.16 shows two wireless networks attached to the Internet in which mobile host B has moved from its home network to a foreign network.

Generally, MSC routers (acting as agents) in a network are connected through high-speed links to all access points (base stations) in a network. An MSC router maintains two databases: a home-location database and a foreign-location database. When a mobile host moves to its home network, a signal is sent to the local base station which forwards this signal to its MSC. The MSC router in turn authenticates the user and registers the user in its home-location database.

6.5.2 Agent Discovery Phase

A home agent maintains a database containing the mobile host's home address. When a mobile host moves to a foreign network, its home and foreign agents establish an association for updating registration with its home agent through the foreign agent. This association is made possible by sending agent advertisement messages. Upon receiving an agent advertisement, the mobile host can learn if it is located in its home network

or in a foreign network depending on the type of message. A mobile host can detect if it is connected to its home link or foreign link. Once the host moves to a new network, it can determine whether it has changed its point of attachment to obtain a foreign address.

Advertisement messages are propagated periodically in a broadcast manner by all agents. It is always possible that a mobile host does not receive the advertisement due to restricted timing. In such a case, the mobile host needs to send a request message to the agent which it is attached to. If the agent to which the host is attached is a home agent, the registration process is the same as traditional host in a fixed place. But, if the agent is a foreign one, the agent replies with a message containing a foreign address for the agent.

6.5.3 Registration

Mobile IP acts as an interface between the mobile's home network and the foreign network where the mobile currently resides. Mobile IP keeps track of the mobile's locations, and maps a home address into a current foreign address. The mobile IP interface delivers messages from the mobile's home network to the mobile host in its current foreign location in a seamless fashion after a registration process with the foreign agent is completed. The procedure of registration with a new network is summarized as follows:

Mobile IP Registration Steps

1. Use UDP (a transport layer protocol to be discussed in Chapter 8) and register with an agent on the new network
2. On the home network, register with an agent to request call forwarding
3. If any registration is about to expire, renew it
4. When returning to the home network, cancel the registration with the new network.

A registration phase involves an exchange of two messages between the mobile host and its home agent: *registration request* and *registration response*. Once a mobile host enters a foreign network, it listens for agent advertisements and then obtains a foreign address from the foreign network it has moved to. The host's home-network agent then adds the foreign network address agent to its home-location database. This is done after the agent authenticates the host through the host's home-network agent. The host's home-network agent now forwards all calls to the host in the foreign network. On the Internet, the location management and routing are done through mobile IP.

A mobile host can also register using a *collocated foreign address*. A collocated foreign address is a local IP address temporarily assigned to a mobile host without using a foreign agent. In a collocated foreign addressing, a mobile host receives an assigned temporary foreign address through its own home network. In the meanwhile, as soon as the mobile host leaves the foreign network, it also requires to deregister.

Example. Consider Figure 6.16 showing two wireless networks connected to the Internet. Network 1 is assigned a CIDR IP address (see Chapter 2) 205.101.8.0/20 and it has three active mobile hosts A, B, and C. Suppose that this network is the home network for hosts A and B and but not for host C as it appears from the home IP addresses of the agent routing entry. Consider a situation in a different time in which host A stays in this network (thus there is no foreign address for it), and host B moves out of this network (thus it obtains a foreign address). Particularly, host B has moved to network 2. Network 2 is also assigned a CIDR IP address 183.133.0.1/21 and it has three active mobile hosts D, E, and B. Network 2 is now considered a foreign network for B and is therefore assigned a foreign address of 183.133.0.1. This address appears in its both associated home and foreign agents as seen in the figure.

6.5.4 Mobile IP Routing

In mobile IP systems, datagrams are encapsulated by a mobile IP header. Figure 6.17 shows the header format of mobile IP registration. The *type* field determines whether the registration is a request or a reply. The *flags/code* field is used in the reply-message to specify forwarding details. The *lifetime* field gives the permitted time (in seconds) a registration is valid. The *home address* and *temporary address* fields are the two addresses explained. The *home agent* field specifies the home-agent address of the host. The *identification* field helps a mobile host prevent repeated messages.

Each datagram forwarded to the mobile host's home address is received by its home agent, and then it is forwarded to the mobile host's foreign address. In this case, mobile host's foreign agent receives the datagram and forwards it to the mobile host. If a mobile

Byte:		1	1	2	4	4	
Type	Flags/Code	Lifetime	Home Address	Temporary Address (Only for Request)	Identification	Extensions	

Figure 6.17 Header format of mobile IP registration

host residing in a foreign network wants to send a message to host outside of its new network, the message is not required to be passed through its home agent. In such a case, the message is handled by the foreign agent.

Mobile IP has two routing schemes: *Delta routing* and *direct routing*. In the Delta routing, a triangular path is established among the host's home agent, host's foreign agent, and a corresponding machine. Suppose that the mobile host belonging to wireless network 1 moves to foreign network 2. While in the foreign network, the mobile host is contacted for communication by a server (as a corresponding machine) fixed in a residential area network. In this case, a datagram (IP packet) from the server is first sent to the mobile's home network using standard IP routing. The host's home agent detects the message, finds the host's foreign address, and forwards the message to the host's foreign agent. The foreign agent delivers the message to the mobile host. In response, the mobile host can send its reply directly to the server through the foreign agent. This routing process forms a triangular-shape path routing and that is why it is called Delta routing.

Now, consider a case that a mobile host is the one who wants to initiate the transmission of a message with a server in the same scenario as explained above. In the first step, the mobile host informs the server of its foreign address. Then, the mobile host can send the message directly to the server through its foreign agent bypassing its home agent. This way a chunk of signaling due to routing to home agent is eliminated. We remember that the corresponding server should initiate a communication with a mobile host always starting to contact the mobile host's home agent since the server does not have a real-time knowledge of the mobile host's whereabouts.

As we see, the routing of mobile users may involve many different challenges. For example, in the previous scenario, if the mobile host moves to yet a new foreign network, say network 3. In this case, the mobile host can inform its previous foreign agent about its new foreign address, so that datagrams (IP packets) routed to the old location can now be routed to the new foreign location.

Virtual Registration and Routing

In order to reduce the cost and the amount of registration with the home agent, the mobile Internet protocol also offers facility called *virtual registration*. In each region, *virtual agents* instead of just a home agent can be defined. Virtual regions are then defined based on statistics and the density of traffic. Each virtual agent covers services over a local virtual region. When a mobile host enters the virtual region, it registers with the virtual agent. Thus, in a scenario of routing messages between a mobile host and a corresponding server described in the previous section, datagrams from

the corresponding server are sent to the mobile's home address, and then routed to the mobile's foreign address. Datagrams are then sent from the home agent to the virtual agent first and, from there, to the foreign agent. In such cases, the mobile host has typically no knowledge of the network for routing decision making.

Tree-Based Routing

The amount of registration between a home network and a foreign network can also be reduced by a carefully designed hierarchy of foreign agents. In a hierarchical structure, multiple foreign agents are advertised in the agent advertisement. With this scheme, a mobile host has to configure to what upper level at the tree its new registration has to go. The mobile host should then transmit the registration to each level of the hierarchy between itself and the closest common parent between its new and previous foreign addresses. If a mobile host currently using the services of one foreign agent moves to a different foreign agent, it may not involve a direct registration with its home agent.

Figure 6.18 shows a tree-based hierarchy of foreign agents. Suppose that a mobile host is currently using the service of foreign agent A16 while at location L1. The mobile host receives agent advertisements from foreign agents A1, A2, A4, A7, A11, and A16. Registration messages are sent to each of these foreign agents and its home agent. However, the home agent of the mobile host can only identify foreign agents in its outside world as far as to foreign agent A1. This means that, the topology of the hierarchy beyond A1 may stay unknown for the home agent even though it receives messages from other agents. The same thing is true for agent A1 which can see only up to its nearest neighbors A2 and A3, and so on for others. In fact, no agent knows exactly where the mobile host is located except for foreign agent A16.

When the mobile host moves to the vicinity of foreign agent A17 at location L2, the host needs a new registration valid to travel upto the vicinity of A11. If the mobile moves to the vicinity of foreign agent A19 at location L3, the situation is different as A17 and A19 are linked directly to a common node as was the case for A16 and A17. In this case, the mobile host receives advertisements specifying the hierarchy of A4, A8, A13, and A19. The mobile host then compares the previous hierarchy and this new one and determines that it has caused the registration to move to as high as level A4 in the tree-based scheme. The same procedure occurs when the mobile host decides to move to location L4.

Mobile Routing with IPv6

Mobile IPv6 offers a simpler mobile routing scheme. With IPv6, no foreign agent is required. A mobile host should use the *address autoconfiguration procedure* embedded

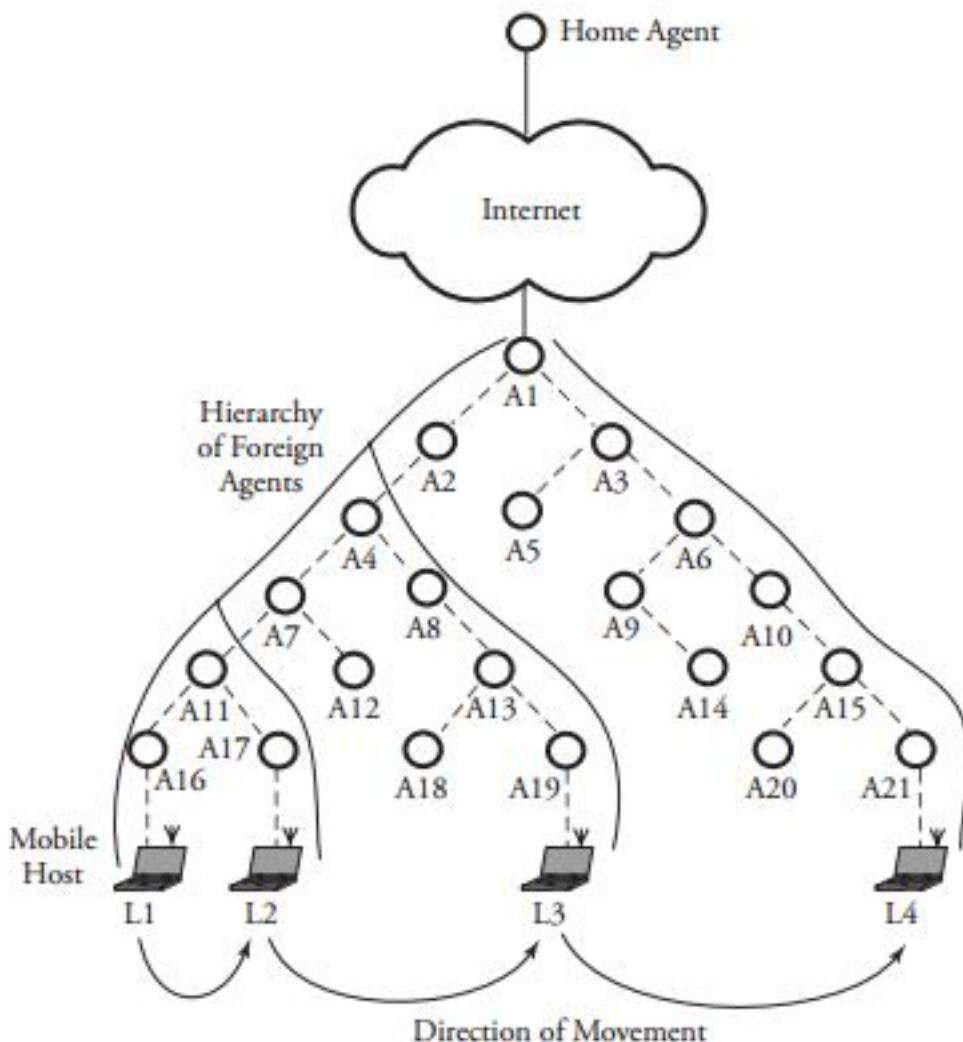


Figure 6.18 Routing in a tree-based structure of foreign agents

in IPv6 to obtain a foreign address on a foreign network. The procedure to route with mobile IPv6 is summarized as follows:

Mobile IPv6 Routing Steps

1. A host informs its home agent and also corresponding machines about its foreign address.
2. If a corresponding machine knows the mobile's current foreign address, it can send packets directly to the mobile host by using the IPv6 routing header; Otherwise the corresponding machine sends packets without the IPv6 routing header.
3. Packets are routed to the mobile host's home agent
4. Packets are forwarded to the mobile host's foreign address
5. If the mobile host moves back to its home network, the host notifies its home agent. ■

It is clear that the routing steps are similar to IPv4 ones except for the elimination of foreign agent in IPv6. Overall, routing with IPv6 is simpler and the option of source routing is also available.

6.5.5 Security

Wireless links are susceptible to eavesdropping or passive traffic monitoring. The inherent broadcast paradigm in wireless networks make them more susceptible to various attacks. The security for these networks involve:

- Network security
- Radio link security
- Hardware security

Radio link security involves preventing the interception of radio signals, defense against jamming attacks, and encrypting traffic to ensure privacy of user location. The security portion of wireless system must prevent the misuse of mobile units by making them tamper resistant. The hardware component of wireless security is especially complex. The details of security in wireless networks are discussed in Chapter 10.

6.6 Wireless Mesh Networks (WMNs)

A *wireless mesh network* (WMN) is a dynamically self-organized wireless network that maintains *mesh* connectivity. WMNs help users stay online anywhere, anytime, for an unlimited time. One key component that makes this happen is the type of wireless router used in mesh infrastructures. We look at applications of WMNs, and WiMax networks, the conductivities of P2P networks with backbone wireless mesh networks.

6.6.1 WiMAX Technology and IEEE 802.16

The *worldwide interoperability for microwave access* (WiMAX) technology is a certification mark for the IEEE 802.16 standard. This standard is implemented for point-to-multipoint broadband wireless access. WiMAX is a wireless WAN technology that can connect IEEE 802.11 WiFi hotspots with one another and to other parts of the Internet. WiMAX also provides a wireless alternative to cable and DSL for broadband access. WiMAX devices are capable of forming wireless connections to allow Internet packets to be carried across a network. Conceptually, WiMAX is similar to WiFi technology but has been improved for use over much greater distances.

The IEEE 802.16 standard offers a significant improvement for communications, as it defines a MAC layer that supports multiple physical-layer specifications, potentially making WiMAX a great framework for wireless broadband communications. The 802.16 MAC is a scheduling MAC whereby the user device competes once for initial entry into the network. After being in the network, the base station allocates a time slot to the user. This time slot can enlarge or constrict, and no other users can use it. Unlike 802.11, the 802.16 scheduling algorithm exhibits stability given overload and offers better bandwidth efficiency. Another advantage is that 802.16 lets the base station offer QoS by balancing the assignments of users.

The IEEE 802.16 standard has determined the frequency range of 10 GHz to 66 GHz. WiMAX improves on the WiFi standard by providing increased bandwidth and stronger encryption. WiMAX makes excellent use of multipath signals. IEEE 802.16 dictates up to 50 km of connectivity services between users without a direct line of sight. This does not mean that a user 50 km away with no line of sight has connectivity, and practically, this distance is 5 km to 8 km. The data rate with WiMAX is up to 70 Mb/s, which is sufficient to simultaneously support more than 60 businesses with T-1-type connectivity. The line of sight is about 1,000 homes at 1 Mb/s DSL-level connectivity.

WiMAX antennas can share a cell tower without impacting the normal operations of the cellular network. A WiMAX antenna can even be connected to an Internet backbone via optical fibers or directional microwave link. WiMAX may be considered for cities or countries willing to skip a wired infrastructure, establishing a wireless infrastructure in an inexpensive, decentralized, deployment-friendly, and effective manner.

6.6.2 Applications of Mesh Networks

In WMNs, two types of nodes perform the routing: *mesh routers* and *mesh users*. Figure 6.19 shows detailed connectivity of a backbone mesh network to WiFi, WiMAX, and wireless cellular networks. In this figure, a WiFi network, a cellular network, and a WiMAX network are connected through mesh routers with *gateway bridges*. A router with gateway bridge capability enables the integration of WMNs with other type networks, although traditional routers with regular *network interface cards* (NICs) can connect to mesh networks.

Mesh users can also operate as routers for mesh networking, making the connectivity much simpler and faster than conventional wireless networks with base stations. Figure 6.20 shows another scenario, in which a wireless mesh network backbone is connected to wireless mesh users. Users are communicating in an ad hoc fashion, with each individual user acting as a router and connected to a mesh router gateway. In this

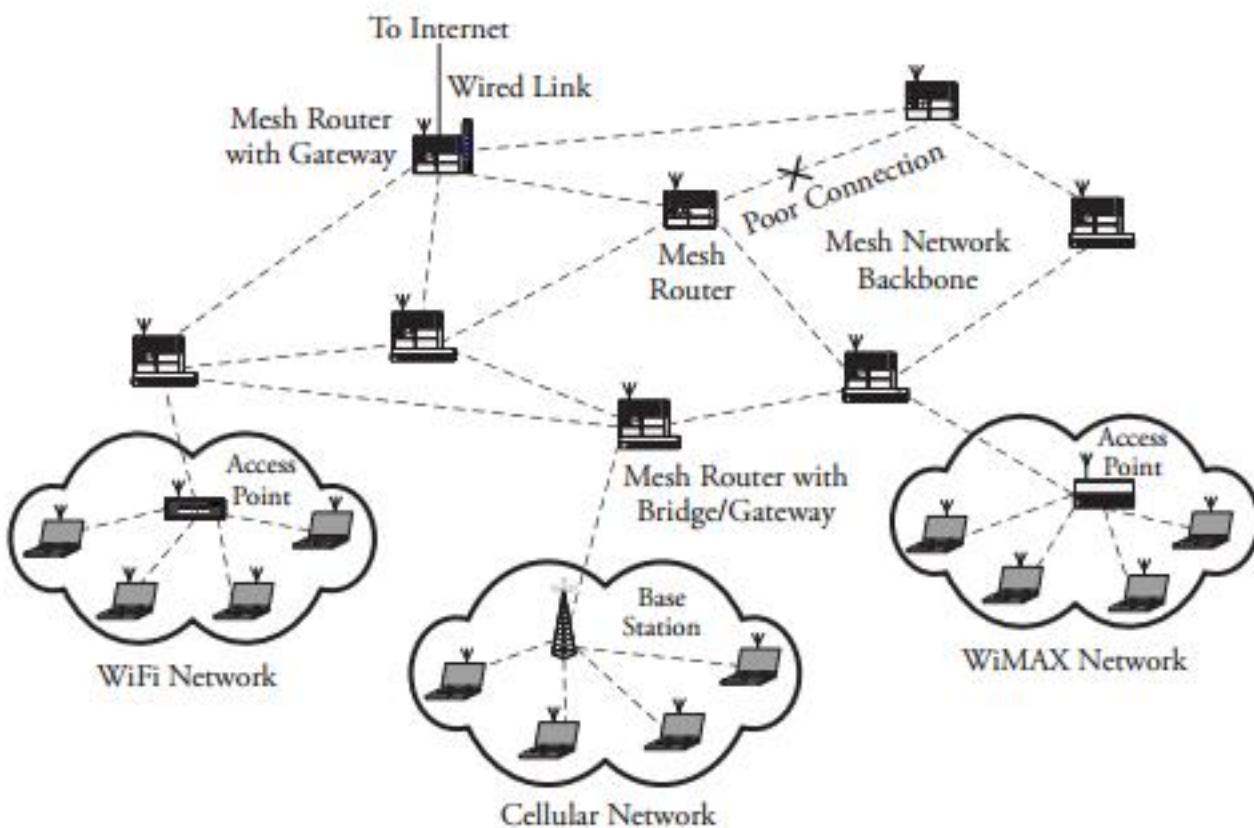


Figure 6.19 Overview of a backbone mesh network and connections to WiFi, WiMAX, and wireless cellular networks

network, wired users, as shown by a LAN in the figure, can also be connected to WMN, using a mesh router gateway. User meshing provides *peer-to-peer* networks among users (discussed in Chapter 16).

The inclusion of multiple wireless interfaces in mesh routers significantly enhances the flexibility of mesh networks. WMNs offer advantages of low cost, easy network maintenance, and remarkably more reliable service coverage than conventional ad hoc networks. Mesh networks are being designed for metropolitan and enterprise networking, and most of standards, such as IEEE 802.11, IEEE 802.15, and IEEE 802.16, are accepted in WMN infrastructures. A widely accepted radio technology is the series of IEEE 802.11 standards.

The benefits of a mesh network are as follows:

- *Scalability*. The WMN infrastructure is designed to be scalable as the need for network access increases.
- *Ad hoc networking support*. WMNs have the capability to self-organize and be connected to certain points of ad hoc networks for a short period of time.

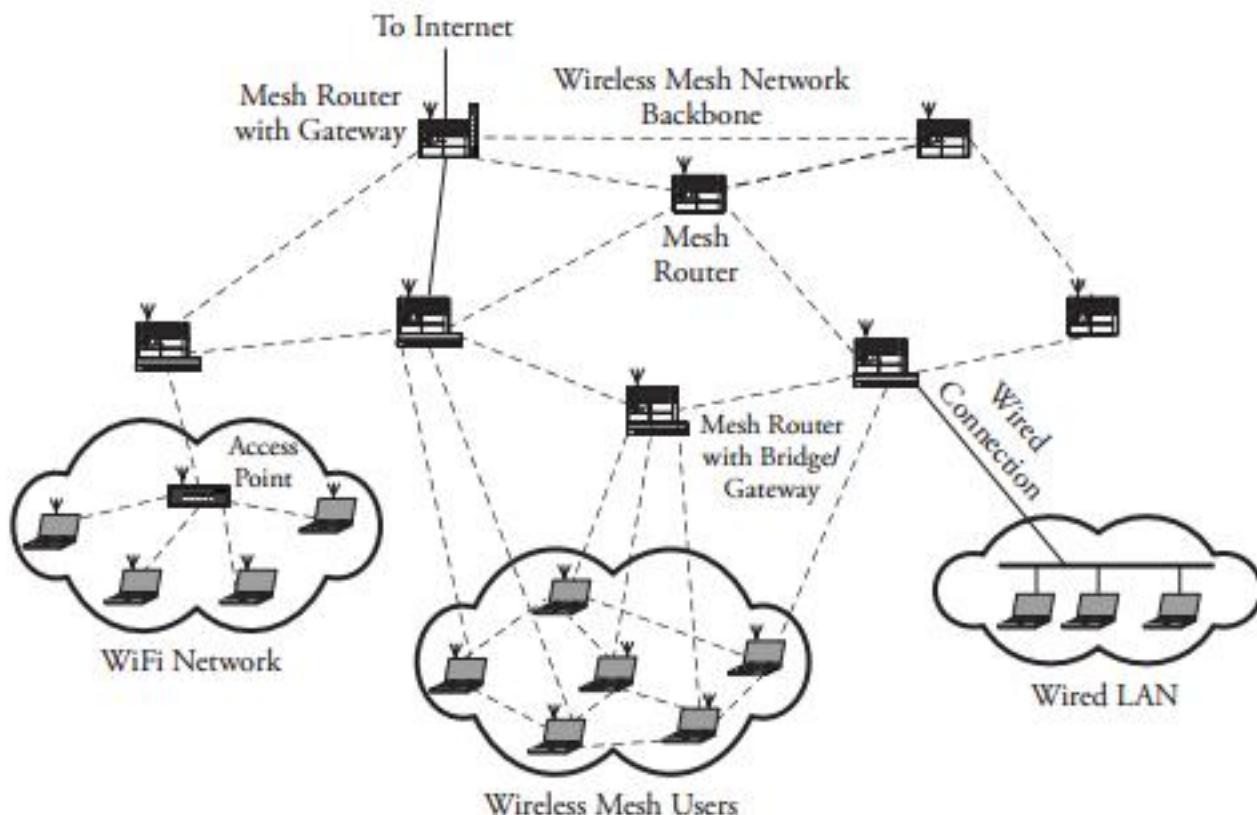


Figure 6.20 Connectivity between a backbone wireless mesh network to wireless users and other networking devices

- *Mobility support of end nodes.* End nodes are supported through the wireless infrastructure.
- *Connectivity to wired infrastructure.* Gateway mesh routers may integrate heterogeneous networks in both wired and wireless fashions.

To achieve scalability in WMNs, all protocols—from the MAC layer to the application layer—must be scalable. “Topology-” and “routing-aware” MAC can substantially improve the performance of WMNs.

The QoS provisioning in wireless mesh networks is different from that of classical ad hoc networks. Several applications are broadband services with heterogeneous QoS requirements. Consequently, additional performance metrics, such as delay jitter and aggregate and per node throughput, must be considered in establishing a route. Application-specific security protocols must also be designed for WMNs. Security protocols for ad hoc networks cannot provide any reliability, as the traffic in such networks can resemble the one flowing in the wired Internet.

6.6.3 Physical and MAC Layers of WMNs

The *physical layer* of wireless mesh networks benefits from existing modulation and coding rates. *Orthogonal frequency multiple access* (OFDM) and *ultrawide band* (UWB) techniques are used to support high-speed wireless communications.

Physical Layer

WMN communication quality and system capacity have been improved through the use of multiantenna systems, such as antenna diversity, smart antenna, and MIMO (*multiple-input multiple-output*) systems. MIMO algorithms send information over two or more antennas. The radio signals reflect objects, making multiple paths that in conventional radios cause interference and fading. A MIMO system uses these paths to carry more information. Another improvement in WMNs includes the use of *cognitive radios*, which dynamically capture unoccupied spectrum. One of the unique features of this technology is that all components of its radio, including RF bands, channel-access modes, and even channel modulations, are programmable.

MAC Layer

The WMN MAC layer is different from classical wireless networks. In WMNs, the MAC layer

1. Is designed to face more than one-hop communication
2. Is distributed to support multipoint-to-multipoint communications
3. Has self-organization features
4. Has moderately lower mobility than in classical wireless networks

Wireless mesh MAC protocols can be designed for both a single channel or multiple channels or to even operate simultaneously. A multiple-channel MAC setup improves network performance significantly by increasing network capacity. Because of the poor scalability of CSMA/CA schemes, these techniques are not efficient solutions for single-channel MAC protocols. The best solution for WMNs is the enhanced versions of TDMA or CDMA, owing to low complexity and cost.

Multichannel MACs can be deployed in various ways. With *multichannel single-transceiver* MAC, only one channel can be active at a time in each network node, as only one transceiver is available. With *multichannel multitransceiver* MAC, several channels can be active simultaneously and only one MAC-layer module is assigned to coordinate all channels. With *multiradio* MAC, each node has multiple radios, and each radio has its own MAC layer and physical layer.

6.7 Summary

This chapter presented basics of wireless networking without touching on the large-scale routing issues. Starting with the fundamental concept at the LAN level, we analyzed several IEEE 802.11 standards. The basic versions—802.11a, 802.11b, and 802.11g—typically use *Carrier Sense Multiple Access with collision avoidance* (CSMA/CA).

A *cellular network*, includes a networked array of *base stations*, each located in a hexagonal *cell* to cover networking services. Each mobile user should register with the regional *mobile switching center*. Because of unexpected interference whenever a user works with radio frequencies, we looked at known interference and *frequency reuse*. Frequency reuse in a certain region of a wireless network occurs when the same frequency used in one area could be reused to cover another area.

We also studied *mobile IP*? We learned that this protocol is responsible for handling the mobility of users attached to the Internet. A mobile host is allowed to hold two addresses simultaneously: a home address and a foreign address. One of the main elements of mobile IP is registration in a foreign network.

At the end of this chapter, we introduced *wireless mesh networks* (WMNs), including WiFi and WiMAX technologies. The *wireless fidelity* (WiFi) technology is a set of standards for wireless local area networks that allows mobile devices, such as laptop computers, digital cameras, and personal digital assistants, to connect to local area networks. The *world wide interoperability for microwave access* (WiMAX) technology is the IEEE 802.16 standard that can connect IEEE 802.11 WiFi hotspots to one another and to other parts of the Internet. We also looked at *wireless mesh networks* (WMNs) constructed as a wireless network backbone for several applications.

In the next chapter, we study routing and internetworking issues in widearea networks. We explore how routing algorithms and protocols are performed both within and beyond a single wide area network.

6.8 Exercises

1. Consider a commercial wireless mobile telephone system whose transmitter and receiver are located 9.2 km apart. Both use isotropic antennas. The medium through which the communication occurs is not a free space, and it creates conditions such that the path loss is a function of d^3 and not d^2 . Assume that the transmitter operating at the frequency of 800 MHz communicates with a mobile receiver with the received power of 10^{-6} microwatts.
 - (a) Find the effective area of the receiving antenna.
 - (b) Find the required transmission power.

2. Assume that cellular networks were modeled by square cells
 - (a) Find the cell coverage area, and compare it to the one using hexagonal cells. Assume that the distance between the cell centers are identical in these two models.
 - (b) What are the disadvantages of this model compared to the one with hexagonal cells?
3. A cellular network over $1,800 \text{ km}^2$ supports a total of 800 radio channels. Each cell has an area of 8 km^2 .
 - (a) If the cluster size is 7, find the system capacity.
 - (b) Find the number of times a cluster of size 7 must be replicated to approximately cover the entire area.
 - (c) What is the impact of the cluster size on system capacity?
4. Consider a cellular network with 128 cells and a cell radius $r = 3 \text{ km}$. Let g be 420 traffic channels for a $N = 7$ -channel cluster system.
 - (a) Find the area of each hexagonal cell.
 - (b) Find the total channel capacity.
 - (c) Find the distance between the centers of nearest neighboring cochannel cells.
5. If cells split to smaller cells in high-traffic areas, the capacity of the cellular networks for that region increases.
 - (a) What would be the trade-off when the capacity of the system in a region increases as a result of cell splitting?
 - (b) Consider a network with 7-cell frequency reuse clustering. Each cell must preserve its base station in its center. Construct the cell-splitting pattern in a cluster performed from the center of the cluster.
6. We would like to simulate the mobility and handoff in cellular networks for case 4 described in this chapter. Assume $25 \text{ mph} \leq k \leq 45 \text{ mph}$ within city and $45 \text{ mph} \leq k \leq 75 \text{ mph}$ for highway. Let d_b be the distance a vehicle takes to reach a cell boundary, ranging from -10 miles to 10 miles.
 - (a) Plot the probability of reaching a cell boundary for which a handoff is required. Discuss why the probability of reaching a boundary decreases in an exponential manner.
 - (b) Show that the probability of reaching a cell boundary for a vehicle that has a call in progress is dependent on d_b .
 - (c) Show the probabilities of reaching a cell boundary as a function of a vehicle's speed.

- (d) Discuss why the probability of reaching a cell boundary is proportional to the vehicle's speed.
7. *Computer simulation project.* Consider again the mobility in cellular networks for case 4 described in this chapter, but this time, we want to simulate the handoff for three states: stop, variable speed, and constant speed. For the variable-speed case, assume that the mobile user moves with a constant acceleration of K_1 m/h. Assume $25 \text{ m/h} \leq k \leq 45 \text{ m/h}$ within city and $45 \text{ m/h} \leq k \leq 75 \text{ m/h}$ for highway. Let d_b be the distance a vehicle takes to reach a cell boundary, ranging from -10 miles to 10 miles.
- Plot the probability of reaching a cell boundary for which a handoff is required. Discuss why the probability of reaching a boundary decreases in an exponential manner.
 - Show that the probability of reaching a cell boundary for a vehicle that has a call in progress is dependent on d_b .
 - Show the probabilities of reaching a cell boundary as a function of a vehicle's speed.
 - Discuss why the probability of reaching a cell boundary is proportional to the vehicle's speed and that the probability of requiring a handoff decreases.

CHAPTER 7

Routing and Internetworking

This chapter focuses on the networking structure of larger networks. One of the most important functions in computer networking, especially in wide area networks, is *routing* packets. In packet-switched networks, this function includes procedures through which a packet uses certain algorithms to find all the necessary paths from its source to its destination. In this chapter, we look at routing algorithms and protocols both within one WAN (*intradomain networking*, or *intranetworking*) and beyond it (*interdomain networking*, or *internetworking*). This chapter covers the following main topics:

- *Network-layer routing*
- *Classification of routing algorithms:*
 - *least-cost-path algorithms*
 - *non-least-cost-path routing*
 - *intradomain routing protocols*
 - *interdomain routing protocols*
- *Network-layer congestion control*

We begin with some basic material about routing, such as the definition of path cost and the classification of routing algorithms. A networking infrastructure deploys a variety of algorithms for routing packets, classified as those using optimal routes and those using nonoptimal routes. We also classify routing protocols by whether they are applied within a domain or beyond a domain.

We also look at congestion-control mechanisms that can be implemented either unidirectionally or bidirectionally among nodes at the network layer. At the end of the discussion on congestion control, an approximation method for calculating *link blocking* is introduced. This method provides a quick approximation method for the performance evaluation of links.

7.1 Network-Layer Routing

Routing algorithms create procedures for routing packets from their sources to their destinations. Routers are responsible mainly for implementing routing algorithms. These routing tasks are essentially methods of finding the best paths for packet transfer in a network. The choice of routing protocol determines the best algorithm for a specific routing task. As seen in Figure 7.1, a host and a server are two end points connected through routers R4, R7, and R1. Each end point belongs to a separate LAN. In such scenarios, a layer 3 (network) route must be set up for IP packets (datagrams); all devices, including end users and routers, process the route at the third layer of the protocol stack, as shown in the figure.

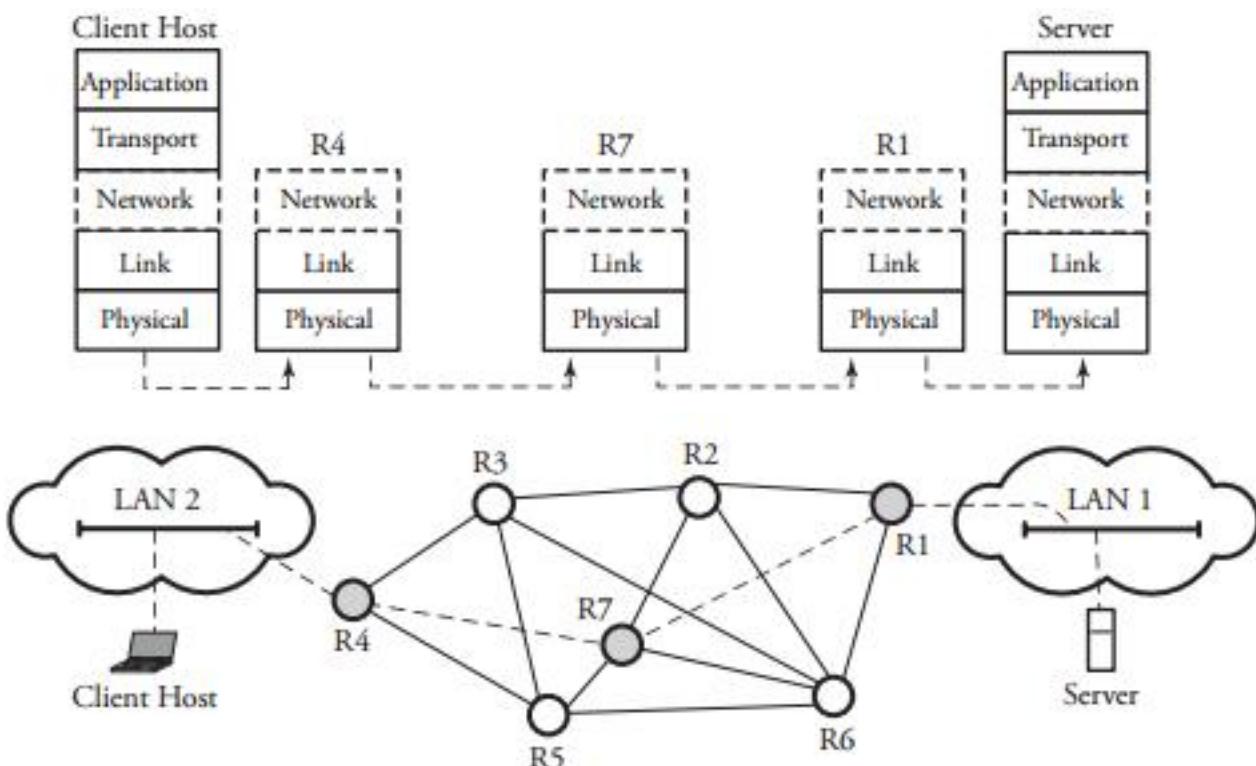


Figure 7.1 End-to-end communication between a client host and a server at the network layer

Routing algorithms can be differentiated on several key characteristics:

- *Accuracy.* An algorithm must operate correctly so that it can find the destination in an appropriate amount of time.
- *Simplicity.* Low complexity of algorithms is particularly important where routers with limited physical resources involve software.
- *Optimality.* This refers to the ability of the routing algorithm to select the best route.
- *Stability.* Routing algorithms must perform correctly in the face of unforeseen circumstances, such as node failure and routing table corruptions.
- *Adaptability.* When a failure happens in a network, an algorithm should be able to adapt load increases or decreases.
- *Convergence.* Routing algorithms must converge rapidly when a network distributes routing update messages.
- *Load balancing.* A good routing algorithm balances over eligible links to avoid having a heavily and temporarily congested link.

In practice, this list is used to determine how efficiently a route is selected. The first factor to account for determining the best path to a destination is the volume of traffic ahead.

7.1.1 Assigning Addresses to Hosts and Routers, and DHCP

Suppose that LAN 1 in Figure 7.1 has an Internet service provider denoted by ISP 1 and that LAN 2 has an ISP denoted by ISP 2. Assume that ISP 1 handles the LAN 1 networking tasks of two independent local departments with IP address blocks 205.101.8.0/23 and 205.101.10.0/23, respectively. Similarly, ISP 2 handles the LAN 2 networking tasks of two independent local departments with IP address blocks 145.76.12.0/22 and 145.76.14.0/22, respectively. (Recall from Chapter 2 the CIDR IP addressing scheme being used here.)

In this scenario, ISP 1 clearly advertises to its outside domains through router R1 that it can process any datagram (IP packet) whose first 22 address bits exactly match 205.101.8.0/22. Note that the outside domains need not know the contents of these two address blocks, as the internal routing within each LAN is handled by an internal server. Similarly, ISP 2 advertises to its outside domains through router R4 that it can process any datagram whose first 21 address bits exactly match 145.76.12.0/22. If for

any reason ISP 1 and ISP 2 merge, the least costly solution to avoid changing the address blocks of ISPs is to keep all address blocks unchanged and to have both R1 and R2 advertise to their outside domains that each can process any datagram whose first 22 address bits exactly match 205.101.8.0/22 or whose first 21 address bits exactly match 145.76.12.0/22.

IP addresses are organized by the *Internet Corporation for Assigned Names and Numbers* (ICANN). An ISP can request a block of addresses from ICANN. Then, an organization can also request a block of addresses from its ISP. A block of addresses obtained from an ISP can be assigned over hosts, servers, and router interfaces by a network manager. Note that always a unique IP address must always be assigned to each host as client or server or router interface (input port or output port). However, under some circumstances, a globally unique IP address assignment can be avoided (Section 7.1.2).

Dynamic Host Configuration Protocol (DHCP)

A different method of assigning addresses to a host is called *Dynamic Host Configuration Protocol* (DHCP), whereby a host is allocated an IP address automatically. DHCP allows a host to learn its subnet mask, the address of its first-hop router, or even the address of other major local servers. Because this addressing automation lets a host learn several key pieces of information in a network, DHCP is sometimes called a *plug-and-play* protocol, whereby hosts can join or leave a network without requiring configuration by network managers.

The convenience of this method of address assignment gives DHCP multiple uses of IP addresses. If any ISP manager does not have a sufficient number of IP addresses, DHCP is used to assign each of its connecting hosts a temporary IP address. When a host joins or leaves, the management server must update its list of available IP addresses. If a host joins the network, the server assigns an available IP address; each time a host leaves, its address is included in the pool of available addresses. DHCP is especially useful in *mobile* IP, with mobile hosts joining and leaving an ISP frequently.

7.1.2 Network Address Translation (NAT)

Any IP-type device requires a unique IP address. Because of the growing number of Internet users and devices, each requiring a unique IP address, the issue is critical, especially when a new LAN needs to be added to a community network despite a limited number of IP addresses available for that community. Even in very small residential networks, the issue arises when new devices are added to the network. Although the

numbers of users, servers, or subnets expands over time, the total allocated IP addresses are still limited to a certain level by the associated ISP.

Besides the popularity of IPv6, explained in Chapter 2, an alternative solution, called *network address translation* (NAT), can be used to overcome the challenge presented by this issue.

The idea behind NAT is that all the users and hosts of a private network do not need to have a globally unique addresses. Instead, they can be assigned private and unique addresses within their own private networks, and a NAT-enabled router that connects the private network to the outside world can translate these addresses to globally unique addresses. The NAT-enabled router hides from the outside world the details of the private network. The router acts as a single networking device with a single IP address to the outside world.

For example, assume that a private network with a NAT-enabled router is connecting to the outside world. Suppose that all the machines in this network are internally assigned 128.0.0.0/9 and that the output port of the router is assigned an IP address of 197.36.32.4. Now, this network has the advantage that additional machines or even LANs can be added to it and that each can use an address in the block 128.0.0.0/9. Therefore, users and servers within the network can easily use 128.0.0.0/9 addressing to transmit packets to each other, but packets forwarded beyond the network into the Internet clearly do not use these addresses. This way, thousands of other networks can use the same block of addresses internally. Given a datagram received by the NAT-enabled router, we now need to know how the router knows to which internal host it should deliver the datagram. The answer lies in the use of a *port number* and a *NAT translation table* in the router.

Example. Assume that a host with internal address 128.0.0.1 and port number 4527 in a private network requests a connection to a server with IP address 144.55.34.2 and arbitrary port number 3843, which resides in a different country. Suppose that the output port of the connecting NAT router is assigned IP address 197.36.32.4.

Solution. To set up this connection, the host sends its request with source address 128.0.0.1,4527 to the NAT router. The router “translates” this address in its NAT routing table by changing the arbitrary port number from 4527 to an official one of 5557 and changing the internal IP address 128.0.0.1 to its own port IP address, 197.36.32.4. The router then makes the connection request to site 144.55.34.2,3843, using address 197.36.32.4,5557. When the router receives the response from the remote site, the router does the reverse translation and delivers the response to host 128.0.0.1. Although NAT protocol solves the shortage of IP addresses in small communities, it has a

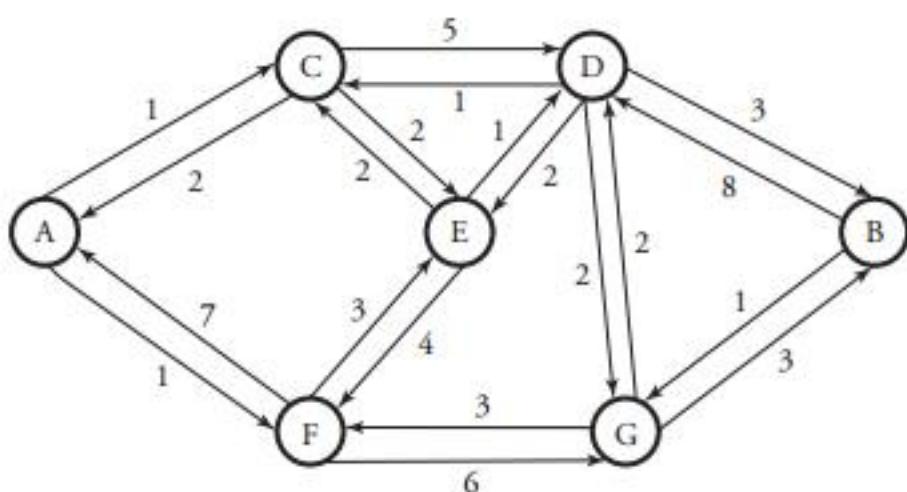


Figure 7.2 A packet-switched network and all link costs

major drawback of avoiding the assignment of a unique IP address to every networking component.

7.1.3 Route Cost

A packet-switched network consists of nodes—routers or switches—that are connected by links. A *link cost* between a pair of source and destination nodes refers to the number of packets currently waiting ahead in the destination node. The goal is to choose a routing based on the *minimum number of hops*, or *least-cost path*. For example, Figure 7.2 shows a network in which the lines between each two nodes represent a link and its cost in its corresponding direction.

The *least-cost path* between each pair of nodes is the minimum cost of the routing between the two nodes, taking into account all possible links between the two nodes. For example, the least-cost path in Figure 7.2 between nodes A and D is not A-C-D. The calculated cost of this path is $1 + 5 = 6$. This is true because the better path is A-C-E-D, with a calculated cost of $1 + 2 + 1 = 4$. Thus, the second case has more paths, but it has a lower cost than the first case, which has a shorter path but a higher cost.

7.1.4 Classification of Routing Algorithms

Routing algorithms can be classified in several ways. One way is to classify them as either *least-cost path*, whereby the lowest-cost path must be determined for routing, or *non-least-cost path*, whereby the determination of a route is not based on the cost of a path.

Another way is based on whether an algorithm is *distributed* or *centralized*. In distributed routing, all nodes contribute in making the routing decision for each packet.

In other words, an algorithm in distributed routing allows a node to gain information from all the nodes, but the least-cost path is determined locally. In centralized routing, only a designated node can make a decision. This central node uses the information gained from all nodes, but if the central node fails, the routing function in the network may be interrupted. A special case of centralized routing is *source routing*, whereby the routing decision is made only by the source server rather than by a network node. The outcome is then communicated to other nodes.

Routing can also be classified as either *static* or *dynamic*. In static routing, a network establishes an initial topology of paths. Addresses of initial paths are loaded onto routing tables at each node for a certain period of time. The main disadvantage of static routing is that the size of the network has to be small enough to be controllable. Also, if a failure happens in the network, it is unable to react immediately. In dynamic routing, the state of the network is learned through the communication of each router with its neighbors. Thus, the state of each region in the network is propagated throughout the network after all nodes finally update their routing tables. Each router can find the best path to a destination by receiving updated information from surrounding nodes.

7.2 Least-Cost-Path Algorithms

In practice, the majority of Internet routing methods are based on least-cost algorithms. In such algorithms, a link cost is proportional to the link's current traffic load. However, the link cost may not always be proportional to the current load. The link cost is defined on both directions between each pair of nodes. Several least-cost-path algorithms have been developed for packet-switched networks. In particular, *Dijkstra's algorithm* and the *Bellman-Ford algorithm* are the most effective and widely used algorithms.

7.2.1 Dijkstra's Algorithm

Dijkstra's algorithm is a centralized routing algorithm that maintains information in a central location. The objective is to find the least-cost path from a given source node to all other nodes. This algorithm determines least-cost paths from a source node to a destination node by optimizing the cost in multiple iterations. Dijkstra's algorithm is as follows:

Begin Dijkstra's Algorithm

1. Define:

s = Source node

k = Set of visited nodes by the algorithm

α_{ij} = Cost of the link from node i to node j

β_{ij} = Cost of the least-cost path from node i to node j

2. Initialize:

$k = \{s\}$

$\beta_{sj} = \alpha_{sj}$ for $j \neq s$

3. Next node:

Find $x \notin k$ that $\beta_{sx} = \min \beta_{sj}$ for $j \notin k$.

Add x to k .

4. Least-cost paths:

$\beta_{sj} = \min(\beta_{sj}, \beta_{sx} + \alpha_{xj})$ for $j \notin k$ ■

If any two nodes i and j are not connected directly, the cost for that link is infinity, indicated by $\beta_{ij} = \infty$. Steps 2 and 3 are repeated until paths are assigned to all nodes. At step 1, k represents s , and β_{sj} computes the cost of the least-cost path from s to node j . At step 2, we want to find x among the neighboring nodes but not in k such that the cost is minimized. At step 3, we simply update the least-cost path. The algorithm ends when all nodes have been visited and included in the algorithm.

Example. Using Dijkstra's algorithm, find the least-cost path from node A to node B in Figure 7.2.

Solution. The detailed operation is shown in the Table 7.1. The first step is to find a path from the source node A to all other nodes. Thus, at the first row, $k = \{A\}$. It is obvious there are direct links from A to nodes C and F. Therefore, the least-cost path for either node is 1, as shown in Figure 7.2. We then fill the table with AC(1) and AF(1), respectively. Given $k = \{A\}$, there is no connections between A and nodes D, E, G, and B. The algorithm continues until all nodes have been included as $k = \{A, C, F, E, D, G, B\}$, and we obtain the least-cost path of ACEDB(7).

7.2.2 Bellman-Ford Algorithm

The *Bellman-Ford algorithm* finds the least-cost path from a source to a destination by passing through no more than ℓ links. The essence of the algorithm consists of the following steps

Begin Bellman-Ford Algorithm

1. Define:

s = Source node

α_{ij} = Cost of the link from node i to node j

$\beta_{ij}(\ell)$ = Cost of the least-cost path from i to j with no more than ℓ links

Table 7.1 Use of Dijkstra's algorithm

k	β_{AC}	β_{AF}	β_{AE}	β_{AD}	β_{AG}	β_{AB}
{A}	AC(1)	AF(1)	×	×	×	×
{A,C}	AC(1)	AF(1)	ACE(3)	ACD(6)	×	×
{A,C,F}	AC(1)	AF(1)	ACE(3)	ACD(6)	AFG(7)	×
{A,C,F,E}	AC(1)	AF(1)	ACE(3)	ACED(4)	AFG(7)	×
{A,C,F,E,D}	AC(1)	AF(1)	ACE(3)	ACED(4)	ACEDG(6)	ACEDB(7)
{A,C,F,E,D,G}	AC(1)	AF(1)	ACE(3)	ACED(4)	ACEDG(6)	ACEDB(7)
{A,C,F,E,D,G,B}	AC(1)	AF(1)	ACE(3)	ACED(4)	ACEDG(6)	ACEDB(7)

2. Initialize:

$$\beta_{sj}(0) = \infty, \text{ for all } j \neq s$$

$$\beta_{ss}(\ell) = 0, \text{ for all } \ell$$

3. Least-cost path:

for any node $j \neq s$ with predecessor node i :

$$\beta_{sj}(\ell + 1) = \min_i [\beta_{si}(\ell) + \alpha_{ij}] \blacksquare$$

If any two nodes i and j are not connected directly, $\beta_{ij}(\ell) = \infty$. At step 2, every value of β is initialized. At step 3, we increase the number of links ℓ in a sequence of iterations. During each iteration, we find the least-cost path, given the value of ℓ . The algorithm ends when all nodes have been visited and included in the algorithm.

Example. Use the Bellman-Ford algorithm to find the least-cost path from node A to node B in Figure 7.2.

Solution. Table 7.2 shows the details of least-cost-path iterations. For iteration $\ell = 1$, only AC with cost 1 and AF with cost 1 exist, owing to the restriction enforced by $\ell = 1$. This trend changes at iteration $\ell = 2$, when ACE with cost 3 can be added to the set of least-cost paths. As seen, the result of the final least-cost path is identical to the one obtained by Dijkstra's algorithm.

We can now compare these two algorithms. In step 2 of Bellman-Ford, the calculation of the link cost to any node j requires knowledge of the link cost to all neighboring nodes. In step 3 of Dijkstra's algorithm, each node requires knowledge of the network topology at each time of iteration. The performance of each algorithm varies network to network and depends on the topology and size of a particular network. The comparison of these two algorithms, therefore, depends on the speed of each to achieve its

Table 7.2 Use of the Bellman-Ford algorithm

ℓ	β_{AC}	β_{AF}	β_{AE}	β_{AD}	β_{AG}	β_{AB}
0	x	x	x	x	x	x
1	AC(1)	AF(1)	x	x	x	x
2	AC(1)	AF(1)	ACE(3)	ACD(6)	AFG(7)	x
3	AC(1)	AF(1)	ACE(3)	AED(4)	AFG(7)	ACDB(9)
4	AC(1)	AF(1)	ACE(3)	AED(4)	ACEG(6)	ACEDB(7)

objective at its corresponding step given a network under routing. A simulation can clarify the efficiency of each algorithm given a certain network topology.

7.3 Non-Least-Cost-Path Routing

Our networking infrastructure deploys a variety of procedures, algorithms, and protocols for routing packets, depending on the applications, their significance, and the budget for building a network. Besides the least-cost-path algorithms that effectively identify the best possible paths, some nonoptimal algorithms can be used for applications that may not need complex and expensive routing protocols. Two examples of such *non-least-cost routing algorithms* are *flood routing* and *deflection routing*.

7.3.1 Flood Routing

Flood routing is a very simple routing strategy involving less hardware setup. The essence of this routing method is that a packet received from a node is copied and transmitted on all outgoing links of that node except for the link that the packet arrived from. After the first transmission, all the routers within one hop receive the packet. After the second transmission, all the routers within two hops receive the packet, and so on. Unless a mechanism stops the transmission, the process continues; as a result, the volume of traffic increases with time, as shown in Figure 7.3.

In this figure, three packets arrive at node A from a source. The first packet is copied to both nodes B and C. At nodes B and C, the copies of the packet are copied to their neighboring nodes. This method has the deficiency of packet reflection: a node can receive an unwanted copy of a packet. Although this problem can be fixed by dropping unwanted packets at any time, the network does not function effectively, owing to increased unnecessary traffic. One way to prevent packet duplication is to set

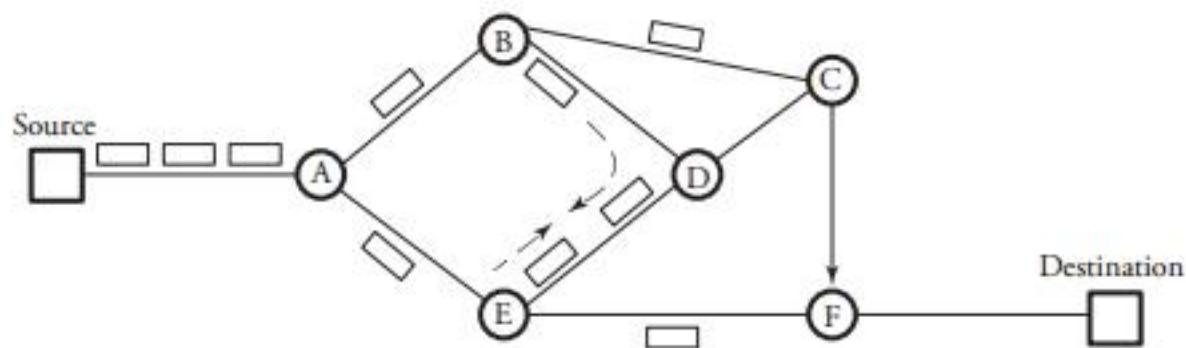


Figure 7.3 Flood routing

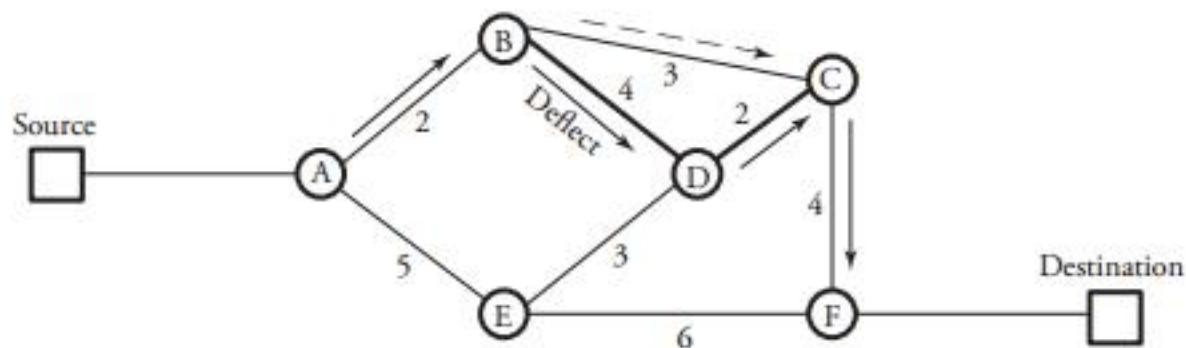


Figure 7.4 Deflection routing

up memory for each node to remember the identity of those packets that have already been retransmitted for blocking them.

7.3.2 Deflection Routing

In *deflection routing*, or *hot-potato routing*, a packet is sent to a destination determined at each router. At each step of the routing, a packet is examined with respect to its destination. If the requested link is free, the packet is sent on that link; otherwise, the packet is *deflected* onto another link, selected at random. The deflected packet is given an increment on its priority field. This increment gives the packet a better chance to win the contention with others in future contentions. For the deflected packet, if a new contention occurs with another packet in its next hop, the packet with the higher priority gets the desired link, and the other one is deflected with, of course, an increment in its priority field. In Figure 7.4, a packet is deflected at node B but eventually gets to node C, with a total of one additional hop and a total cost difference of 3, using node B.

7.4 Intradomain Routing Protocols

There are two classes of routing protocols: *intradomain routing protocol* or *intranetwork routing protocol* or *intranet*, and *interdomain routing protocol* or *internetwork routing protocol* or *extranet*. An intradomain routing protocol routes packets within a defined domain, such as for routing e-mail or Web browsing within an institutional network. By contrast, an interdomain routing protocol is a procedure for routing packets on networks of domains and is discussed in Section 7.5.

Figure 7.5 shows intradomain routing; each point-to-point link connects an associated pair of routers and indicates the corresponding cost of the connection. A *host* is an end system that can be directly connected to the router. A *cost* is considered with the output side of each router interface. When nothing is written on an arrow, no cost is associated with that connection. An arrow that connects a network and a router has a zero cost. A database is gathered from each router, including costs of paths in all separate directions. In the figure, two hosts within a domain are exchanging data facing their own LANs (N1 and N6), several other LANs (N2 through N5), and several routers (R1 through R8).

The most widely used intranetworking routing protocols are the two unicast routing protocols RIP and OSPF.

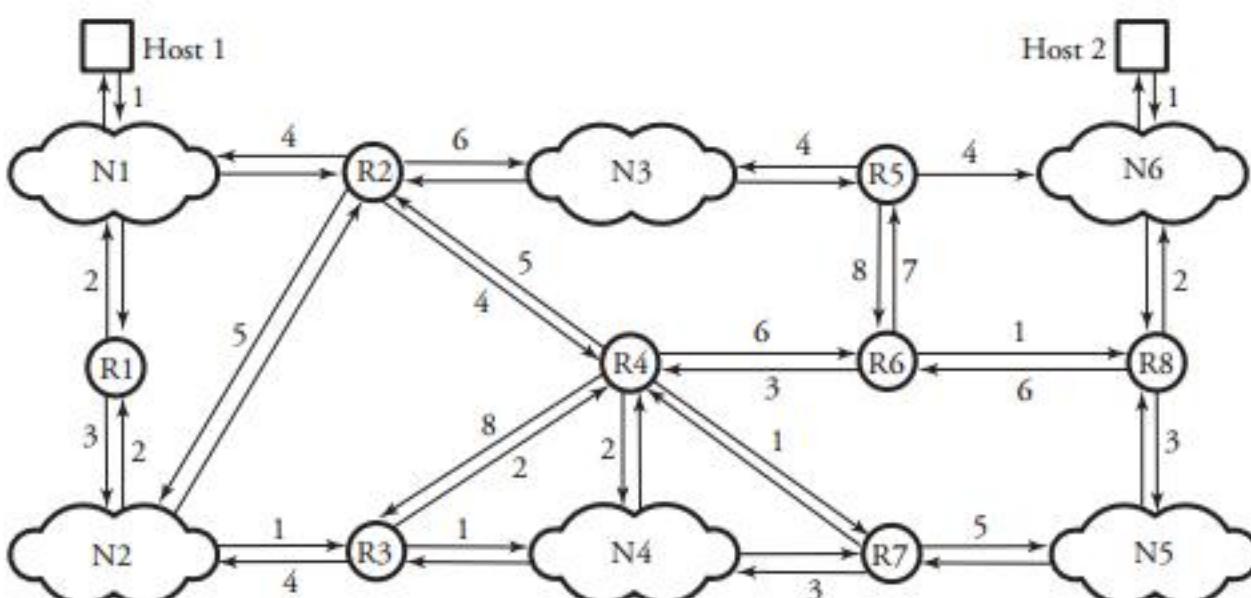


Figure 7.5 Intradomain routing, with two hosts in a large domain exchanging data facing several LANs and routers

Table 7.3 A routing table in router R1 in Figure 7.5a, using RIP

Destination Network	Next Router	Path	Cost
N1	—	R1-N1	2
N2	—	R1-N2	3
N3	R2	R1-N2-R2-N3	8
N4	R3	R1-N2-R3-N4	4
N5	R3	R1-N2-R3-N4-R7-N5	9
N6	R2	R1-N2-R3-N4-R7-N5-R8-N6	11

7.4.1 Routing Information Protocol (RIP)

The *routing information protocol* (RIP) is a simple intradomain routing protocol. RIP is one of the most widely used routing protocols in the Internet infrastructure but is also appropriate for routing in smaller domains. In RIP, routers exchange information about reachable networks and the number of hops and associated costs required to reach a destination. This protocol is summarized as follows.

Begin RIP Algorithm

1. Apply the Bellman-Ford algorithm in a distributed fashion, including all hosts and routers.
2. For each router, form an optimum vector of distance indicating the cost of routing and other necessary parameters, using *distance-vector routing* (discussed next).
3. If a cost of routing at any point changes, propagate the change periodically to the neighboring routers and hosts of that point.
4. Update tables of routers and hosts periodically. ■

Table 7.3 shows routing information maintained in router R1 of the network shown in Figure 7.5. For example, at the third row of the routing table, the path to N3 is initiated through router R2 and established through path R1-N2-R2-N3, with a total cost of 8. Because routers and hosts within the network use vectors of data, a technique called *distance vector routing* is used.

Distance Vector Routing

The *distance vector algorithm* was designed mainly for small network topologies. The term *distance vector* derives from the fact that the protocol includes its routing updates

with a vector of distances, or hop counts. In distance vector routing, all nodes exchange information only with their neighboring nodes. Nodes participating in the same local network are considered neighboring nodes. In this protocol, each individual node i maintains three vectors:

$$\begin{aligned}\mathbf{B}_i &= [b_{i,1}, \dots, b_{i,n}] = \text{link-cost vector} \\ \mathbf{D}_i &= [d_{i,1}, \dots, d_{i,m}] = \text{distance vector} \\ \mathbf{H}_i &= [h_{i,1}, \dots, h_{i,m}] = \text{next-hop vector}\end{aligned}$$

For the calculation of the link-cost vector, we consider node i to be directly connected to n networks out of the total of m existing networks. The link-cost vector \mathbf{B}_i is a vector containing the costs of node i to its directly connected network. For example, $b_{i,1}$ refers to the cost of node i to network 1. Distance vector \mathbf{D}_i contains the estimated minimum delays from node i to all networks within its domain. For example, $d_{i,1}$ is the estimated minimum delay from node i to network 1. Finally, the next-hop vector \mathbf{H}_i is a matrix vector containing the next node in the minimum-delay path from node i to its directly connected network. As an example for this vector, $h_{i,1}$ refers to the next node in the minimum-delay path from node i to Network 1. Each node exchanges its distance every r seconds with all its neighbors. Node i is responsible for updating both of its vectors by:

$$d_{i,j} = \min_{k \in \{1, \dots, n\}} [d_{k,j} + b_{i,g(x,y)}] \quad (7.1)$$

and

$$h_{i,j} = k, \quad (7.2)$$

where $\{1, \dots, n\}$ is the set of network nodes for node i , and $g(x, y)$ is a network that connects node x to node y ; thus, $b_{i,g(x,y)}$ is the cost of node i to network $g(x, y)$. Equations (7.1) and (7.2) are a distributed version of the Bellman-Ford algorithm, which is used in RIP. Each router i begins with $d_{i,j} = b_{i,j}$ if it is directly connected to network j . All routers concurrently replace the distance vectors and compute Equation (7.1). Routers repeat this process again. Each iteration is equal to one iteration of step 2 in the Bellman-Ford algorithm, processed in parallel at each node of the graph. The shortest-length paths with a distance of at most one hop are considered after the first

iteration. Then, the second iteration takes place, with the least-cost paths with at most two hops; this continues until all least-cost paths have been discovered.

Updating Routing Tables in RIP

Since RIP depends on distance vector routing, each router transmits its distance-vector to its neighbors. Normally, updates are sent as replies whether or not requested. When a router broadcasts an RIP request packet, each router in the corresponding domain receiving the request immediately transmits a reply. A node within a short window of time receives distance-vectors from all its neighbors, and the total update occurs, based on incoming vectors. But this process is not practical, since the algorithm is asynchronous, implying that updates may not be received within any specified window of time.

RIP packets are sent typically in UDP fashion, so packet loss is always possible. In such environments, RIP is used to update routing tables after processing each distance vector. To update a routing table, if an incoming distance vector contains a new destination network, this information is added to the routing table. A node receiving a route with a smaller delay immediately replaces the previous route. Under certain conditions, such as after a router reset, the router can receive all the entries of its next hop to reproduce its new table.

The RIP *split-horizon* rule asserts that it is not practical to send information about a route back in the direction from which it was received. The split-horizon advantage is increased speed and removal of incorrect route within a timeout. The RIP *poisoned-reverse* rule has a faster response and bigger message size. Despite the original split horizon, a node sends updates to neighbors with a hop count of 16 for routing information arrived from those neighbors.

RIP Packet Format

Figure 7.6 shows the packet format of an RIP header. Each packet consists of several address distances. The header format contains the following specifications for the first address distance.

- *Command* indicates a request with value 1 or a reply with value 2.
- *Version number* specifies the version: RIP-1 or RIP-2.
- *Address family identifier* shows the type of address, such as an IP address.
- *IP address* provides the IP address in a particular network.
- *Metric* identifies the distance from a router to a specified network.

Byte:

	1	1	2	2	2	4	4	4	4
Command			Address Family Identifier			IP Address			Metric
Version Number	0			0			0	0	

Figure 7.6 Routing Information Protocol (RIP) header

Table 7.4 Current routing table with updated values for Host 1 in Figure 7.5, using RIP

Destination Network	Next Router	Cost	Updated Router	Next Router	Updated Cost
N1	—	1	—		1
N2	R1	4	R2		2
N3	R2	7	R2		7
N4	R1	8	R1		8
N5	R2	11	R2		11
N6	R2	13	R2		9

If a link cost is set to 1, the *metric* field acts as a hop count. But if link costs have larger values, the number of hops becomes smaller. Each packet consists of several address distances. If more than one address distance is needed, the header shown in Figure 7.6 can be extended to as many address distances as are requested, except for the *command* and *version number* and its 2 bytes of 0s; the remaining field structure can be repeated for each address distance.

Example. Apply RIP for connecting Host 1 to all six networks in Figure 7.5, given that at a certain time, the cost of R2-N2 changes from 5 to 1, and the cost of R4-R6 changes from 6 to 2.

Solution. The current routing table for the host is shown in Table 7.4. In particular, note the status of the table on destination network N2, where the next router is R1 with path Host1-N1-R1-N2 and a cost of 4. Changes on the link costs affect the status of the N2 and N6 entries. The new updates are reflected in Table 7.4, where the new

path becomes Host1-R2-N2, with a new, lower cost of 2. The status of N6 also changes to a new path, Host1-R2-R4-R6-R8-N6, with a new, lower cost of 9.

Issues and Limitations of RIP

One of the disadvantages of RIP is its slow convergence in response to a change in topology. *Convergence* refers to the point in time at which the entire network becomes updated. In large networks, a routing table exchanged between routers becomes very large and difficult to maintain, which may lead to an even slower convergence. Also, RIP might lead to suboptimal routes, since its decision is based on hop counts. Thus, low-speed links are treated equally or sometimes preferred over high-speed links. Another issue with RIP is the *count-to-infinity* restriction. Distance vector protocols have a limit on the number of hops after which a route is considered inaccessible. This restriction would cause issues for large networks.

Other issues with RIP stem from the distance vector algorithm. First, reliance on hop counts is one deficiency, as is the fact that routers exchange all the network values via periodic broadcasts of the entire routing table. Also, the distance vector algorithm may cause loops and delays, since they are based on periodic updates. For example, a route may go into a *hold* state if the corresponding update information is not received in a certain amount of time. This situation can translate into a significant amount of delay in update convergence in the network before the network discovers that route information has been lost.

Another major deficiency is the lack of support for variable-length subnet masks. RIP does not exchange mask information when it sends routing updates. A router receiving a routing update uses its locally defined subnet mask for the update, which would lead to complications and misinterpretation in a variably subnetted network. Distance vector networks do not have hierarchies, which makes them incapable of integrating to larger networks.

7.4.2 Open Shortest Path First (OSPF)

The limitations of RIP make it a nonpreferred routing strategy as networks scale up. The *Open Shortest Path First* (OSPF) protocol is a better choice of intradomain routing protocols, especially for TCP/IP applications. OSPF is based on Dijkstra's algorithm, using a tree that describes the network topology to define the shortest path from each router to each destination address. Since it keeps track of all paths of a route, OSPF has more overhead than RIP, but provides more stability and useful options. The essence of this protocol is as follows.

Begin OSPF Protocol

1. Apply Dijkstra's algorithm or another efficient algorithm in a distributed fashion for each host and router.
2. In each host or router, calculate the least-cost path, and propagate it to all nodes, using *link-state routing*.
3. Periodically propagate any changes in routing cost to all routers and hosts.
4. Update the routing tables of routers and hosts. ■

Before focusing on the details of OSPF, we need to describe the essence of the *link-state routing* protocol.

Link-State Routing

In the *link-state routing*, routers collaborate by exchanging packets carrying status information about their adjacent links. A router collects all the packets and determines the network topology, thereby executing its own shortest-route algorithm.

As explained in the section on RIP, with distance vector routing, each router must send a distance vector to all its neighbors. If a link cost changes, it may take a significant amount of time to spread the change throughout the network. *Link-state routing* is designed to resolve this issue; each router sends routing information to all routers, not only to the neighbors. That way, the transmitting router discovers link-cost changes, so a new link cost is formed. Since each router receives all link costs from all routers, it is able to calculate the least-cost path to each destination of the network. The router can use any efficient routing algorithm, such as Dijkstra's algorithm, to find the shortest path.

The core function of the link-state algorithm is flood routing, which requires no network topology information. Let's review the three important properties of flooding used in link-state routing. First, a packet always traverses completely between a source and a destination, given at least one path between these two nodes. This property makes this routing technique robust. Second, at least one copy of the packet arriving at the destination must possess the minimum delay, since all routers in the network are tried. This property makes the flooded information propagate to all routers very quickly. Third, all nodes in the network are visited whether they are directly or indirectly connected to the source node. This property makes every router receive all the information needed to update its routing table.

Details of OSPF Operation

To return to the OSPF operation: Every router using OSPF is aware of its local link-cost status and periodically sends updates to all routers. After receiving update packets, each

Byte:

	1	1	2	4	4	2	2	8	
Version Number	Type	Packet Length	Router ID	Area ID	Checksum	Authentication Type	Authentication		Data

Figure 7.7 OSPF header format

router is responsible for informing its sending router of receipt of the update. These communications, though, result in additional traffic, potentially leading to congestion. OSPF can provide a flexible link-cost rule based on type of service (TOS). The TOS information allows OSPF to select different routes for IP packets, based on the value of the TOS field. Therefore, instead of assigning a cost to a link, it is possible to assign different values of cost to a link, depending on the TOS the value for each block of data.

TOS has five levels of values: *level 1* TOS, with the highest value to act by default, through *level 5* TOS, with the lowest value for minimized delay. By looking at these five categories, it is obvious that a router can build up to five routing tables for each TOS. For example, if a link looks good for delay-sensitive traffic, a router gives it a low, level 5 TOS value for low delay and a high, level 2 TOS value for all other traffic types. Thus, OSPF selects a different shortest path if the incoming packet is normal and does not require an optimal route.

OSPF Packet Format

An IP packet that contains OSPF has a standard broadcast IP address of 224.0.0.5 for flood routing. All OSPF packets use a 24-byte header as follows:

- *Version number* indicates the version of OSPF.
- *Type* is one of the five types of packets for OSPF to choose from: *hello*, *database description*, *link-state request*, *link-state update*, and *link-state acknowledgment*.
- *Packet length* specifies the length of the OSPF packet.
- *Router ID* specifies the packet's source router ID.
- *Area ID* refers to the area that the source router belongs to.
- *Checksum* specifies the standard IP checksum of the packet contents (see Chapter 4).

- *Authentication type* identifies which authentication method to choose (see Section 10.5).
- *Authentication* specifies the authentication method (see Section 10.5).

The *hello* packet specified in the *Type* field is used to detect each router's active neighbors. Each router periodically sends out *hello* packets to its neighbors to discover its active neighboring routers. Each packet contains the identity of the neighboring router interface taken from the *hello* packet already received from it. The *database description* packets are used for database structure exchange between two adjacent routers to synchronize their knowledge of network topology. The *link-state request* packet is transmitted to request a specific portion of link-state database from a neighboring router. The *link-state update* packet transfers the link-state information to all neighboring routers. Finally, the *link-state acknowledgment* packet acknowledges the update of a link-state packet.

In summary, when a router is turned on, it transmits *hello* packets to all neighboring routers and then establishes routing connections by synchronizing databases. Periodically, each router sends a link-state update message describing its routing database to all other routers. Therefore, all routers have the same description of the topology of the local network. Every router calculates a shortest-path tree that describes the shortest path to each destination address, thereby indicating the closest router for communications.

7.5 Interdomain Routing Protocols

Unlike intradomain routing protocols, *interdomain routing protocols* create a network of networks, or an *internetwork*. Interdomain routing protocols route packets outside of a defined domain. Each domain consists of several networks and routers that can be accessed publicly. Figure 7.8 shows an example of internetworking, in which a packet at router R₁ faces two defined larger service provider domains, A and B. Each service domain is shown by a number of interconnected circles, each representing a network or a router. Obviously, at router R₁, finding the best path for a packet to go to which domain is a challenge. The Border Gateway Protocol helps meet this challenge.

7.5.1 Border Gateway Protocol (BGP)

The *Border gateway protocol* (BGP) is a preferred routing protocol for interdomain communications and TCP/IP connections. BGP allows routers to carry specific policies or constraints that they must meet. With BGP, routers exchange more comprehensive information about routes to a certain destination instead of simply costs and the best

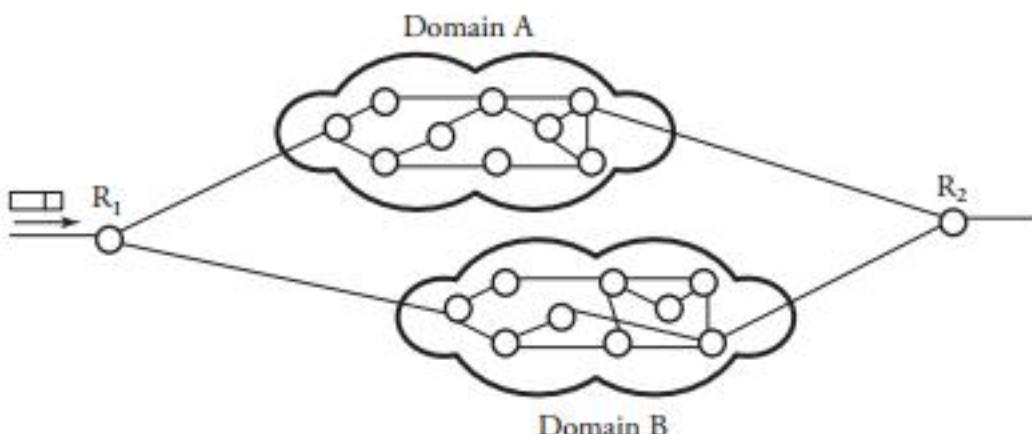


Figure 7.8 Internetworking, in which router packet faces two defined service domains

link. In BGP, two contributing routers can exchange routing information even if they are located in two different autonomous systems. When an external destination is chosen, a router sends the information to all internal neighbors. Then, all routers decide whether the new route is possible, and if so, the new route is added to the router's database. Thus, the new update message is propagated. One of the most important techniques in BGP is path-vector routing.

Path-Vector Routing

RIP and OSPF are not suitable for interdomain routing protocols. As discussed earlier, distance vector routing is used to send information to each of a router's neighbors, and then each router builds up a routing database. However, a router is not aware of the identity of routers on any particular path. Two problems arise. First, if different routers give different information to an assigned cost, it is impossible to have stable and loop-free routes. Second, an autonomous system can have limitations about which specific autonomous system should be used. This is true while the distance vector algorithm has no information about the autonomous systems.

Each router sends its link cost to all other routers and then starts routing calculations. Two issues can arise in link-state routing. First, different independent systems can use different costs and have different limitations. The link-state protocol allows a router to make the topology, and its metrics may be different for each independent system. In this case, it is impossible to create a reliable routing algorithm. Second, when flood routing occurs, the use of an interdomain routing protocol across the independent system can be unstable.

To resolve these issues, consider an alternative solution: the *path vector routing protocol*, which provides information about how to reach a network given a certain router and identifies which autonomous system (or domain) should be visited, as in the case for router R₁ in Figure 7.8. The path vector routing protocol is different from the distance vector algorithm, in which each path has information about cost and distance. In the path vector routing protocol, these packages of information are not included, and all visited autonomous systems and all components of domain A in Figure 7.8 reaching the destination network are listed in each routing information package. Thus, a router can be programmed to refuse the acceptance of a particular path if the information about the path is not included in the package it receives.

Details of BGP

BGP was created to find a solution to interdomain routing among autonomous (independent) systems. BGP works well for making a connection when a long-haul TCP session must be established. BGP has three functional components:

1. Neighbor relationship
2. Neighbor maintenance
3. Network maintenance

The neighbor relationship refers to an agreement between two routers in two different autonomous systems to exchange routing information on a regular basis. A router may reject its participation in establishing a neighbor relationship for several reasons, such as the rule of the domain, overload, or a temporary malfunctioning of external links. Neighbor maintenance is a process of maintaining the neighbor relationship already established. Normally, each corresponding router needs to find out whether the relationship with the other router is still available. For this reason, two routers send *keep-alive* messages to each other. The last BGP process is network maintenance. Each router keeps the database of the subnetworks that it can reach and tries to get the best route for that subnetwork.

BGP Packets

There are four different BGP packets as shown in Figure 7.9, as follows:

- *Open packet*. This packet requests establishment of a relationship between two routers.
- *Update packet*. This packet conveys update information about routes.

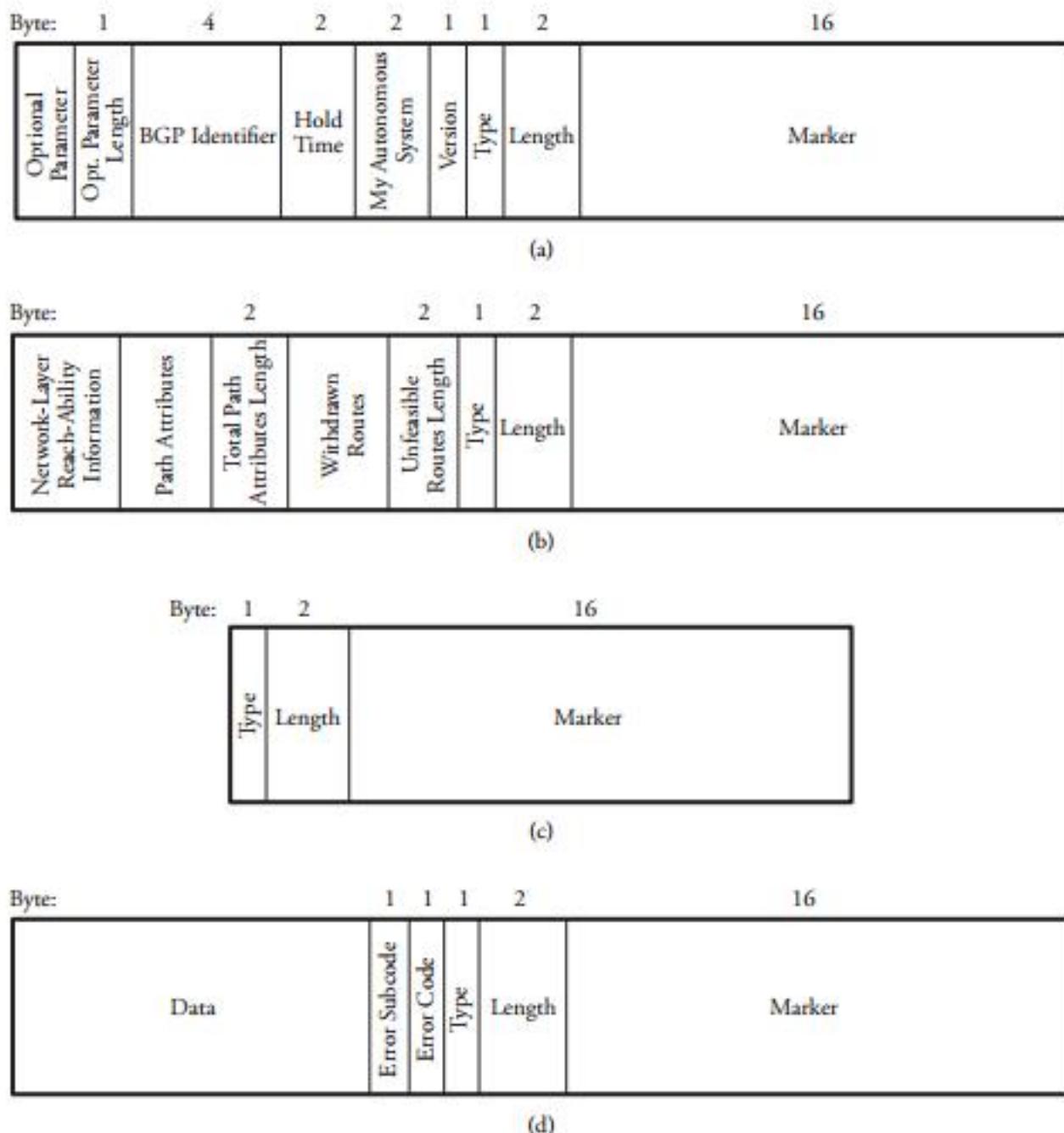


Figure 7.9 Four types of BGP packets and their fields: (a) open packet, (b) update packet, (c) keep-alive packet, and (d) notification packet

- *Keep-alive packet.* Once a relationship between two routers is established, this packet confirms its neighbor relationship frequently.
- *Notification packet.* This packet is used when an error occurs.

Figure 7.10 shows a BGP connection. Assume that router R₂ in one Internet service provider, network 2 opens a TCP connection to its desired router, R₃, in another ISP

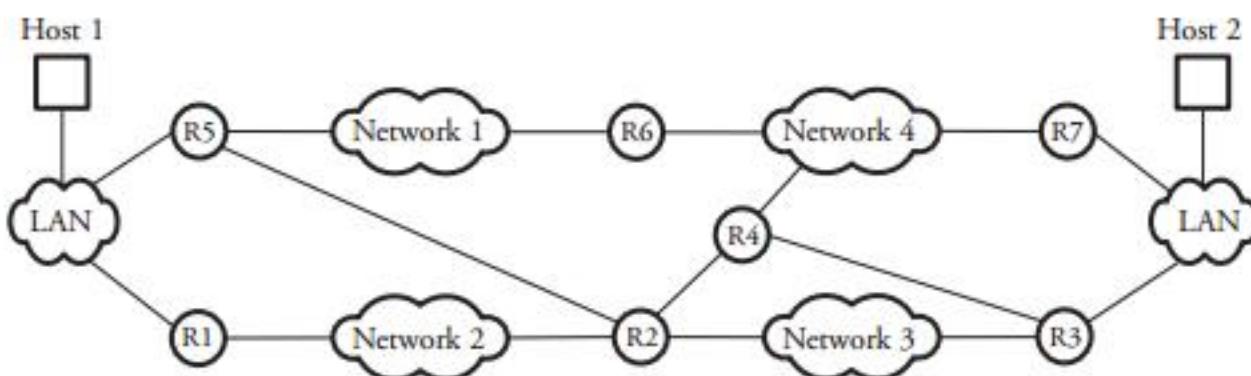


Figure 7.10 Use of BGP in an interdomain routing protocol

domain of network 3. Router R_2 sends an *open* packet to R_3 . This packet is identified by R_3 , telling it which domain the sender belongs to. A router sends a *keep-alive* packet to its neighbors to prevent an agreed hold time from expiring.

The first type of update packet can include the information of a single route that passes through a network. This information can be divided into three fields: the *network-layer readability* field, the *path attributes* field, and the *total path attributes length* field. The *network-layer readability* field has a list of subnetwork identifiers that can be found by the router. The *path attributes* field has a list of attributes that can be referred to a specific route. The second type of update information is used to remove one or more routes identified by the IP address of the destination subnetwork. In this case, the *notification* packet is sent in case of an error, which may be authentication errors, validity errors in the update packet, or an expired hold-time error.

7.6 Congestion Control at Network Layer

Congestion represents an overloaded condition in a network. Congestion control can be achieved by optimum usage of the available resources in the network. Figure 7.11 shows a set of performance graphs for networks in which three possible cases are compared: no congestion, moderate congestion, and severe congestion. These plots indicate that if a network has no congestion control, the consequence may be a severe performance degradation, even to the extent that the carried load starts to fall with increasing offered load. The ideal situation is the one with no loss of data, as a result of no congestion, as shown in the figure. Normally, a significant amount of engineering effort is required to design a network with no congestion.

Congestion can be either *logical* or *physical*, as shown in Figure 7.12. The queueing feature in two routers can create a logical bottleneck between user A and user B. Meanwhile, insufficient bandwidth—a resource shortage—on physical links between

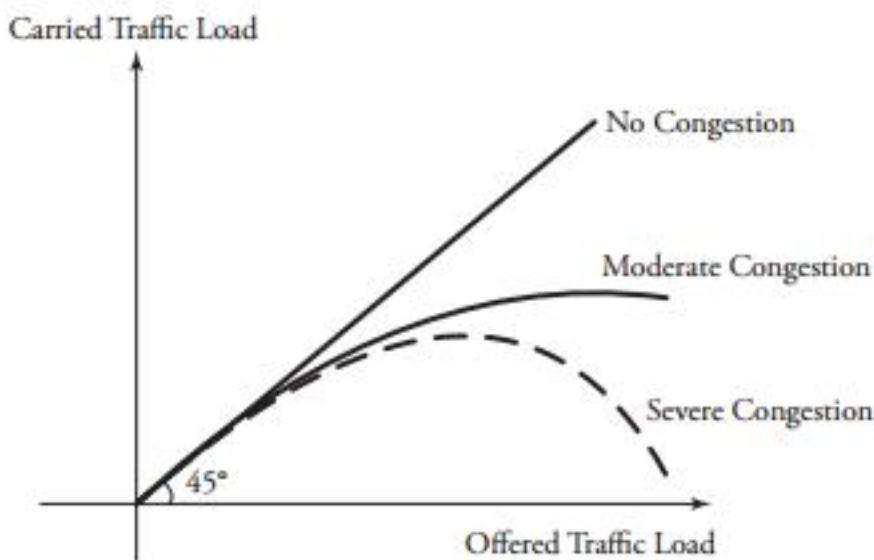


Figure 7.11 Comparison among networks in which congestion, moderate congestion, and severe congestion exist

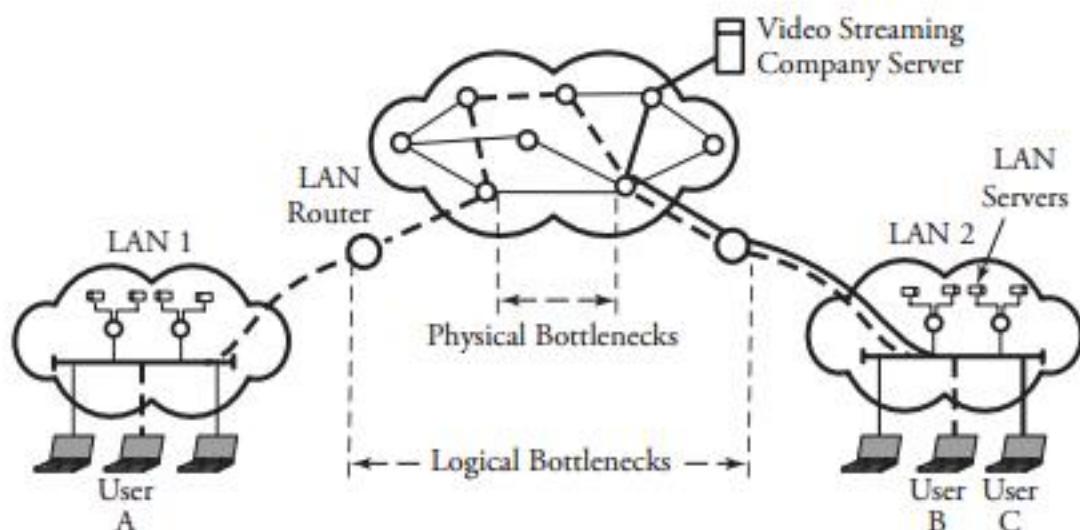


Figure 7.12 Forms of bottleneck along a round-trip path between two users

routers and the network can also be a bottleneck, resulting in congestion. Resource shortage can occur

- At the *link layer*, where the link bandwidth runs out
- At the *network layer*, where the queues of packets at nodes go out of control
- At the *transport layer*, where logical links between two routers within a communication session go out of control

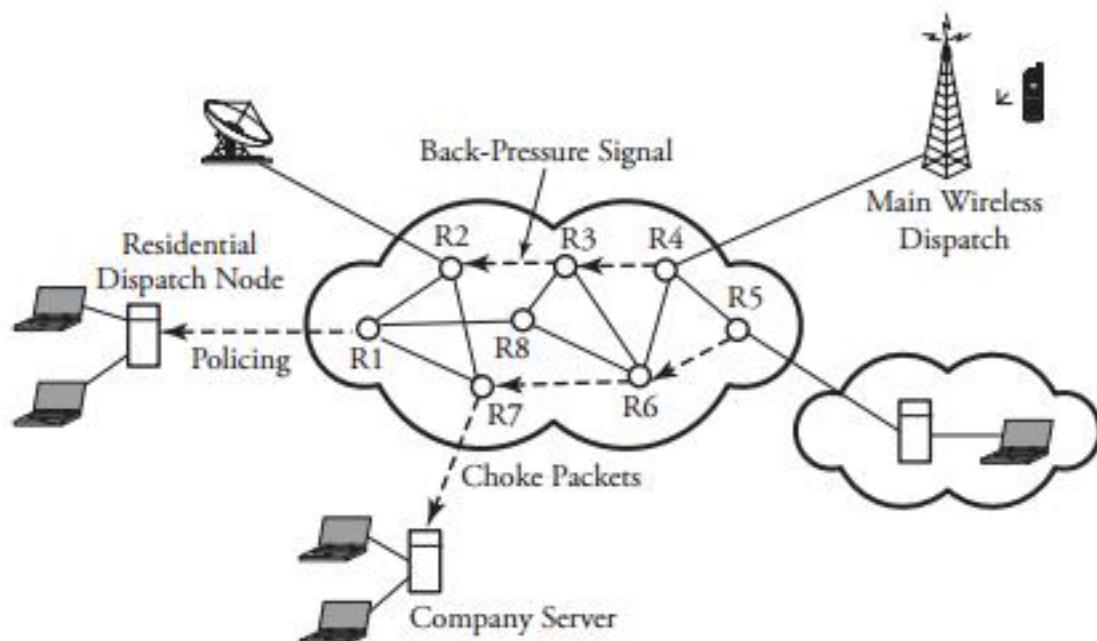


Figure 7.13 Unidirectional congestion control

One key way to avoid congestion is to carefully allocate network resources to users and applications. Network resources, such as bandwidth and buffer space, can be allocated to competing applications. Devising optimum and fair resource-allocation schemes can control congestion to some extent. In particular, congestion control applies to controlling data flow among a group of senders and receivers, whereas flow control is specifically related to the arrangement of traffic flows on links. Both resource allocation and congestion control are not limited to any single level of the protocol hierarchy. Resource allocation occurs at switches, routers, and end hosts. A router can send information on its available resources so that an end host can reserve resources at the router to use for various applications.

General methods of congestion control are either *unidirectional* or *bidirectional*. These two schemes are described next.

7.6.1 Unidirectional Congestion Control

A network can be controlled unidirectionally through *back-pressure signaling*, *transmission of choke packets*, and *traffic policing*. Figure 7.13 shows a network of eight routers: R1 through R8. These routers connect a variety of servicing companies: cellphones, satellite, residential, and company LANs. In such a configuration, congestion among routing nodes may occur at certain hours of the day.

The first type of congestion control is achieved by generating a *back-pressure signal* between two routers. The back-pressure scheme is similar to fluid flow in pipes. When one end of a pipe is closed, the pressure propagates backward to slow the flow of water at the source. The same concept can be applied to networks. When a node becomes congested, it slows down the traffic on its incoming links until the congestion is relieved. For example, in the figure, router R4 senses overloading traffic and consequently sends signals in the form of back-pressure packets to router R3 and thereby to router R2, which is assumed to be the source of overwhelming the path. Packet flow can be controlled on a hop-by-hop basis. Back-pressure signaling is propagated backward to each node along the path until the source node is reached. Accordingly, the source node restricts its packet flow, thereby reducing the congestion.

Choke-packet transmission is another solution to congestion. In this scheme, choke packets are sent to the source node by a congested node to restrict the flow of packets from the source node. A router or even an end host can send these packets when it is near full capacity, in anticipation of a condition leading to congestion at the router. The choke packets are sent periodically until congestion is relieved. On receipt of the choke packets, the source host reduces its traffic-generation rate until it stops receiving them.

The third method of congestion control is *policing* and is quite simple. An edge router, such as R1 in the figure, acts as a traffic police and directly monitors and controls its immediate connected consumers. In the figure, R1 is policing a residential dispatch node from which traffic from a certain residential area is flowing into the network.

7.6.2 Bidirectional Congestion Control

Figure 7.14 illustrates *bidirectional congestion control*, a host-based resource-allocation technique. The destination end host controls the rate at which it sends traffic, based on observable network conditions, such as delay and packet loss. If a source detects long delays and packet losses, it slows down its packet flow rate. All sources in the network adjust their packet-generation rate similarly; thus, congestion comes under control. Implicit signaling is widely used in packet-switched networks, such as the Internet.

In bidirectional signaling, network routing nodes alert the source of congested resources by setting bits in the header of a packet intended for the source. An alternative mechanism for signaling is to send control packets as choke packets to the source, alerting it of any congested resources in the network. The source then slows down its rate of packet flow. When it receives a packet, the source checks for the bit that indicates congestion. If the bit is set along the path of packet flow, the source slows down its

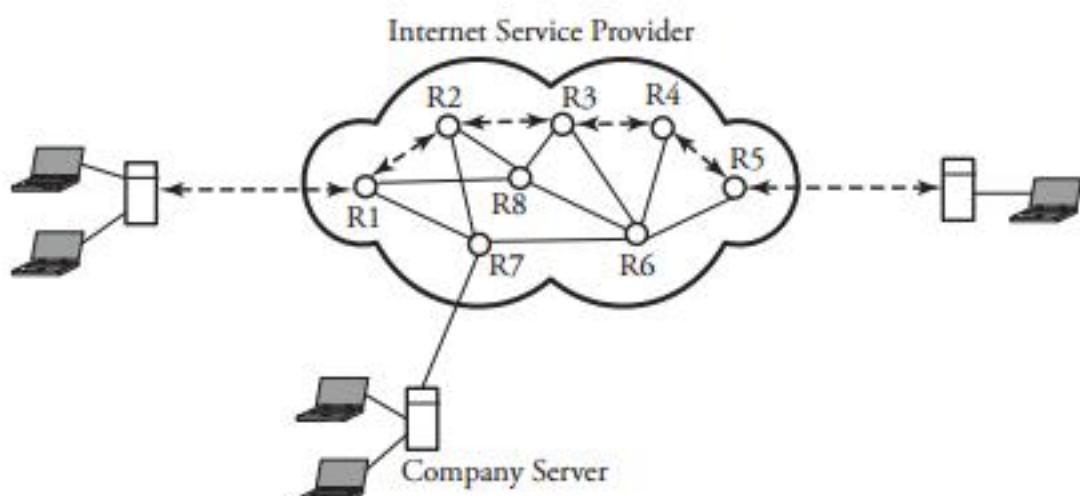


Figure 7.14 Bidirectional congestion control

sending rate. Bidirectional signaling can also be imposed by the window-based and rate-based congestion-control schemes discussed earlier.

7.6.3 Random Early Detection (RED)

Random early detection (RED) avoids congestion by detecting and taking appropriate measures early. When packet queues in a router's buffer experience congestion, they discard all incoming packets that could not be kept in the buffer. This *tail-drop policy* leads to two serious problems: global synchronization of TCP sessions and prolonged congestion in the network. RED overcomes the disadvantages of the tail-drop policy in queues by randomly dropping the packets when the average queue size exceeds a given minimum threshold.

From the statistical standpoint, when a queueing buffer is full, the policy of random packet drop is better than multiple-packet drop at once. RED works as a feedback mechanism to inform TCP sessions that the source anticipates congestion and must reduce its transmission rate. The packet-drop probability is calculated based on the weight allocation on its flow. For example, heavy flows experience a larger number of dropped packets. The average queue size is computed, using an exponentially weighted moving average so that RED does not react to spontaneous transitions caused by bursty Internet traffic. When the average queue size exceeds the maximum threshold, all further incoming packets are discarded.

RED Setup at Routers

With RED, a router continually monitors its own queue length and available buffer space. When the buffer space begins to fill up and the router detects the possibility of

congestion, it notifies the source implicitly by dropping a few packets from the source. The source detects this through a time-out period or a duplicate ACK. Consequently, the router drops packets earlier than it has to and thus implicitly notifies the source to reduce its congestion window size.

The “random” part of this method suggests that the router drops an arriving packet with some drop probability when the queue length exceeds a threshold. This scheme computes the average queue length, $E[N_q]$, recursively by

$$E[N_q] = (1 - \alpha)E[N_q] + \alpha N_i, \quad (7.3)$$

where N_i is the instantaneous queue length, and $0 < \alpha < 1$ is the weight factor. The average queue length is used as a measure of load. The Internet has bursty traffic, and the instantaneous queue length may not be an accurate measure of the queue length. RED sets minimum and maximum thresholds on the queue length, N_{min} and N_{max} , respectively. A router applies the following scheme for deciding whether to service or drop a new packet. If $E[N_q] \geq N_{max}$, any new arriving packet is dropped. If $E[N_q] \leq N_{min}$, the packet is queued. If $N_{min} < E[N_q] < N_{max}$, the arriving packet is dropped with probability P given by

$$P = \frac{\delta}{1 - c\delta}, \quad (7.4)$$

where coefficient c is set by the router to determine how quickly it wants to reach a desired P . In fact, c can be thought of as the number of arriving packets that have been queued. We can then obtain δ from

$$\delta = \frac{E[N_q] - N_{min}}{N_{max} - N_{min}}. \quad (7.5)$$

In essence, when the queue length is below the minimum threshold, the packet is admitted into the queue. Figure 7.15 shows the variable setup in RED congestion avoidance. When the queue length is between the two thresholds, the packet-drop probability increases as the queue length increases. When the queue length is above the maximum threshold, the packet is always dropped. Also, shown in Equation (7.4), the packet-drop probability depends on a variable that represents the number of arriving packets from a flow that has been queued. When the queue length increases, all that is needed is to drop one packet from the source. The source then halves its congestion window size.

Once a small number of packets are dropped, the associated sources reduce their congestion windows if the average queue length exceeds the minimum threshold, and

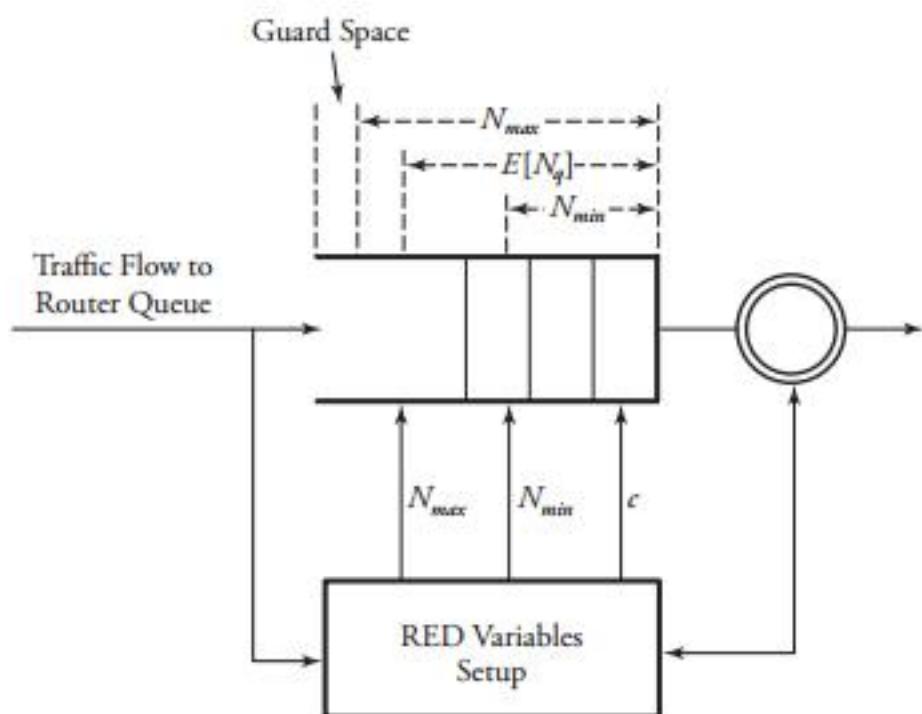


Figure 7.15 Variables setup in the RED congestion-avoidance method

therefore the traffic to the router drops. With this method, any congestion is avoided by an early dropping of packets.

RED also has a certain amount of fairness associated with it; for larger flows, more packets are dropped, since P for these flows could become large. One of the challenges in RED is to set the optimum values for N_{min} , N_{max} , and c . Typically, N_{min} has to be set large enough to keep the throughput at a reasonably high level but low enough to avoid congestion. In practice, for most networks on the Internet, the N_{max} is set to twice the minimum threshold value. Also, as shown by *guard space* in Figure 7.15, there has to be enough buffer space beyond N_{max} , as Internet traffic is bursty.

7.6.4 A Quick Estimation of Link Blocking

A number of techniques can be used to evaluate a communication network's blocking probabilities. These techniques can vary according to accuracy and to network architectures. One of the most interesting and relatively simple approaches to calculating the level of blocking involves the use of Lee's probability graphs. Although Lee's technique requires approximations, it nonetheless provides reasonably accurate results.

Serial and Parallel Connection Rules

Lee's method is based on two fundamental rules of *serial* and *parallel* connections. Each link is represented by its blocking probability, and the entire network of links

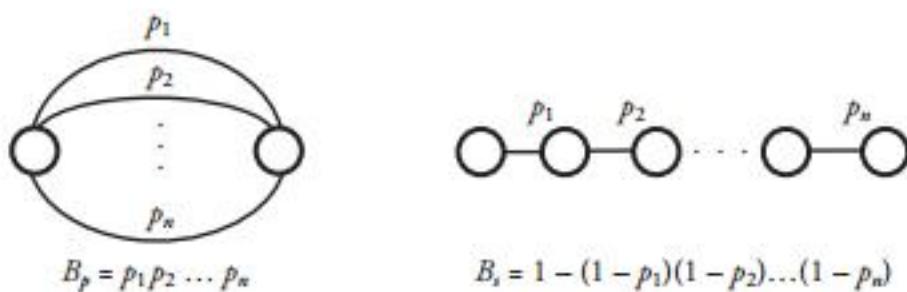


Figure 7.16 Models for serial and parallel connection rules

is evaluated, based on blocking probabilities represented on each link, using one or both of the rules. This approach is easy to formulate, and the formula directly relates to the underlying network structures, without requiring any other detailed parameters. Thus, Lee's approach provides insight into the network structures, giving a remarkable solution for performance evaluations.

Let p be the probability that a link is busy, or the percentage of link utilization. Thus, the probability that a link is idle is denoted by $q = 1 - p$. Now, consider a simple case of two links in parallel to complete a connection with probabilities p_1 and p_2 . The composite blocking probability, B_p , is the probability that both links are in use, or

$$B_p = p_1 p_2. \quad (7.6)$$

If these two links are in series to complete a connection, the blocking probability, B_s , is determined as 1 minus the probability that both links are available:

$$B_s = 1 - (1 - p_1)(1 - p_2). \quad (7.7)$$

We can generalize the estimation of a network of links by two fundamental rules.

1. Rule 1: For a parallel connection of links, the blocking probability is estimated by forming the product of the blocking probabilities for the subnetworks, as shown in Figure 7.16. Let a source and a destination be connected in general through n links with probabilities p_1 through p_n , respectively, as shown in Figure 7.16. Therefore, if the source and the destination are linked through parallel connections, the probability of blocking B_p is obtained from a product form as follows:

$$B_p = p_1 p_2 \dots p_n. \quad (7.8)$$

2. Rule 2: For a serial connection of links, the probability of blocking is estimated by forming the product of the probabilities of no blocking for the network. This method makes the assumption that the probability that a given link is busy is

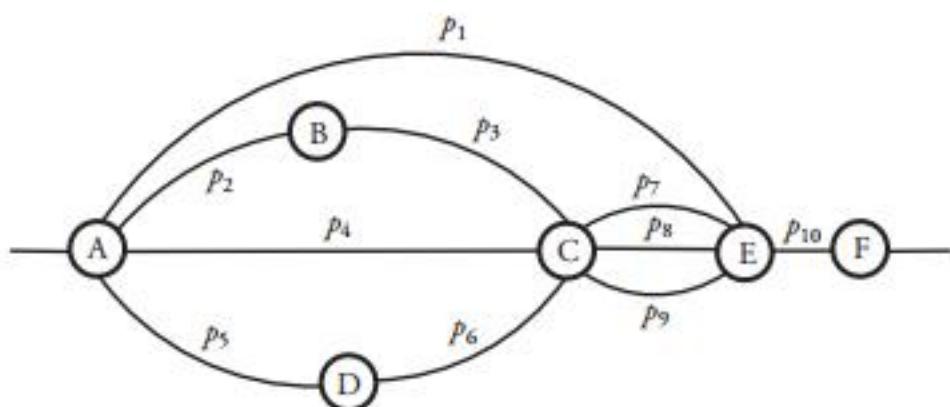


Figure 7.17 A network of links and their blocking probabilities

independent from link to link. Although this independence assumption is not strictly correct, the resulting estimates are sufficiently accurate. If links are in series, the probability of blocking is obtained from

$$B_s = 1 - (1 - p_1)(1 - p_2)\dots(1 - p_n). \quad (7.9)$$

Example. The network in Figure 7.17 has six nodes, A through F, interconnected with ten links. The corresponding blocking probabilities of links, p_1 through p_{10} , are indicated on the corresponding links. For this network, find the overall blocking probability from A to F.

Solution. This network consists of a number of serial and parallel connections: for example, $p_{ADC} = 1 - (1 - p_5)(1 - p_6)$, $p_{ABC} = 1 - (1 - p_2)(1 - p_3)$, and $p_{CE} = p_7p_8p_9$. Thus:

$$B = p_{AF} = 1 - \left\{ 1 - [1 - (1 - p_4 \cdot p_{ABC} \cdot p_{ADC})(1 - p_{CE})] p_1 \right\} (1 - p_{10}).$$

7.7 Summary

This chapter focused on layer 3 of the protocol stack reference model. Two least-cost-path algorithms are Dijkstra's algorithm—a centralized least-cost algorithm designed to find the best path from a source to a destination through the optimization of path costs—and the Bellman-Ford algorithm—a distributed least-cost approach by which each node retrieves information from reachable nodes to each of its neighbors. Non-least-cost algorithms—*deflection*, or flood algorithms—are not optimal but they have their own useful applications.

A number of practical routing protocols exist. One category is intradomain routing protocols, such as OSPF and RIP. A router equipped with OSPF is aware of its local link-cost status and periodically sends updates to all surrounding routers. OSPF is based on *link-state routing*, by which routers exchange packets carrying status information of their adjacent links. A router collects all the packets and determines the network topology, thereby executing its own shortest-route algorithm. RIP is one of the most widely used routing protocols in the Internet infrastructure; routers exchange information about reachable networks and the number of hops. RIP uses the Bellman-Ford algorithm.

By contrast, *interdomain routing protocols*, such as BGP, let two contributing routers exchange routing information even if they are located in two different autonomous systems. Each router sends information to all internal neighbors and decides whether the new route is possible.

Congestion control at the network layer is a major issue. In computer networks, congestion represents some form of overload, generally resulting from resources shortage at links and devices. *Unidirectional congestion control* can take several forms: *back-pressure signaling*, *transmission of choke packets*, and *traffic policing*. *Bidirectional congestion control* is a host-based resource-allocation scheme. In this category, *random early detection* is one congestion-avoidance technique. An approximation method to calculate *link blocking* follows two simple rules: two links in series and two links in parallel. These two rules can be used to estimate the blocking in any complex combination of links.

The next chapter presents fundamentals of the transport layer and end-to-end protocols. The transport layer is responsible for signaling and file transfer.

7.8 Exercises

1. Consider a packet-switching network for server communications with n nodes. For each of the following topologies, sketch the network, and give the average number of hops between a pair of servers (include the server-to-node link as a hop):
 - (a) *Star*: one central node with all servers attached to the central node
 - (b) *Bidirectional ring*: each node connects to two other nodes to form a loop, with each node connected to one server
 - (c) *Fully connected*: each node is directly connected to all other nodes, with each node connected to one server
 - (d) *A ring with $n - 1$ nodes connected to a central node*, with each node in the ring connected to one server.

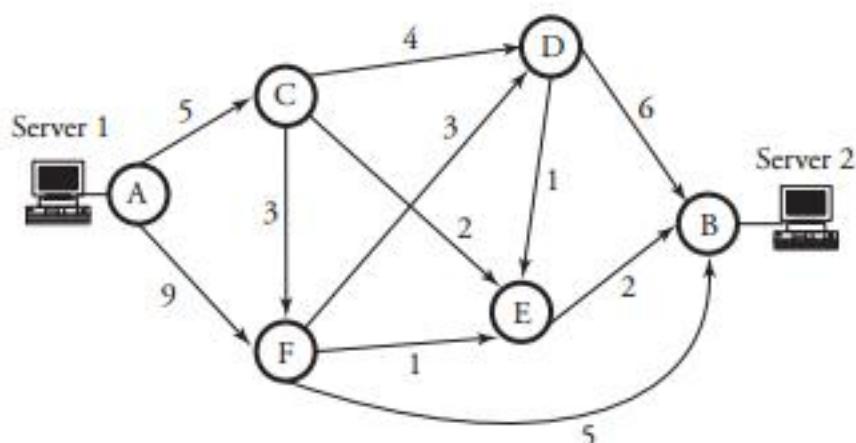


Figure 7.18 Exercises 2–4 network example

2. Figure 7.18 shows a network.
 - (a) Find the least-cost path between the two servers, using Dijkstra's algorithm.
 - (b) Show an iteration graph as the least-cost path is developed.
3. For the network shown in Figure 7.18:
 - (a) Find the least-cost path between the two servers, using the Bellman-Ford algorithm.
 - (b) Show an iteration graph as the least-cost path is developed.
4. Consider again the network in Figure 7.18. Assume that each link is bidirectional and that the cost of either direction is identical.
 - (a) Find the least-cost path between the two servers, using Dijkstra's algorithm.
 - (b) Show an iteration graph as the least-cost path is developed.
5. The network shown in Figure 7.19 is a snapshot of a practical consisting of seven routers. The load on each link is normalized to a number indicated on that link.
 - (a) Find the least-cost path between the two routers R1 and R7, using Dijkstra's Algorithm.
 - (b) Show an iteration graph as the least-cost path is developed.
6. For the network shown in Figure 7.19:
 - (a) Find the least-cost path between the two servers, using the Bellman-Ford algorithm.
 - (b) Show an iteration graph as the least-cost path is developed.
7. The practical network shown in Figure 7.20 is a WAN consisting of seven routers R₁ through R₇ interconnecting four LANs. The load on each link is normalized

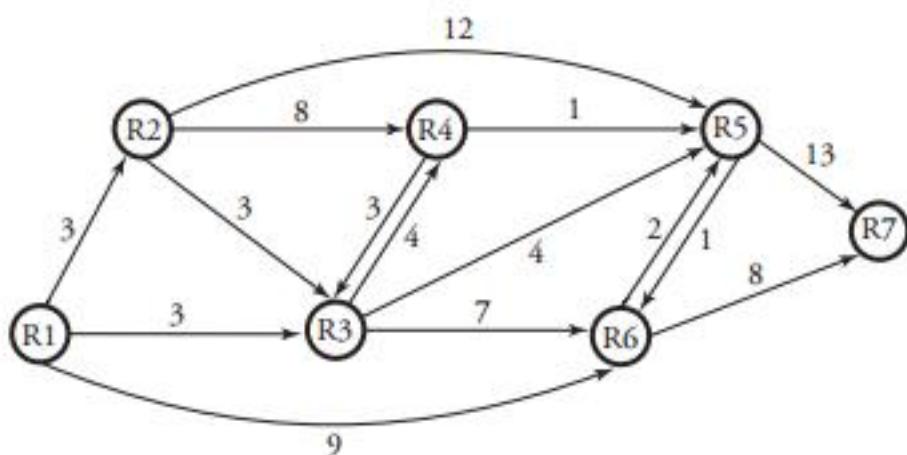


Figure 7.19 Exercises 5–6 network example

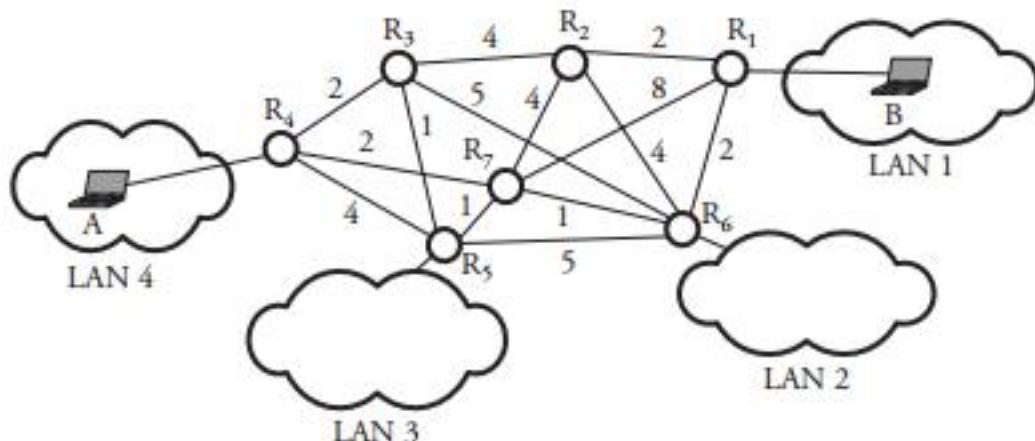


Figure 7.20 Exercise 7 network example

to a number indicated on that link. Assume that all links are bidirectional with equal indicated load.

- Find the least-cost path between the two routers R1 and R₇ connecting users A and B, using Dijkstra's algorithm.
 - Show an iteration graph as the least-cost path is developed.
8. The practical network shown in Figure 7.20 is a WAN consisting of seven routers R₁ through R₇ interconnecting four LANs. The load on each link is normalized to a number indicated on that link. Assume that all links are bidirectional with equal indicated load.
- Find the least-cost path between the two routers R1 and R₇ connecting users A and B, using Bellman-Ford's algorithm.
 - Show an iteration graph as the least-cost path is developed.

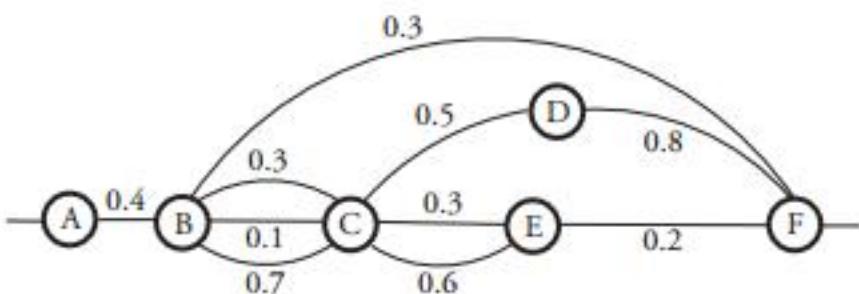


Figure 7.21 Exercise 8 network of links

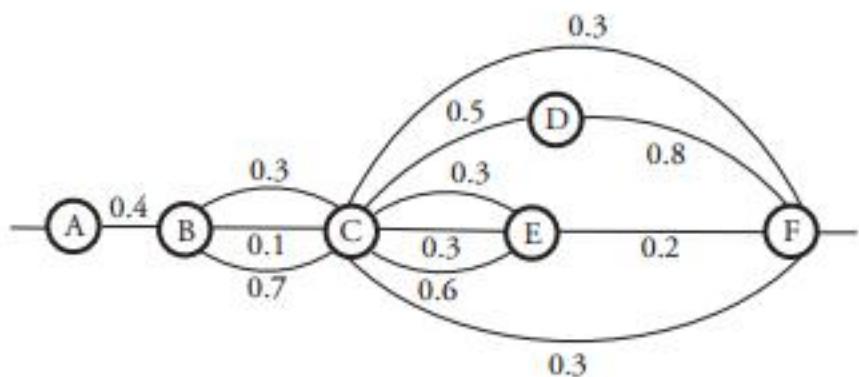


Figure 7.22 Exercise 9 network of links

9. The network in Figure 7.21 has six interconnected nodes, A through F. The corresponding blocking probabilities of links are indicated on the corresponding links. For this network, find the overall blocking probability from A to F.
10. Figure 7.22 shows a large communication network of six interconnected nodes, A through F. The corresponding blocking probabilities of links are indicated on the corresponding links. For this network, find the overall blocking probability from A to F.
11. *Computer simulation project.* Write a computer program to simulate Dijkstra's algorithm for the seven-node network shown in Figure 7.20. Your program must
 - (a) Use a scheme that runs for any hypothetical link cost. Users A and B use the Bellman-Ford algorithm.
 - (b) Compute the least-cost path between any two routers.
12. *Computer simulation project.* Write a computer program to simulate the Bellman-Ford algorithm for the seven-node network shown in Figure 7.20. Your program must
 - (a) Use a scheme that runs for any hypothetical link cost. Users A and B use the Bellman-Ford Algorithm.
 - (b) Compute the least-cost path between any two routers.

CHAPTER 8

Transport and End-to-End Protocols

So far, we have covered networking activities at the physical, link, and network layers of the Internet Protocol stack. This chapter focuses on foundations of layer 4, the transport layer. We study several techniques for Transmission Control Protocol (TCP) congestion control. These techniques use a form of end-to-end congestion control in a TCP session when a sender sends a packet and a receiver acknowledges receipt of the packet. This chapter covers the following topics:

- *Transport layer*
- *Transmission Control Protocol (TCP)*
- *User Datagram Protocol (UDP)*
- *Mobile Transport Protocols*
- *TCP Congestion Control*

We first take a close look at *layer 4, the transport layer*, and explain how a file is transferred. Layer 4 handles the details of data transmission and acts as an interface protocol between a communicating host and a server through a network. We explain reliable end-to-end connection establishment under the *Transmission Control Protocol (TCP)*.

Next, we present the *user datagram protocol (UDP)*, a connectionless transport-layer protocol that is placed on top of the network layer. We also discuss TCP congestion control. Normally, it would take a significant amount of engineering effort to design a network with low or no congestion. *Congestion* in communication networks represents

a state in which the traffic flow exceeds the availability of network resources to service the flow. Generally, congestion occurs at a network because of a lack of resources at links and devices. We distinguish between these two categories of resource deficits and explore precautionary solutions: *flow control* and *congestion control*, respectively. We end the chapter with a discussion of TCP and UDP applications.

8.1 Transport Layer

The *transport layer* handles the details of data transmission. This layer provides a logical communication between application processes, as shown in Figure 8.1. Transport-layer protocols are implemented in the end points but not in network routers. The transport layer ensures a complete data transfer for an application process, free from the issue of physical infrastructure. This layer clearly acts as an interface protocol between a communicating host and a network. For example, a client host in LAN 2 is running an application on a server through a *logical link*. However, the end points can be on the opposite sides of the network, connected via a number of routers, as shown by the two end points via R4-R7-R1.

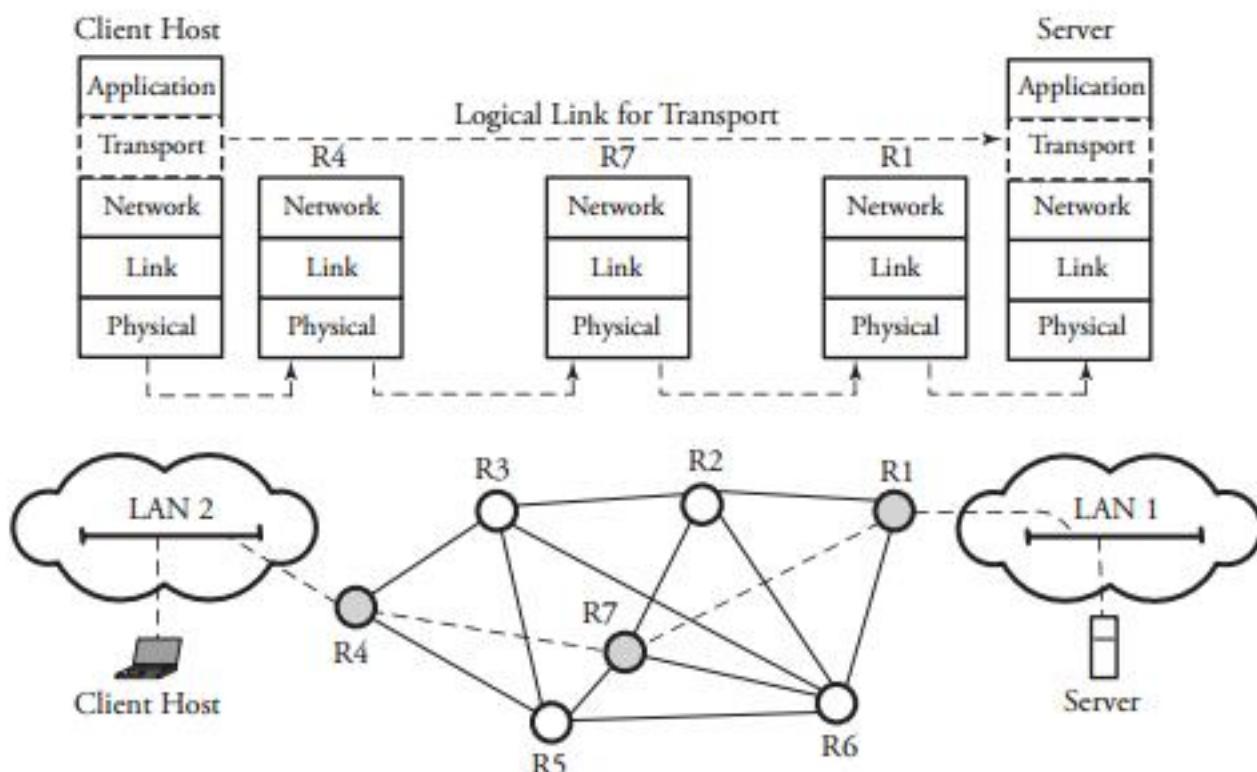


Figure 8.1 Demonstrating “logical” end-to-end communication between a client host and a server at the transport layer

The two most important forms of transportation in the TCP/IP network are governed by two protocols:

- Transmission Control Protocol (TCP)
- User Datagram Protocol (UDP)

TCP is connection oriented and is therefore a reliable service to an invoking application. On the contrary, UDP is a *connectionless* service and is unreliable to the invoking application. A network application design involves making a careful choice between these two transport protocols, as each acts differently to any invoking application.

The transport layer converts application messages into transport-layer packets known as *segments*. The layer manages all end-to-end communication details, such as packet segmentation, packet reassembly, and error checking. The segmentation is performed by breaking the application messages into smaller chunks and attaching a transport-layer header to each chunk to create a segment. Once a segment is formed, it is passed to the network layer, where it is encapsulated within a network-layer packet—a datagram—and transmitted to the destination.

8.1.1 Interaction of Transport and Network Layers

The direct logical link shown in Figure 8.1 reflects the fact that three routers—R4, R7, and R1 act only on the network-layer fields of the datagram and never inspect the fields of the transport-layer segment. This resembles a direct logical, but not physical, link between the two end points. Once the segment is received by the end server, the network layer extracts the transport-layer segment from the datagram and makes it available to the transport layer. At the receiving end, the transport layer processes the segment by extracting the application data in the segment.

Transport protocols can generally offer reliable data transfer service to a certain application in spite of the fact that the underlying network protocol can be unreliable owing to losing packets. A transport protocol can also use its own security procedures to guarantee that application messages are not intruded by unknowns.

8.2 Transmission Control Protocol (TCP)

The *Transmission Control Protocol* (TCP) is a transport-layer protocol that provides a reliable service by using an *automatic repeat request* (ARQ). In addition, TCP provides congestion control, using a sliding-window scheme, that introduced in Section 4.6.2.

TCP is built over IP services by facilitating a two-way connection between the host application and the destination application.

As a connection-oriented protocol, TCP requires a connection to be established between each two applications. A connection is set up by defining variables the protocol requires and storing them in the *transmission control block*. After establishing the connection, TCP delivers packets in sequence and in bytestream. TCP can either send the data received from the application layer as a single packet or split it into multiple packets and transmit them if an underlying physical network poses a limitation.

The requisites of layer 4 are to transfer data without errors to make sure that packets follow the same sequence. A host application that needs to send a certain sort of data stores it in a send buffer. The data is then transmitted as a bytestream. The transmitting host creates a *segment* that contains a sequence of bytes. The segment is appended to a TCP header that specifies the destination application port and a *sequence number*. When it arrives at the destination, the segment is verified for its integrity. After making sure that the packet is not a duplicate and that the segment number lies in the range of the local buffer, the receiver accepts the packet. The receiver can accept packets out of order. In that case, the receiver simply repositions them in the buffer to obtain an in-order sequence before making the data available to the application. Acknowledgments are cumulative and traverse in the reverse direction.

8.2.1 TCP Segment

As defined earlier, a TCP *segment* is a TCP session packet containing part of a TCP bytestream in transit. The fields of the TCP header segment are shown in Figure 8.2. The TCP segment contains a minimum of 20 bytes of fixed fields and a variable-length options field. The details of the fields are as follows.

- *Source port* and *destination port* specify, respectively, the user's port number, which sends packets and the user's port number, which receives packets.
- *Sequence number* is a 32-bit filed that TCP assigns to each first data byte in the segment. The sequence number restarts from 0 after the number reaches $2^{32} - 1$.
- *Acknowledgment number* specifies the sequence number of the next byte that a receiver waits for and acknowledges receipt of bytes up to this sequence number. If the SYN field is set, the acknowledgment number refers to the *initial sequence number* (ISN).
- *Header length (HL)* is a 4-bit field indicating the length of the header in 32-bit words.

Byte:

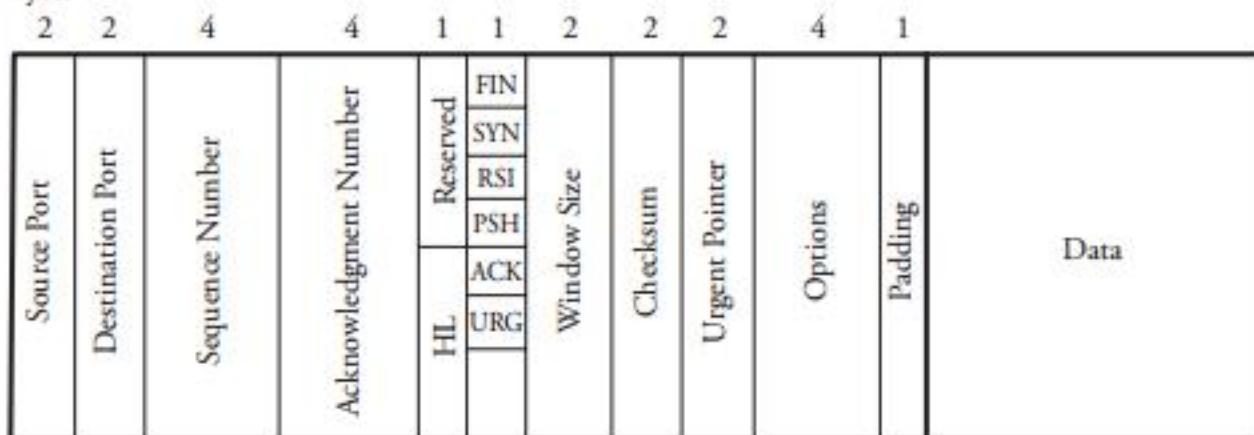


Figure 8.2 TCP segment format

- *Urgent* (URG) is a 1-bit field implying that the urgent-pointer field is applicable.
- *Acknowledgment* (ACK) shows the validity of an acknowledgment.
- *Push* (PSH), if set, directs the receiver to immediately forward the data to the destination application.
- *Reset* (RST), if set, directs the receiver to abort the connection.
- *Synchronize* (SYN) is a 1-bit field used as a connection request to synchronize the sequence numbers.
- *Finished* (FIN) is a 1-bit field indicating that the sender has finished sending the data.
- *Window size* specifies the advertised window size.
- *Checksum* is used to check the validity of the received packet. (See Section 4.5.2 for details.)
- *Urgent pointer* (URG), if set, directs the receiver to add up the values in the urgent-pointer field and the sequence number field to specify the last byte number of the data to be delivered urgently to the destination application.
- *Options* is a variable-length field that specifies the functions that are not available as part of the basic header.

One of the possible *options* is *maximum segment size* (MSS). A receiver uses this option to specify the maximum segment size it can receive. A total of 16 bits are provided to specify this option. Thus, the maximum segment size is limited to 65,535 bytes minus 20 bytes of TCP header and minus 20 bytes of IP header, resulting in

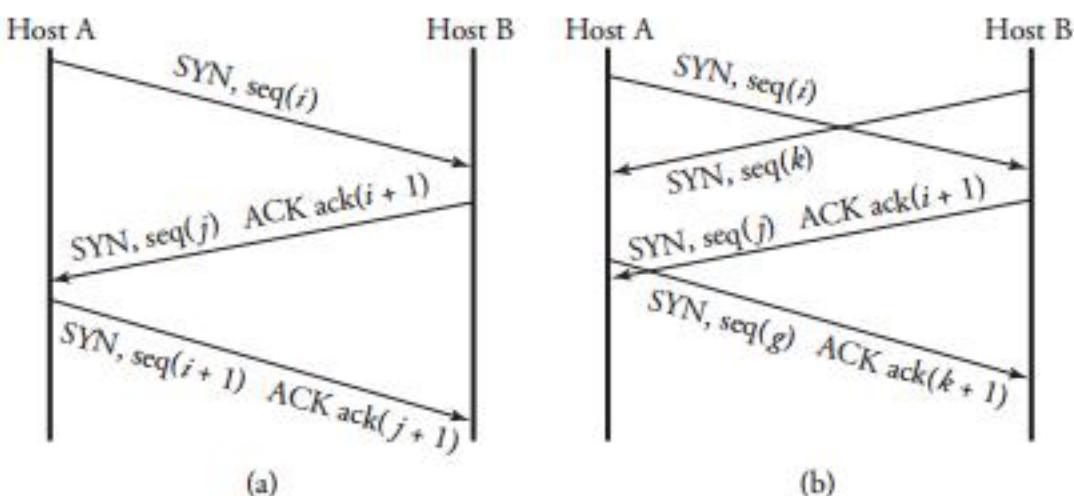


Figure 8.3 TCP signaling: (a) establishment of a connection; (b) collision of two connection requests

65,495 bytes. The default MSS for TCP is 536 bytes. Another possible *option* is the *window scale*. This option is used to increase the size of the advertised window beyond the specified $2^{16} - 1$ in the header. The advertised window can be scaled to a maximum of 2^{14} .

8.2.2 Connection Setup

As a connection-oriented protocol, TCP requires an explicit connection set-up phase. Connection is set up using a three-step mechanism, as shown in Figure 8.3 (a). Assume that host A is a sender and host B a destination. First, the sender sends a connection request to the destination. The connection request comprises the initial sequence number indicated by $seq(i)$, with the SYN bit set. Next, on receipt of the connection request, the destination sends an acknowledgment, $ack(i + 1)$, back to the source, indicating that the destination is waiting for the next byte. The destination also sends a request packet comprising the sequence number, $seq(j)$, and the SYN bit set. Finally, at the third step, the sender returns an acknowledgment segment, $ack(j + 1)$, specifying that it is waiting for the next byte. The sequence number of this next segment is $seq(i + 1)$. This process establishes a connection between the sender and the receiver.

The connection process uses different initial sequence numbers between the sender and the receiver to distinguish between the old and new segments and to avoid the duplication and subsequent deletion of one of the packets. In Figure 8.3, both end-point hosts simultaneously try to establish a connection. In this case, since they recognize the connection requests coming from each other, only one connection is established. If one of the segments from a previous connection arrives late, the receiver accepts the

packet, presuming that it belongs to the new connection. If the packet from the current connection with the same sequence number arrives, it is considered a duplicate and is dropped. Hence, it is important to make sure that the initial sequence numbers are different.

When one of the end points decides to abort the connection, a segment with the RST bit set is sent. Once an application has no data to transmit, the sender sends a segment with the FIN bit set. The receiver acknowledges receipt of this segment by responding with an ACK and notifies the application that the connection is terminated. Now, the flow from the sender to the receiver is terminated. However, in such cases, the flow from the receiver to the sender is still open. The receiver then sends a segment with the FIN bit set. Once the sender acknowledges this by responding with an ACK, the connection is terminated at both ends.

8.3 User Datagram Protocol (UDP)

The *user datagram protocol* (UDP) is another transport-layer protocol that is placed on top of the network layer. UDP is a connectionless protocol, as no handshaking between sending and receiving points occurs before sending a segment. UDP does not provide a reliable service. The enhancement provided by UDP over IP is its ability to check the integrity of flowing packets. IP is capable of delivering a packet to its destination but stops delivering them to an application. UDP fills this gap by providing a mechanism to differentiate among multiple applications and deliver a packet to the desired application. UDP can perform error detection to a certain extent but not to the level that TCP can.

8.3.1 UDP Segment

The format of the UDP segment is shown in Figure 8.4. The segment starts with the *source port*, followed by the *destination port*. These port numbers are used to identify the ports of applications at the source or the destination, respectively. The source port identifies the application that is sending the data. The destination port helps UDP to demultiplex the packet and directs it to the right application. The *UDP length* field indicates the length of the UDP segment, including both the header and the data. *UDP checksum* specifies the computed checksum when transmitting the packet from the host. If no checksum is computed, this field contains all zeroes. When this segment is received at the destination, the checksum is computed; if there is an error, the packet is discarded.

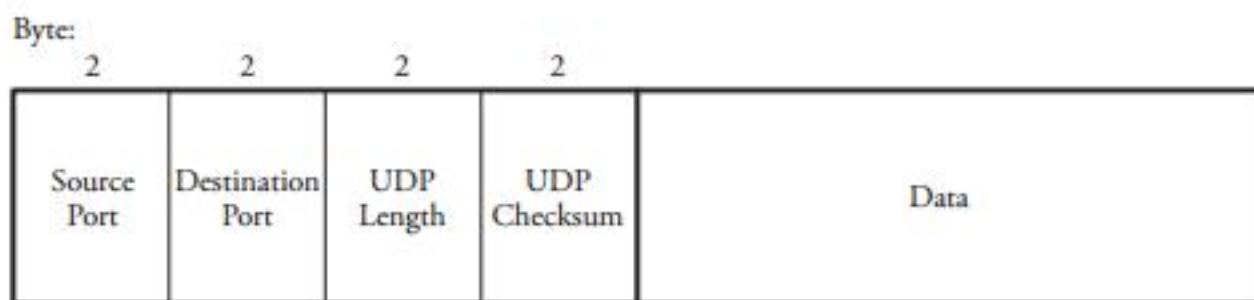


Figure 8.4 User datagram protocol segment

UDP takes messages from the application process, attaches source and destination port number fields and two other fields, and makes this segment available to the network layer. The network layer encapsulates the segment into an IP datagram (packet) and finds the best path to deliver the segment to the other end host.

8.3.2 Applications of TCP and UDP

Although TCP provides a reliable service and UDP does not, many applications fit better in the communication system by using UDP, for the following reasons:

- *Faster delivery of the application object.* TCP is equipped with congestion-control mechanism and requires the guarantee of data delivery at any timing cost. This may not be well suited for real-time applications.
- *Support for a larger number of active clients.* TCP maintains connection state and does track connection parameters. Therefore, a server designated to a particular application supports fewer active clients when the application runs over TCP rather than over UDP.
- *Smaller segment header overhead.* UDP has only 8 bytes of overhead, whereas the TCP segment has 20 bytes of header.

The first category includes such applications as *e-mail*, *the Web*, *remote terminal access*, and *file transfer*. These applications run over TCP, as they all require reliable data-transfer service. The second category of applications, including *DNS*, *RIP routing table updates*, and *SNMP network management*, run over UDP rather than over TCP. For example, RIP updates are transmitted periodically, and reliability may not be an issue, as a more recent update can always replace a lost one.

8.4 Mobile Transport Protocols

In wireless mobile networks, both UDP and TCP have their own applications. However, some modifications are needed in these protocols to become appropriate for wireless networks. One obvious reason for the modification requirement is the fact that a user (host) may move to a remote area and a seamless connection still is desirable.

8.4.1 TCP for Mobility

TCP provides a reliable data delivery owing to the feature of its connection-oriented nature. The most challenging aspect of providing TCP services to a mobile host is the prevention of disruption caused by poor wireless link quality. A poor link quality typically causes to lose TCP data segments leading to a possible timeout. If the poor quality wireless channel is persistent for even a short persistent, the window remains small causing a low throughput.

One option to solve this problem is to disallow a sender to shrink its congestion window when packets are lost for any reason. If a wireless channel soon recovers from disconnection, the mobile host begins to receive data immediately. There are a few other protocols for this purpose among which the *Indirect Transmission Control Protocol* (I-TCP), and the *fast transmit* will be the focus of our discussion.

Indirect TCP

Assume two hosts, one mobile and the other one fixed, are trying to establish an I-TCP connection as shown in Figure 8.5. The I-TCP scheme first splits the connection into two separate connections. One connection is established between the mobile host and the mobile switching center (MSC) which normally attached to the wireless base station, and the other between the MSC and the fixed host. So a connection consists of two pieces: the wireless link and the fixed link. At this point, the wireless and wired link-characteristics remain hidden from the transport layer. The separation of connections into two distinct parts is advantageous to the MSC to better manage communication overheads for a mobile host. This way the throughput of connection is enhanced since the mobile host may be near the from the MSC.

A TCP connection on the wireless portion can support disconnections, and user mobility in addition to wired TCP features such as notification to higher layers on changes in the available bandwidth. Also, the flow control and congestion control mechanisms on the wireless link remain separated from those on the wired link. In the I-TCP scheme, the TCP acknowledgments are separate for the wireless and the wired links of the connection. This resembles two different TCP connections linked

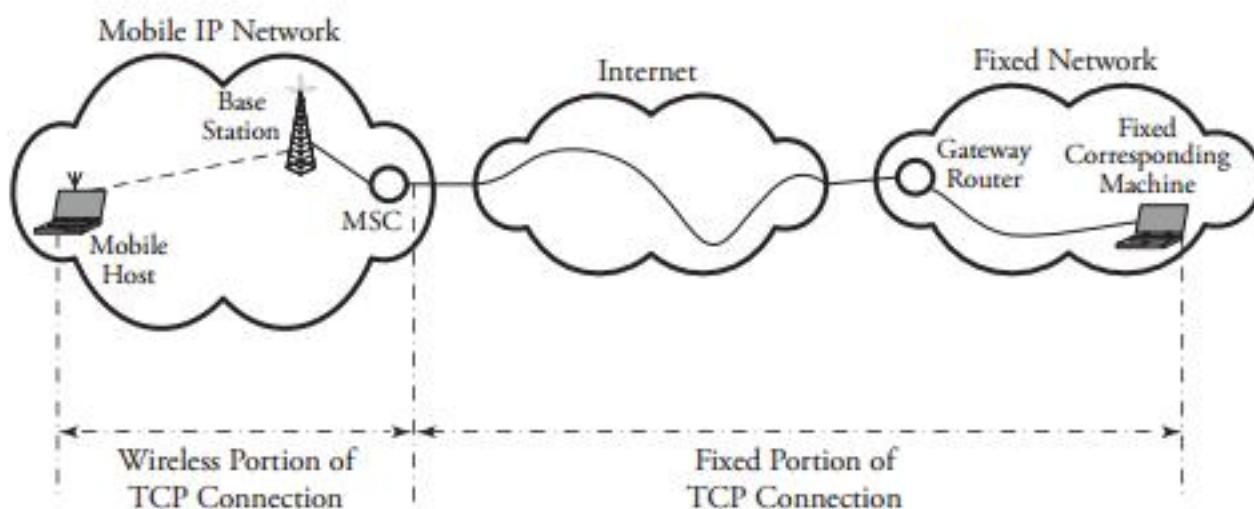


Figure 8.5 Indirect TCP for mobile hosts

together. Note that, if for any reason the mobile host disconnects the communication on the wireless portion, the sender may not become aware of the disconnection as the wired portion is still in tact, and the base station still delivers the TCP segments to the mobile host. Consequently, the sender of segments may not know of segments being delivered to the mobile host.

Fast Retransmit Mobile TCP

The *fast retransmit* protocol is another TCP scheme for mobility. Fast Retransmit improves the connection throughput especially during a cell handoff. This scheme does not split the TCP connection to wireless and wired connections. Once two wireless cell MSCs hand off the switching function for a mobile host, the mobile host stops receiving TCP segments. This may be interpreted by the sender as a situation of congestion thereby a congestion control such as window size reduction or retransmitting may begin to be implemented. This may also result in a long timeout causing the mobile host to wait a long period of time. With the fast retransmit TCP, the last old acknowledgment is triplicated and retransmitted by the mobile host as soon as it finishes a handoff. This results in significant reduction of the congestion window.

8.4.2 UDP for Mobility

UDP is used in wireless mobile IP networks for several reasons. We learned in Chapter 6 that a mobile host needs to register with a foreign agent. The registration with an agent on a foreign network is done using UDP. This process starts with a foreign agent propagating advertisements using UDP connection. Since the traditional UDP does not

use acknowledgments and does not perform flow control, it is not a preferred choice of transport protocol. One way of handling this situation is to stop sending datagrams to a mobile host once it reports fading. But this method cannot be practical due to its poor quality of connection.

8.5 TCP Congestion Control

TCP uses a form of end-to-end flow control. In TCP, when a sender sends a packet, the receiver acknowledges receipt of the packet. A sending source can use the acknowledgment arrival rate as a measure of network congestion. When it successfully receives an acknowledgment, a sender knows that the packet reached its desired destination. The sender can then send new packets on the network. Both the sender and the receiver agree on a common *window size* for packet flow. The window size represents the number of bytes that the source can send at a time. The window size varies according to the condition of traffic in the network to avoid congestion. Generally, a file of size f with a total transfer time of Δ on a TCP connection results in a *TCP transfer throughput* denoted by r and obtained from

$$r = \frac{f}{\Delta}. \quad (8.1)$$

We can also derive the *bandwidth utilization*, ρ_u , assuming that the link bandwidth is B , by

$$\rho_u = \frac{r}{B}. \quad (8.2)$$

TCP has three congestion-control methods: *additive increase*, *slow start*, and *retransmit*. The following subsections describe these three mechanisms, which are sometimes combined to form the TCP congestion-control scheme.

8.5.1 Additive Increase, Multiplicative Decrease Control

An important variable in TCP congestion control is the value of the *congestion window*. Each connection has congestion window size, w_g . The congestion window represents the amount of data, in bytes, that a sending source is allowed to have in transit at a particular instant of time in a network. *Additive increase, multiplicative decrease control* performs a slow increase in the congestion window size when the congestion in the network decreases and a fast drop in the window size when congestion increases. Let w_m be the *maximum window size*, in bytes, representing the maximum amount of

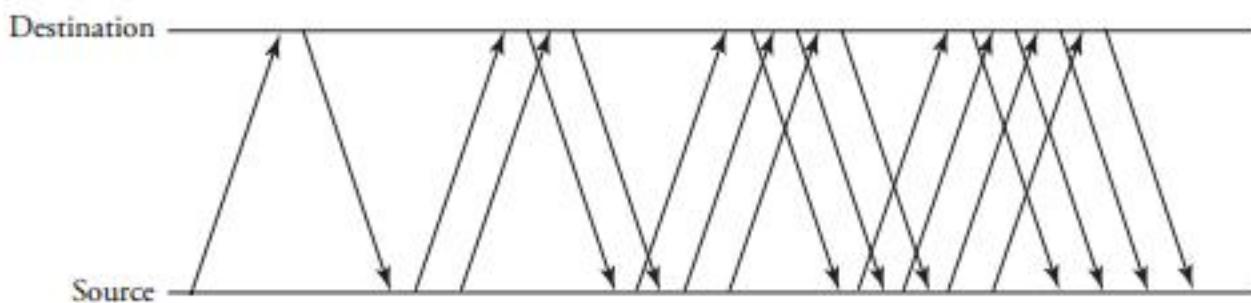


Figure 8.6 Additive increase control for TCP congestion control

unacknowledged data that a sender is allowed to send. Let w_a be the advertised window sent by the receiver, based on its buffer size. Thus,

$$w_m = \min(w_g, w_a). \quad (8.3)$$

By having w_m replace w_a , a TCP source is not permitted to transmit faster than the network or the destination. The challenge in TCP congestion control is for the source node to find a right value for the congestion window. The congestion window size varies, based on the traffic conditions in the network. TCP watches for timeout as a sign of congestion. One can arrange timeouts to be used as acknowledgments to find the best size for the congestion window. This is done because the implications for having too large a window are much worse than having too small a window. This TCP technique requires that the timeout values be set properly. Two important factors in setting timeouts follow.

1. Average *round-trip times* (RTTs) and RTT standard deviations are based to set timeouts.
2. RTTs are sampled once every RTT is completed.

Figure 8.6 depicts the additive-increase method. The congestion window is interpreted in terms of packets rather than bytes. Initially, the source congestion window is set to one packet. Once it receives an acknowledgment for the packet, the source increments its congestion window by one packet. So the source transmits two packets at that point. On successful receipt of both acknowledgments, the source once again increments the congestion window by one packet (additive increase).

In practice, the source increments its congestion window by a small amount for each acknowledgment instead of waiting for both acknowledgments to arrive. If a timeout occurs, the source assumes that congestion is developing and therefore sets the congestion window size to half its previous value (multiplicative decrease). The minimum congestion window size is called maximum *segment size*, which represents

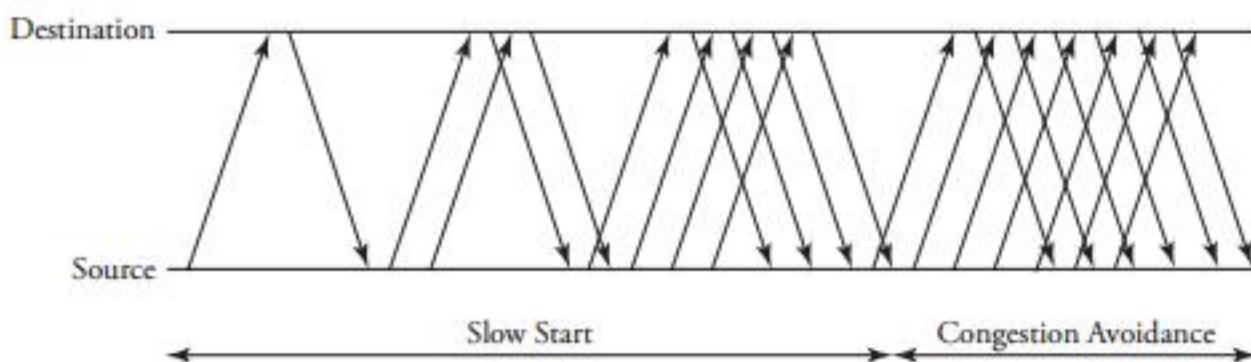


Figure 8.7 Slow-start timing between a source and a destination

one packet. In general, a TCP segment is defined as a TCP session packet containing part of a TCP bytestream in transit.

8.5.2 Slow Start Method

Additive increase is ideally suited when the network operates near capacity. Initially, it would take a considerable amount of time to increase the congestion window. The *slow-start method* increases the congestion window size nonlinearly and in most cases exponentially, as compared to the linear increase in additive increase. Figure 8.7 shows the slow-start mechanism. In this case, the congestion window is again interpreted in packets instead of bytes.

A source initially sets the congestion window to one packet. When its corresponding acknowledgment arrives, the source sets the congestion window to two packets. Now, the source sends two packets. On receiving the two corresponding acknowledgments, TCP sets the congestion window size to 4. Thus, the number of packets in transit doubles for each round-trip time. This nonlinearity trend of increase in the window size continues as seen in the figure. With this method of congestion control, routers on a path may not be able to service the flow of traffic, as the volume of packets increases nonlinearly. This congestion-control scheme by itself may lead to a new type of congestion. The slow-start method is normally used

1. just after a TCP connection is set up or
2. when a source is blocked, waiting for a timeout.

A new variable, *congestion threshold*, is defined. This variable is a saved size of the congestion window when a timeout arrives. When a timeout occurs, the threshold is set to half the congestion window size. Then the congestion window is reset to one packet and ramped up all the way to the congestion threshold, using the slow-start method. Once the connection is established, a burst of packets is sent during a slow

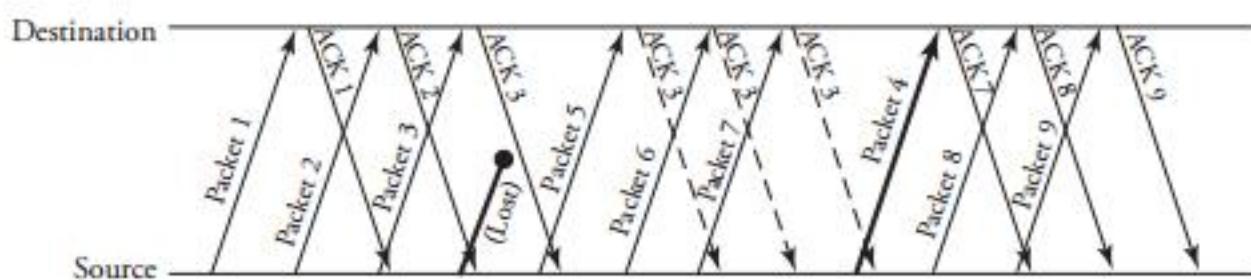


Figure 8.8 Timing of retransmit method between a source and a destination

start. Then, a number of packets are lost, and the source admits no waiting time for acknowledgments. Finally, a timeout occurs, and the congestion window is reduced. Thus, a timeout results in the reduction of the congestion window, as in the previous scheme. The congestion threshold and the congestion window are reset. Slow start is used to increase the congestion window size exponentially.

After the congestion threshold is reached, additive increase is used. At this point, packets may be lost for a while, and the source removes the waiting time for acknowledgments. Then, a timeout occurs immediately, and the congestion window size is reduced. The congestion threshold is reset, and the congestion window is reset to one packet. Now, the source uses slow start to ramp up, and then additive increase is used. After reaching the congestion threshold, additive increase is used. This pattern continues, creating a pulse-type plot. The reason for the large packet loss initially with slow start is that it is more aggressive in the beginning in order to learn about the network. This may result in a few packet losses, but it seems to be better than the conservative approach, in which the throughput is very small.

8.5.3 Fast Retransmit Method

Fast retransmit is based on the concept of duplicate acknowledgment (ACK). The additive-increase and slow-start mechanisms have idle periods, during which the source admits no waiting time for an acknowledgment. Fast retransmit of segments sometimes leads to a retransmission of the lost packet before the associated timeout periods.

Each time it receives an out-of-order packet, the destination should respond with a duplicate ACK of the last successful in-order packet that has arrived. This must be done even if the destination has already acknowledged the packet. Figure 8.8 illustrates the process. The first three packets are transmitted, and their acknowledgments are received.

Now, assume that packet 4 is lost. Since it is not aware of this lost packet, the source continues to transmit packet 5 and beyond. However, the destination sends the

duplicate acknowledgment of packet 3 to let the source know that it has not received packet 4. In practice, once the source receives three duplicate acknowledgments, it retransmits the lost packet. Fast recovery is another improvement to TCP congestion control. When congestion occurs, instead of dropping the congestion window to one packet, the congestion window size is dropped to half, and additive increase is used. Thus, the slow-start method is used only during the initial connection phase and when a timeout occurs. Otherwise, additive increase is used.

8.5.4 TCP Congestion Avoidance Methods

Network congestion is a traffic bottleneck between a source and a destination. *Congestion avoidance* uses precautionary algorithms to avoid possible congestion in a network. Otherwise, TCP congestion control is applied once congestion occurs in a network. TCP increases the traffic rate to a point where congestion occurs and then gradually reduces the rate. It would be better if congestion could be avoided. This would involve sending some precautionary information to the source just before packets are discarded. The source would then reduce its sending rate, and congestion could be avoided to some extent.

Source-Based Congestion Avoidance

Source-based congestion avoidance detects congestion early from end-hosts. An end host estimates congestion in the network by using the round-trip time and throughput as it measures. An increase in round-trip time can indicate that routers' queues on the selected routing path are increasing and that congestion may happen. The source-based schemes can be classified into four basic algorithms:

1. *Use of round trip time (RTT) as a measure of congestion in the network.* As queues in the routers build up, the RTT for each new packet sent out on the network increases. If the current RTT is greater than the average of the minimum and maximum RTTs measured so far, the congestion window size is reduced.
2. *Use of RTT and window size to set the current window size.* Let w be the current window size, w_o be the old window size, r be the current RTT, and r_o be the old RTT. A window RTT product is computed based on $(w - w_o)(r - r_o)$. If the product is positive, the source decreases the window size by a fraction of its old value. If the product is negative or 0, the source increases the window size by one packet.
3. *Use of throughput as a measure to avoid congestion.* During every RTT, a source increases the window size by one packet. The achieved throughput is then compared

with the throughput when the window size was one packet smaller. If the difference is less than half the throughput at the beginning of the connection when the window size was one packet, the window size is reduced by one packet.

4. *Use of throughput as a measure to avoid congestion.* But this time, the algorithm uses two parameters: the current throughput and the expected throughput to avoid congestion.

TCP normalized is presented next, as an example for the fourth algorithm.

TCP Normalized Method

In the *TCP normalized method*, the congestion window size is increased in the first few seconds, but the throughput remains constant, because the capacity of the network has been reached, resulting in an increase in the queue length at the router. Thus, an increase in the window size results in any increase in the throughput. This traffic over and above available bandwidth of the network is called *extra data*. The idea behind TCP normalized is to maintain this extra data at a nominal level. Too much of extra data may lead to longer delays and congestion. Too little extra data may lead to an underutilization of resources, because the available bandwidth changes owing to the bursty nature of Internet traffic. The algorithm defines the expected value of the rate $E[r]$ as

$$E[r] = \frac{w_g}{r_m}, \quad (8.4)$$

where r_m is the minimum of all the measured round-trip times, and w_g is the congestion window size. We define A_r as the actual rate and $(E[r] - A_r)$ as the rate difference. We also denote the maximum and minimum threshold to be ρ_{max} and ρ_{min} , respectively. When the rate difference is very small—less than ρ_{min} —the method increases the congestion window size to keep the amount of extra data at a nominal level. If the rate difference is between ρ_{min} and ρ_{max} , the congestion window size is unaltered. When the rate difference is greater than ρ_{max} , there is too much extra data, and the congestion window size is reduced. The decrease in the congestion window size is linear. The TCP normalized method attempts to maintain the traffic flow such that the difference between expected and actual rates lies in this range.

8.6 Summary

All end-to-end communications, such as packet segmentation, packet reassembly, and error checking are managed through the *transport layer*. The transport layer is responsible for connection signaling and file transfer. The two most important forms of

transport are the *Transmission Control Protocol* (TCP) and the *User Datagram Protocol* (UDP).

TCP provides a reliable service. Reliability is provided using an *automatic repeat request* (ARQ) protocol. TCP also provides flow control using a sliding-window scheme, as well as congestion control. *Additive increase, multiplicative decrease control* performs a slow increase in the congestion window size when network congestion decreases and a fast drop in the window size when congestion increases. A better technique is the *slow-start method*, which increases the congestion window size nonlinearly. The *fast retransmit* of segments sometimes leads to a retransmission of the lost packet before the associated timeout periods. We looked at various congestion-avoidance mechanisms, including the TCP normalized mechanism.

UDP is another transport-layer protocol described in this chapter. UDP is placed on top of the network layer and is a connectionless protocol, as there is no handshaking between sending and receiving points before sending a segment.

The next chapter presents the application layer in the protocol stack. This layer is responsible for networking applications, *e-mail* and the *World Wide Web* (WWW). The chapter ends with a discussion of network management.

8.7 Exercises

1. Consider a wide area network in which two hosts, A and B, are connected through a 100 km communication link with the data speed of 1 Gb/s. Host A wants to transfer the content of a CD-ROM with 200 Kb of music data while host B reserves portions of its 10 parallel buffers, each with the capacity of 10,000 bits. Use Figure 8.3 and assume that host A sends a SYN segment, where $ISN = 2,000$ and $MSS = 2,000$ and that host B sends $ISN = 4,000$ and $MSS = 1,000$. Sketch the sequence of segment exchange, starting with host A sending data at time $t = 0$. Assume host B sends ACK every five frames.
2. Consider that host 1 transfers a large file of size f to host 2 with $MSS = 2,000$ bytes over a 100 Mb/s link.
 - (a) Knowing that the TCP sequence number field has 4 bytes, find f such that TCP sequence numbers are not exhausted.
 - (b) Find the time it takes to transmit f . Include the link, network, and transport headers attached to each segment.
3. Assume that a TCP connection is established. Find the number of round-trip times the connection takes before it can transmit n segments, using
 - (a) Slow-start congestion control

- (b) Additive increase congestion control
4. We want to understand the fairness index of resource allocations. Suppose that a congestion-control scheme can face five possible flows with the following throughput rates: $B_1 = 1 \text{ Gb/s}$, $B_2 = 1 \text{ Gb/s}$, $B_3 = 1 \text{ Gb/s}$, $B_4 = 1.2 \text{ Gb/s}$, and $B_5 = 16 \text{ Gb/s}$.
- Calculate the fairness index for this scheme for B_1 , B_2 , and B_3 .
 - What useful information does the result of part (a) provide?
 - Now, consider all five flows, and calculate the fairness index for this scheme.
 - What would the result of part (c) mean to each flow?
5. Assume that a TCP connection is established over a moderately congested link. The connection loses one packet every five segments (packets).
- Can the connection survive at the beginning with the linear portion of congestion avoidance?
 - Assume that the sender knows that the congestion remains in the network for a long time. Would it be possible for the sender to have a window size greater than five segments? Why?
6. A TCP connection is established over a 1.2 Gb/s link with a round-trip time of 3.3 ms. To transmit a file of size 2 MB, we start sending it, using 1 K packets.
- How long does the transmission take if an additive increase, multiplicative decrease control with a window size of $w_g = 500 \text{ K}$ is used?
 - Repeat part (a), using slow-start control.
 - Find the throughput of this file transfer.
 - Find the bandwidth utilization for this transfer.
7. Consider that an established TCP connection has a round-trip time of approximately 0.5 second and forms a window size of $w_g = 6 \text{ K}$. The sending source transmits segments (packets) every 50 ms, and the destination acknowledges each segment every 50 ms. Now assume that a congestion state develops in this connection such that the destination does not receive a segment. This loss of segment is detected by the fast-retransmit method at the fourth receipt of duplicate ACK.
- Find the amount of time the sending source has lost if the source uses the arrival of duplicate ACKs as a sign for moving the window forward one segment.
 - Repeat part (a), this time under a condition that the sending source waits to receive the ACK of the retransmitted packet before moving the window forward one segment.

CHAPTER 9

Applications and Network Management

Having presented basic concepts and definitions in networking to the point that an end-to-end connection can be established, we now look at the last layer of the protocol stack—the *application layer*—and certain network-management issues. This chapter examines Internet applications and their supporting protocols and services. These services demonstrate how users perceive a computer network and express the power of Internet technology. This chapter focuses on the following topics:

- *Application-layer overview*
- *Domain Name System (DNS)*
- *Remote login protocols*
- *Electronic mail (e-mail)*
- *File transfer and FTP*
- *World Wide Web (WWW) and HTTP*
- *Network management*

Layer 5, the *application layer*, determines how a specific user application should use a network. The *Domain Name System (DNS)* server is an essential entity for translating machine or domain names into numerical IP addresses. *Remote login protocols* allow applications to run at a remote site, with results transferred back to their local sites. Two remote login protocols are TELNET and SSH.

The most-applied Internet tool is *e-mail*. The *Simple Mail Transfer Protocol* (SMTP) is one of the protocols for sending *electronic mail* (e-mail) from the mail server of a source to the mail servers of destinations. The *World Wide Web* (WWW), or simply *Web*, is a global network of servers linked together by protocols allowing access to all connected *hypertext* resources.

Finally, this chapter discusses network management. ASN.1 is a formal language for each managed device to be defined in a network under management. MIB is another management tool for accommodating a database of information and characteristics for devices. The *Simple Network Management Protocol* (SNMP) enables a network manager to find the location of a fault. SNMP runs on top of UDP and uses client/server configurations. SNMP commands define how to query information from a server and forward information to a server or a client.

9.1 Application-Layer Overview

The *application layer* is built on the transport layer and provides network services to user applications. The application layer defines and performs such applications as electronic mail (e-mail), remote access to computers, file transfers, newsgroups, and the Web, as well as streaming video, Internet radio and telephony, P2P file sharing, multiuser networked games, streaming stored video clips, and real-time video conferencing.

The application layer has its own software dependencies. When a new application is developed, its software must be able to run on multiple machines, so that it does not need to be rewritten for networking devices, such as routers, that function at the network layer. In a *client/server* architecture for example, a *client* end host requests services from a *server* host. A client host can be on sometimes or always. Figure 9.1 shows an example of application-layer communication.

9.1.1 Client and Server Model

A *client/server model* provides specific computational services, such as partial-time usage services to multiple machines. Reliable communication protocols, such as TCP, allow interactive use of remote servers as well. For example, we can build a server that provides remote image-processing services to clients. Implementing such a communication service requires a server loaded with the application protocol to accept requests and a client to make such requests. To invoke remote image processing, a user first executes a client program establishing a TCP connection to a server. Then, the client begins transmitting pieces of a raw image to the server. The server processes the received objects and sends the results back.

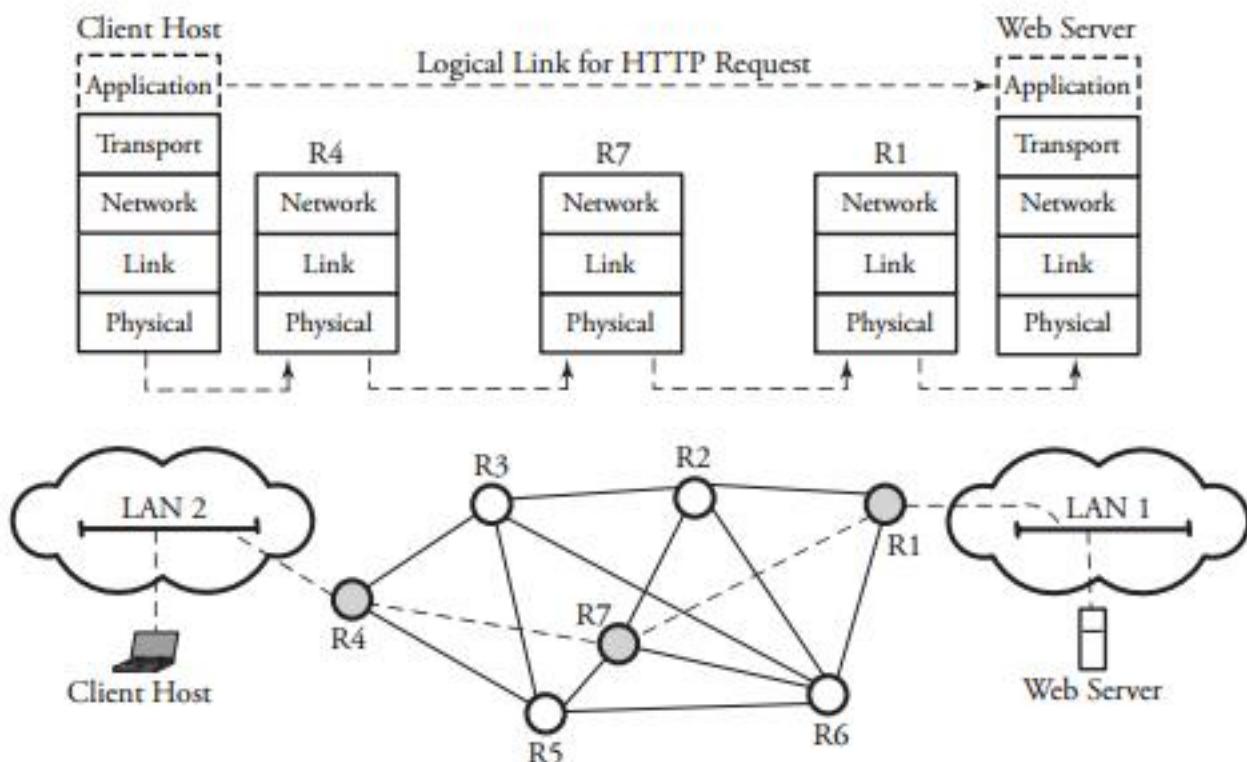


Figure 9.1 Web communication between two end systems

9.2 Domain Name System (DNS)

One of the most important components of the application layer is the *Domain Name System* (DNS) server. DNS is a distributed hierarchical and global directory that translates machine or domain names to numerical IP addresses. DNS can be thought as a distributed database system used to map host names to IP addresses, and vice versa. DNS is a critical infrastructure, and all hosts contact DNS servers when they initiate connections. DNS can run over either UDP or TCP. However, running over UDP is usually preferred, since a fast response for a transaction provided by UDP is required. Some of the information-processing functions a DNS server handles are

- Finding the address of a particular host
- Delegating a subtree of server names to another server
- Denoting the start of the subtree that contains cache and configuration parameters, and giving corresponding addresses
- Naming a host that processes incoming mail for the designated target
- Finding the host type and the operating system information
- Finding an alias for the real name of a host
- Mapping IP addresses to host names

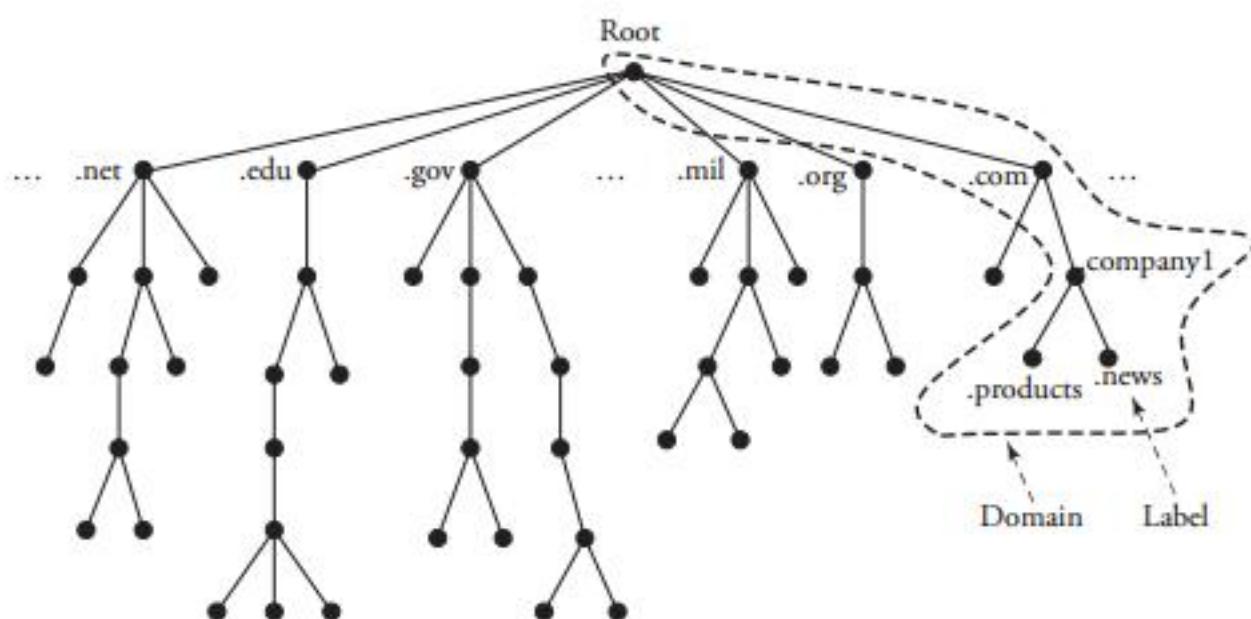


Figure 9.2 Hierarchy of domain name space, labels, and domain names

DNS is an application-layer protocol, and every Internet service provider—whether for an organization, a university campus, or even a residence—has a DNS server. In the normal mode of operation, a host sends UDP queries to a DNS server. The DNS server either replies or directs the queries to other servers. The DNS server also stores information other than host addresses.

The DNS routinely constructs a query message and passes it to the UDP transport layer without any handshaking with the UDP entity running on the destination end system. Then, a UDP header field is attached to the message, and the resulting segment is passed to the network layer. The network layer always encapsulates the UDP segment into a *datagram*. The datagram, or packet, is now sent to a DNS server. If the DNS server does not respond, the fault may be UDP's unreliability.

9.2.1 Domain Name Space

Any entity in the TCP/IP environment is identified by an IP address, which thereby identifies the connection of the corresponding host to the Internet. An IP address can also be assigned a *domain name*. Unique domain names assigned to hosts must be selected from a *name space* and are generally organized in a hierarchical fashion.

Domain names are defined in a tree-based structure with the root at the top, as shown in Figure 9.2. A tree is structured with a maximum of 128 levels, starting at level 0 (root). Each level consists of nodes. A node on a tree is identified by a *label*, with a string of up to 63 characters, except for the root label, which has empty string.

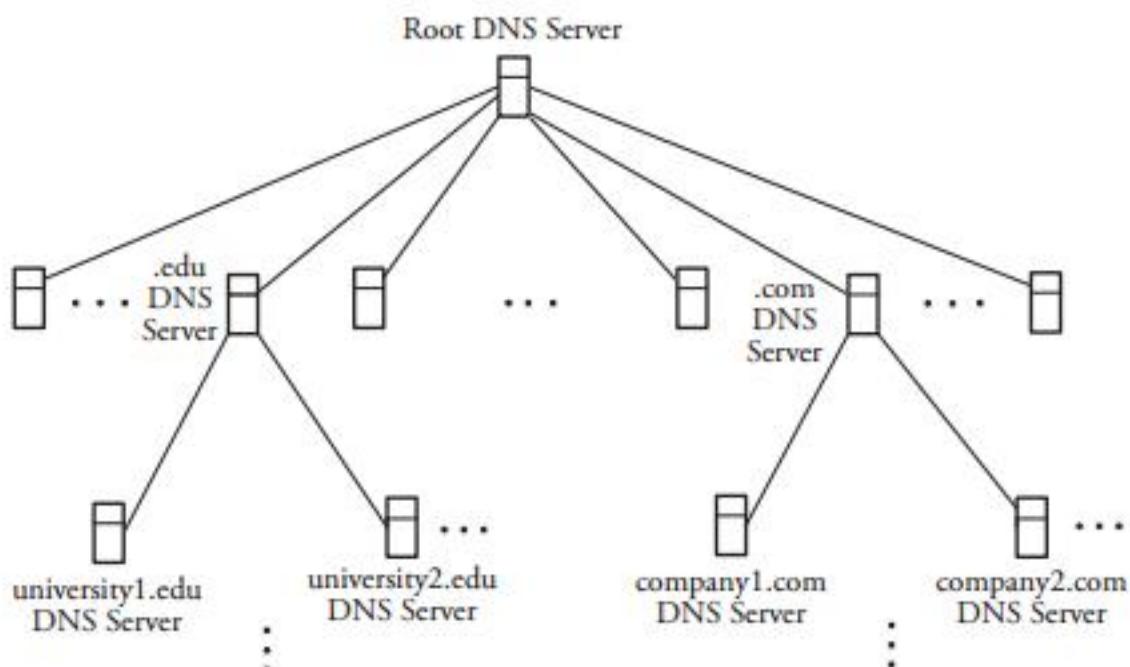


Figure 9.3 Hierarchy of DNS domain name servers

The last label of a domain name expresses the type of organization; other parts of the domain name indicate the hierarchy of the departments within the organization. Thus, an organization can add any suffix or prefix to its name to define its host or resources. A domain name is a sequence of labels separated by dots and is read from the node up to the root. For example, moving from right to left, we can parse as follows: domain name news.company1.com, a commercial organization (.com) and the “news” section of “company1” (news.company1). Domain names can also be partial. For example, company1.com is a partial domain name.

Domain-Name Servers

The domain name space is divided into subdomains, and each domain or subdomain is assigned a *domain name server*. This way, we can form a hierarchy of servers, as shown in Figure 9.3, just as we did for the hierarchy of domain names. A domain name server has a database consisting of all the information for every node under that domain. Each server at any location in the hierarchy can partition part of its domain and delegate some responsibility to another server. The *root server* supervises the entire domain name space. A root server typically does not store any information about domains and keeps references only to servers over which it has authority. Root servers are distributed around the world.

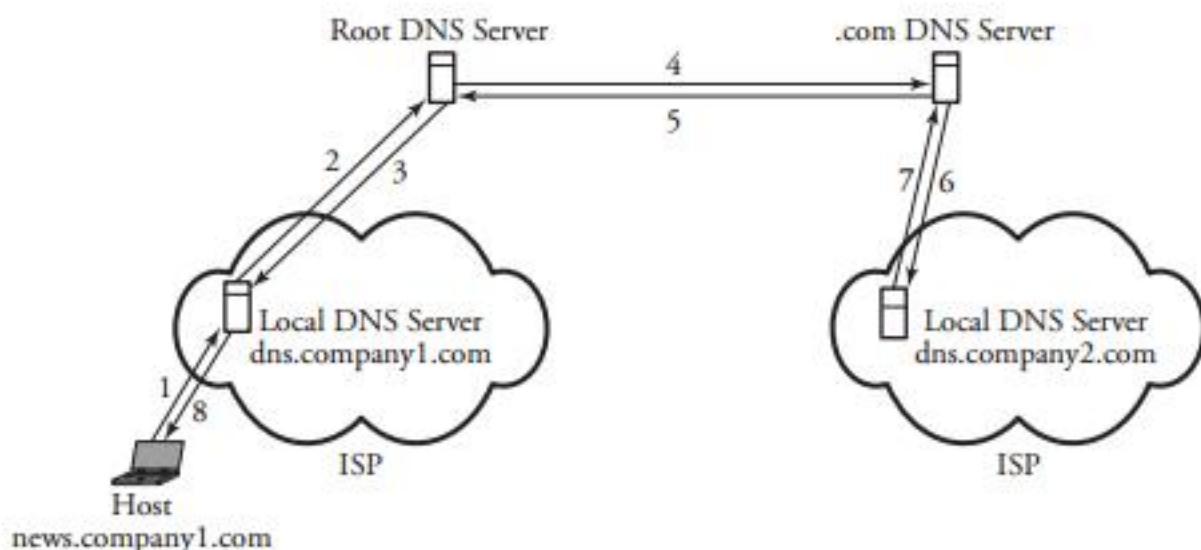


Figure 9.4 Recursive mapping

9.2.2 Name/Address Mapping

DNS operates based on the client/server application. Any client host can send an IP address to a domain name server to be mapped to a domain name. Each host that needs to map an address to a name or vice versa should access the closest DNS server with its request. The server finds and releases the requested information to the host. If the requested information is not found, the server either delegates the request to other servers or asks them to provide the information. After receiving the mapping information, the requesting host examines it for correctness and delivers it to the requesting process.

Mapping can be of either *recursive* or *iterative*. In recursive mapping (Figure 9.4), the client host makes the request to its corresponding DNS server. The DNS server is responsible for finding the answer recursively. The requesting client host asks for the answer through its local DNS server, news.company1.com. Assume that this server contacts the root DNS server, and still the information has not been found. This time, the root DNS server sends the query to the .com server, but the transaction still remains unsuccessful. Finally, .com server sends the query to the local DNS server of the requested place, as dns.company2.com, and finds the answer. The answer to a query in this method is routed back to the origin, as shown in the figure. The local DNS server of the requested place is called the *authoritative server* and adds information to the mapping, called time to live (TTL).

In the iterative approach, the mapping function is as shown in Figure 9.5. In this case, if it does not have the name to provide, the server returns to the client host.

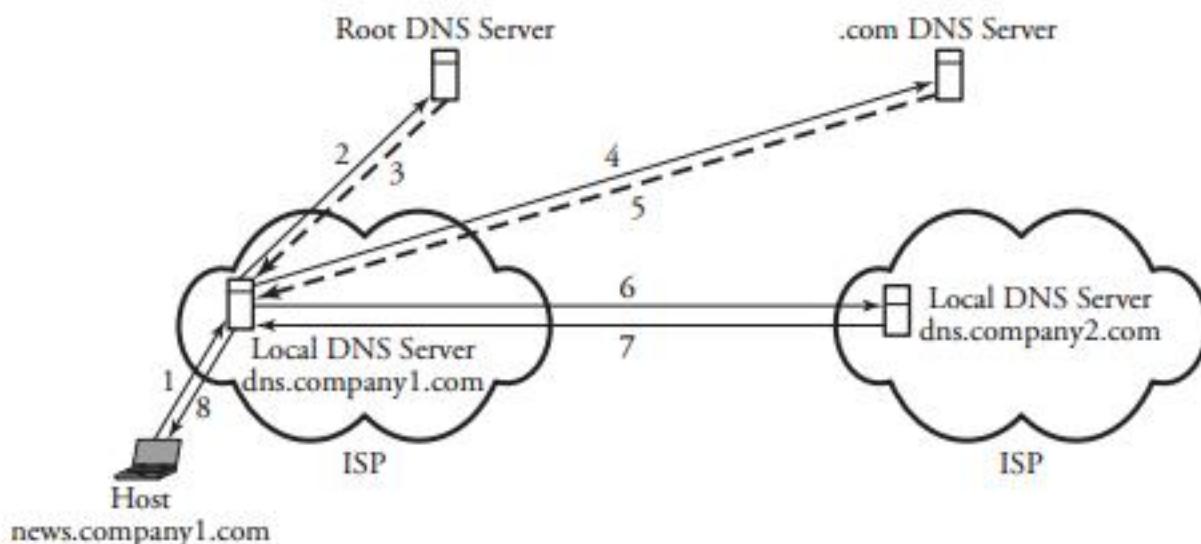


Figure 9.5 Iterative mapping

The host must then repeat the query to the next DNS server that may be able to provide the name. This continues until the host succeeds in obtaining the name. In Figure 9.5, the news.company1.com host sends the query to its own local DNS server, dns.company1.com—thus trying the root DNS server first—and then tries .com server, finally ending up with the local DNS server of the requested place: dns.company2.com.

9.2.3 DNS Message Format

DNS communication is made possible through *query* and *reply* messages. Both message types have the 12-byte header format shown in Figure 9.6. The query message consists of a header and a question message only. The reply message consists of a header and four message fields: *question*, *answer*, *authority*, and *additional information*.

The header has six fields as follows. A client uses the *identification* field to match the reply with the query. This field may appear with a different number each time a client transmits a query. The *flags* field contains subfields that represent the type of the message, such as the type of answer requested or requested DNS recursive or iterative mapping. The *number of questions* field indicates how many queries are in the *question* portion of the message. The *number of answers* shows how many answers are in the *answer* field. For the query message, this field contains all zeros. The *number of authoritative records* field consists of the number of authoritative records in the *authority* portion of a reply message. Similarly, this field is filled by zeros for a query message. Finally, the *number of additional records* field records

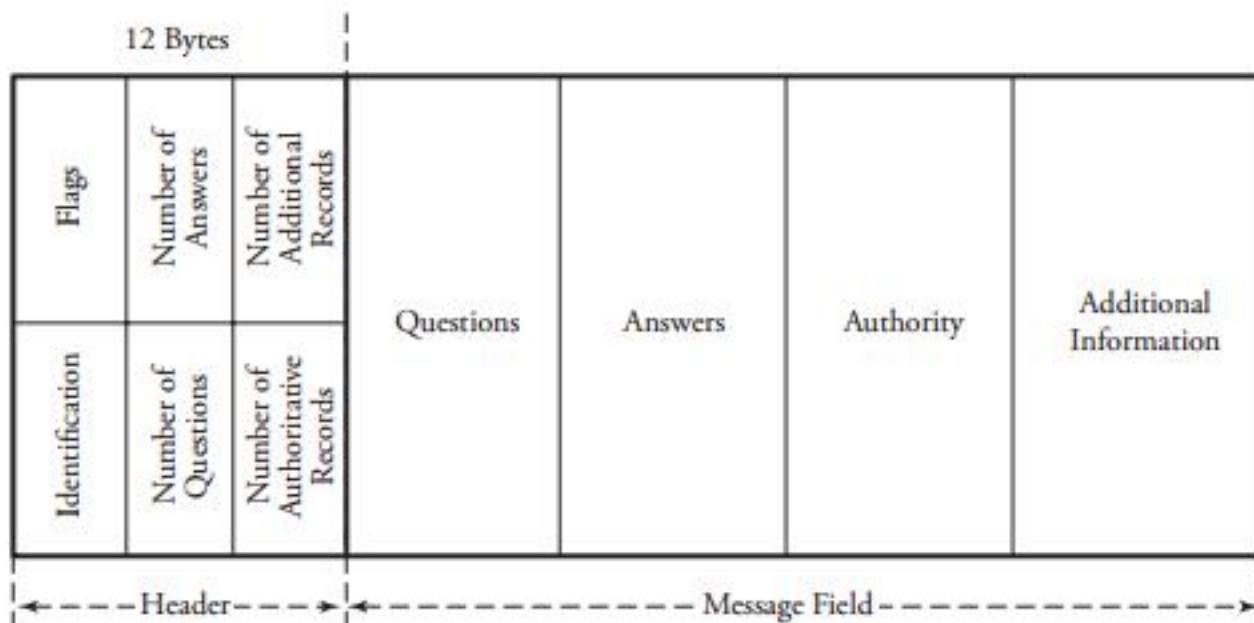


Figure 9.6 DNS message format

are in the additional information portion of a reply message and is similarly filled by zeros in a query message.

The *questions* field can contain one or more questions. The *answers* field belongs only to a reply message and consists of one or more replies from a DNS server to the corresponding client. The *authority* field is present only in reply messages and provides the domain name information about one or more authoritative servers. Finally, the *additional information* field is present only in reply messages and contains other information, such as the IP address of the authoritative server. This additional information helps a client further identify an answer to a question.

The next section explains how domains are added to DNS database. New information and names are included into a DNS database through a *registrar*. On a request for inclusion of a new domain name, a DNS registrar must verify the uniqueness of the name and then enter it into its database.

9.3 Remote Login Protocols

A client/server model can create a mechanism that allows a user to establish a session on the remote machine and then run its applications. This application is known as *remote login*. A user may want to run such applications at a remote site, with results to be transferred back to its local site. For example, an employee working at home can log in to his/her work server to access application programs for doing a project. This

can be done by a client/server application program for the desired service. Two remote login protocols are TELNET and SSH.

9.3.1 TELNET Protocol

TELNET (*terminal network*) is a TCP/IP standard for establishing a connection to a remote system. TELNET allows a user to log in to a remote machine across the Internet by first making a TCP connection and then pass the detail of the application from the user to the remote machine. This application can be interpreted as if the text being transferred had been typed on a keyboard attached to the remote machine.

Logging to Remote Servers

With TELNET, an application program on the user's machine becomes the client. The user's keyboard and its monitor also attach directly to the remote server. The remote-logging operation is based on timesharing, whereby an authorized user has a login name and a password. TELNET has the following properties.

- Client programs are built to use the standard client/server interfaces without knowing the details of server programs.
- A client and a server can negotiate data format options.
- Once a connection is established through TELNET, both ends of the connection are treated symmetrically.

When a user logs in to a remote server, the client's terminal driver accepts the keystrokes and interprets them as characters by its operating system. Characters are typically transformed to a universal character set called *network virtual terminal* (NVT), which uses 7-bit USASCII representation for data. The client then establishes a TCP connection to the server. Texts in the NVT format are transmitted using a TCP session and are delivered to the operating system of the remote server. The server converts the characters back from NVT to the local client machine's format.

The NVT process is necessary because computers to be remotely logged in to differ. In such cases, a specific terminal emulator must also be used by the TELNET client and servers. The client accepts keystrokes from the user's keyboard while accepting characters that the server sends back. On the server side, data moves up through the server's operating system to the server application program. The remote operating system then delivers characters to the application program the user is running. In the meantime, remote character echo is transmitted back from the remote server to the

client over the same path. If the application at the server's site stops reading input for any reason, the associated operating system and, therefore, the server are overwhelmed.

TELNET also offers several options that allow clients and servers to negotiate on nonroutine transactions. For example, one option that the client and the server to pass 8-bit data. In such cases, both client and server must agree to pass 8-bit data before any transmission.

9.3.2 Secure Shell (SSH) Protocol

Secure Shell (SSH), another remote login protocol, is based on UNIX programs. SSH uses TCP for communications but is more powerful and flexible than TELNET and allows the user to more easily execute a single command on a remote client. SSH has the following advantages over TELNET.

- SSH provides a secure communication by encrypting and authenticating messages (discussed in Chapter 10).
- SSH provides several additional data transfers over the same connection by multiplexing multiple channels that are used for remote login.

SSH security is implemented by using *public-key encryption* between the client and remote servers. When a user establishes a connection to a remote server, the data being transmitted remains confidential even if an intruder obtains a copy of the packets sent over an SSH connection. SSH also implements an authentication process on messages so that a server can find out and verify the host attempting to form a connection. Normally, SSH requires users to enter a private password.

A simple SSH interactive session starts with the server's listening on its port specifically designated for secure transmissions. After a password is submitted, SSH starts a shell for the session. SSH can handle several data transfers simultaneously in a same session. This type of remote login service is multiplexed over an SSH connection. SSH can also be used between two machines to carry out *port forwarding* by establishing a secure tunnel. In the SSH remote login utility, a user can allow SSH to automatically splice an incoming TCP connection to a new connection across a tunnel. (The details of tunneling and its applications are explained in Chapter 16.) Data sent over the Internet is ensured to be delivered safe from snooping and alteration.

SSH resembles a tunneling function. For example, when it forms an SSH connection for its port k_1 to a remote server, a client can determine that an incoming TCP connection for this port be automatically forwarded across the tunnel to the server and then spliced to another connection to port k_2 of a second server. This way, the client

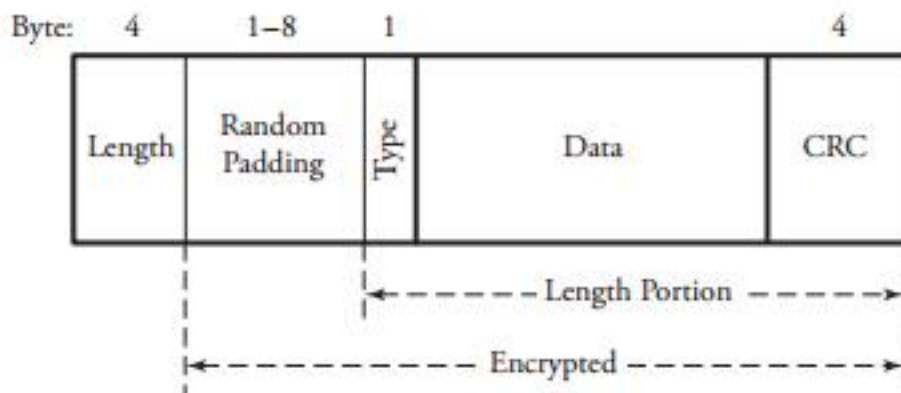


Figure 9.7 SSH packet format

has a TCP connection established on its machine, and the second server makes a TCP connection to the first server. The advantage of port forwarding is that application data can be passed between two sites—the client and the second server—without requiring a second client and server—the first server as a client and the second server.

Figure 9.7 shows the format of an SSH packet.

- *Length* indicates the size of the packet, not including the *length* field or the variable-length *random padding* field that follows it.
- *Padding* causes an intrusion to be more difficult.
- *Type* identifies the type of message.
- *CRC*, or cyclic redundancy check, is an error-detection field (see Chapter 4).

When encryption is enabled, all fields except *length* are encrypted. SSH also permits optional compression of the data, which is useful when SSH is used in low-bandwidth situations. In such cases, the client and the server negotiate compression, and only the *type* and *data* fields are compressed.

9.4 Electronic Mail (E-mail)

9.4.1 Simple Mail Transfer Protocol (SMTP) and E-mail

The *Simple Mail Transfer Protocol* (SMTP) plays a major role in transferring Internet electronic mail. This protocol transfers *electronic mail (e-mail)* from the mail server of a source to the mail servers of destinations. SMTP is older than the Hypertext Transfer Protocol (HTTP), the Web communication protocol, and imposes certain restrictions, such as limits on the size of e-mail content.

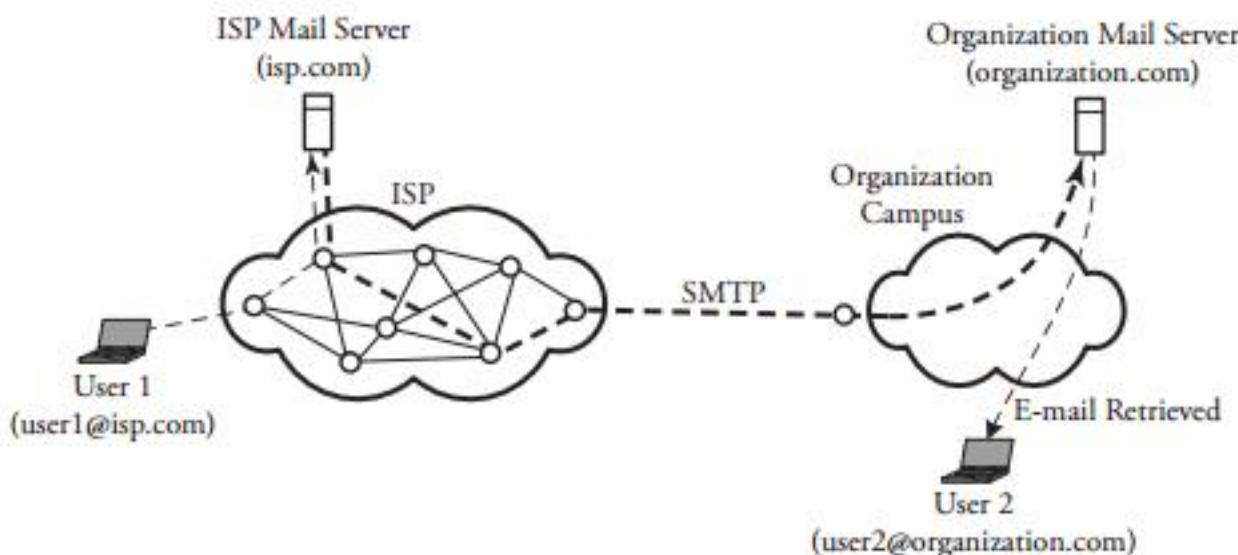


Figure 9.8 Two users exchanging e-mail through SMTP

In Figure 9.8, user 1 is in a residential area, has an Internet service provider (ISP), and is sending an e-mail to user 2, working in an organization. Suppose that the mail servers are `isp.com` and `organization.com`, respectively. Thus, user 1 and user 2 have e-mail addresses of `user1@isp.com` and `user2@organization.com`, respectively. The procedure for an e-mail exchange between user 1 and user 2 is as follows.

Begin SMTP Between Two Users

1. User 1 provides user 2's e-mail address (`user2@organization.com`) and composes its message.
2. User 1 sends the message to its mail server (`isp.com`).
3. Server `isp.com` places the message in its queue.
4. SMTP on user 1's mail server notices the message in the queue and opens a TCP connection with the organization mail server (`organization.com`).
5. Initial SMTP handshaking takes place between the two servers.
6. The message is sent to `organization.com`'s mail server, using the established TCP connection.
7. User 2's mail server receives the message and then puts it in user 2's mailbox, ready to be retrieved by user 2. ■

A *user mailbox* is a space in the mail server allocated to the user to keep its e-mail. Also, SMTP is designed to connect only the two mail servers of the associated parties, regardless of the distance between the two users. Consequently, this protocol involves only the two mail servers of the communicating users.

9.5 File Transfer and FTP

File transfer is another computer networking application. It is always essential that files and information geographically distributed over different locations be shared among the members of a working group. In a certain application, files are typically saved in a server. A user then uses a file transfer protocol to access the server and transfer the desired file. Two file transfer protocols are FTP and SCP.

9.5.1 File Transfer Protocol (FTP)

File Transfer Protocol (FTP) is part of the TCP/IP suite and is very similar to TELNET. Both FTP and TELNET are built on the client/server paradigm, and both allow a user to establish a remote connection. However, TELNET provides a broader access to a user, whereas FTP allows access only to certain files. The essence of this protocol is as follows.

Begin File Transfer Protocol

1. A user requests a connection to a remote server.
2. The user waits for an acknowledgment.
3. Once connected, the user must enter a user ID, followed by a password.
4. The connection is established over a TCP session.
5. The desired file is transferred.
6. The user closes the FTP connection. ■

FTP can also run through a Web browser.

9.5.2 Secure Copy Protocol (SCP)

The *Secure Copy Protocol* (SCP) is similar to TELNET but is secure. Incorporated in the SCP structure are a number of encryption and authentication features that are similar to those in SSH. Also similar is the exchange of commands between local and remote hosts. SCP commands automatically prompt the user for the password information when it is time to access a remote machine. SCP cannot handle file transfer between machines of significantly different architectures.

9.6 World Wide Web (WWW) and HTTP

Application-layer software is the intelligence built for end servers. The *World Wide Web* (WWW), or simply *Web*, is a global network of servers linked by a common protocol

allowing access to all connected hypertext resources. When a client host requests an object, a Web server responds by sending the requested object through browsing tools. A *browser* is a user agent displaying the requested Web page. The *Hyper Text Transfer Protocol* (HTTP) transfers that page at the application layer. HTTP uses TCP rather than UDP, since reliability of delivery is important for Web pages with text. The TCP connection-establishment delay in HTTP is one of the main contributing delay factors associated with downloading Web documents.

HTTP is based on the client/server idea, having a client and a server program, both of which can be executed on different end systems. The communication is carried out through an exchange of HTTP messages. This protocol specifies the structure of these messages. For example, HTTP defines how a pair of client/server hosts should exchange messages. In this context, a Web page consists of files, such as Hypertext Markup Language (HTML) file or an image that can be addressed by a single *uniform resource locator* (URL). A URL is a global address of an HTML document and has two parts. The first part indicates what protocol is used, and the second part determines the IP address of the associated resource.

9.6.1 Web Caching (Proxy Server)

An HTTP request from a user is first directed to the network *proxy server*, or *Web cache*. Once configured by the network, a browser's request for an object is directed to the Web cache, which must contain updated copies of all objects in its defined proximity. The main reason for Web caching is to reduce the response time for a user request. This benefit is much more obvious when the bandwidth to a requested server is limited because of traffic at certain hours of the day. Normally, each organization or ISP should have its own cache providing a high-speed link to its users. Consequently, it is to users' advantages that this rapid method of finding objects be available. This method of Internet access also reduces traffic on an organization's access link to the Internet. The details of Web caching are as follows:

Begin Web Caching Algorithm

1. The source browser makes a TCP connection to the Web cache.
2. The user browser transmits its HTTP request to the Web cache.
3. If it has a copy of the requested object, the Web cache forwards the object to the user browser.

Otherwise the Web cache establishes a TCP connection to the requested server and asks for the object. Once it receives the requested object, the Web cache stores a copy of it and forwards another copy to the requesting user browser over the existing TCP connection. ■

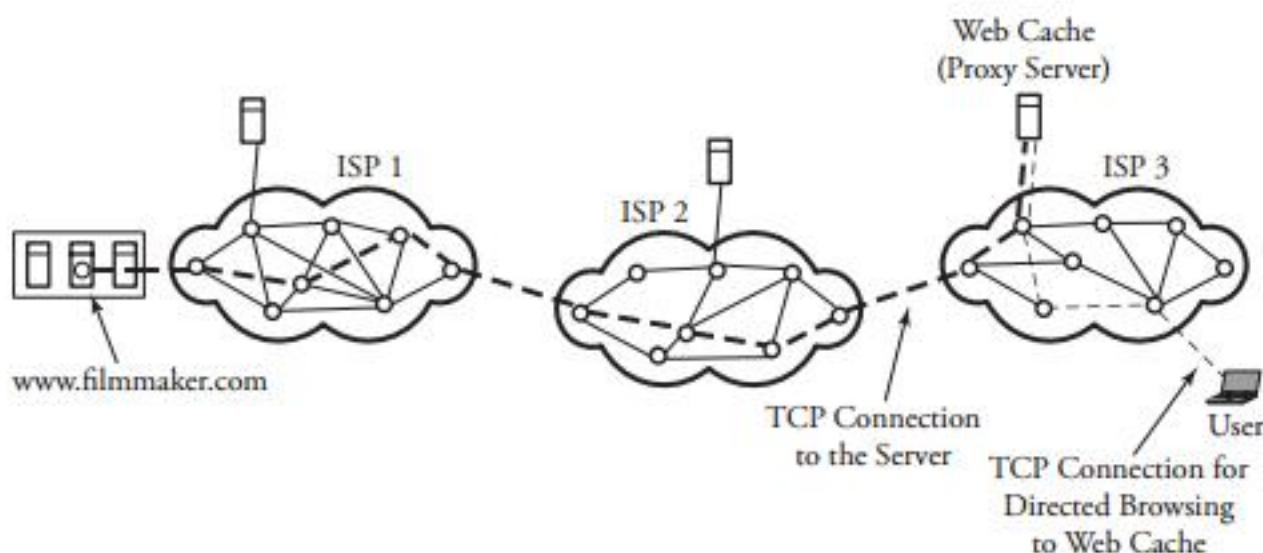


Figure 9.9 A user's browser requesting an object through the Web cache

Figure 9.9 shows three Internet service providers (ISPs). A user in ISP domain 3 is browsing to find and watch an object named <http://www.filmmaker.com> in ISP domain 1. The request for this object is directed to the Web cache, shown by dashed lines. In this example, the Web cache has no record of the requested object and therefore is establishing another TCP connection to update its record.

9.7 Network Management

The main purpose of *network management* is to monitor, manage, and control a network. A network can be structured with many links, routers, servers, and other physical-layer devices, which can be equipped with many network protocols that coordinate them. Imagine when thousands of such devices or protocols are tied together by an ISP and how drastic their management can become to avoid any interruptions in routine services. In this context the purpose of network management is to monitor, test, and analyze the hardware, software, and human elements of a network and then to configure and control those elements to meet the operational performance requirements of the network.

Figure 9.10 illustrates a simple network management scenario in which LANs connect to the Internet. LAN 1 is dedicated to the network administrator facilities. The network administrator can periodically send management packets to communicate with a certain network entity. A malfunctioning component in a network can also initiate communication of its problem to the network administrator.

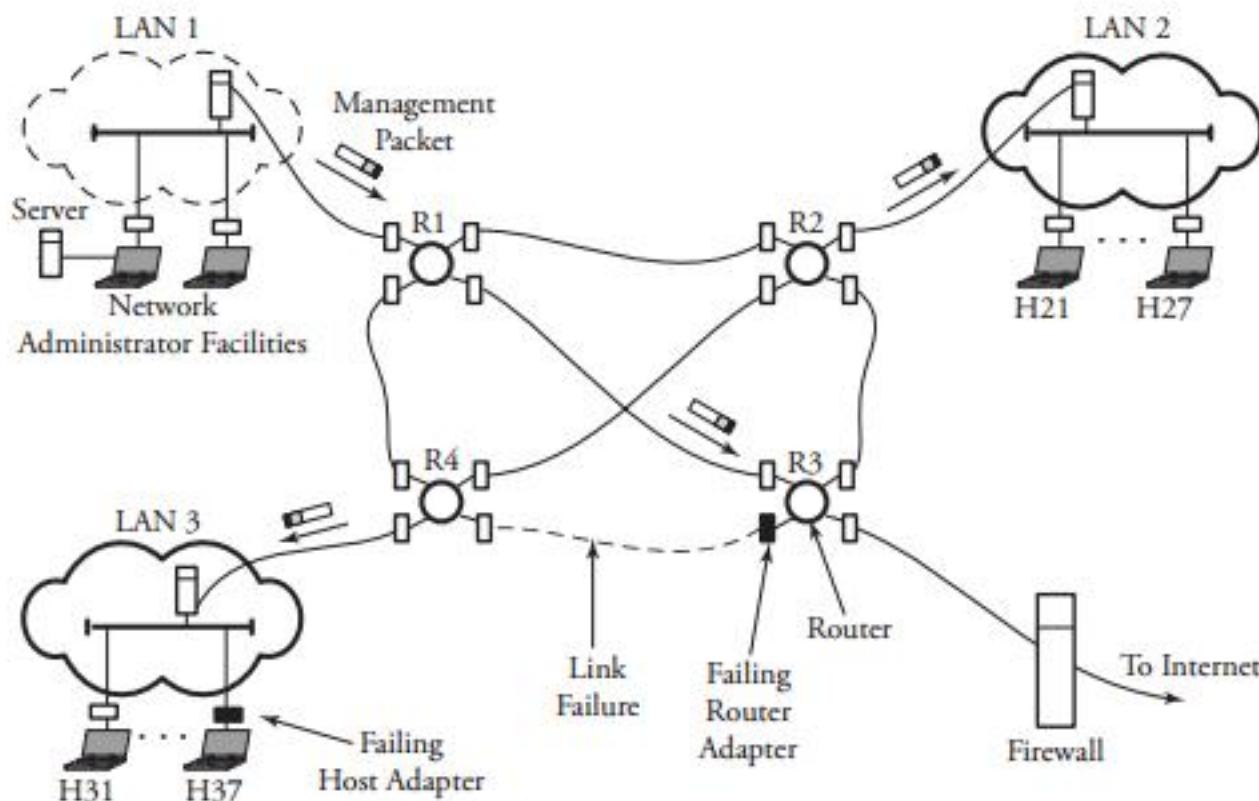


Figure 9.10 Simple network management in a scenario of LANs connecting to the Internet

Network management tasks can be characterized as follows:

- *QoS and performance management.* A network administrator periodically monitors and analyzes routers, hosts, and utilization of links and then redirect traffic flow to avoid any overloaded spots. Certain tools are available to detect rapid changes in traffic flow.
- *Network failure management.* Any fault in a network, such as link, host, or router hardware or software outages, must be detected, located, and responded to by the network. Typically, increased checksum errors in frames is an indication of possible error. Figure 9.10 shows adapter failures at router R3 and host H37; these failures can be detected through network management.
- *Configuration management.* This task involves tracking all the devices under management and ensuring that all devices are connected and operate properly. If there is an unexpected change in routing tables, a network administrator wants to discover the misconfigured spot and reconfigure the network before the error affects the network substantially.
- *Security management.* A network administrator is responsible for the security of its network. This task is handled mainly through firewalls, as discussed in Chapter 10.

A firewall can monitor and control access points. In such cases, the network administrator wants to know about any intrusion from a suspicious source to the network. For example, a host in a network can be attacked by receiving a large number of SYN packets.

- *Billing and accounting management.* The network administrator specifies user access or restrictions to network resources and issues all billing and charges, if any, to users.

Locating a failing point, such as an adapter failure at a host or a router, can be done by appropriate network management tools. Normally, a standard packet format is specified for network management.

9.7.1 Elements of Network Management

Network management has three main components: network management: a *managing center*, a *managed device*, and a *network management protocol*. The managing center consists of the network administrator and his or her facilities. Typically, the managing center comprises a substantial human network. A managed device is the network equipment, including its software, that is controlled by the managing center. Any hub, bridge, router, server, printer, or modem can be a managed device. The network management protocol is a policy between the managing center and the managed devices. The protocol in this context allows the managing center to obtain the status of managed devices. In network management, an *agent* is a managed device, such as a router, hub, or bridge. A *manager* is a network administrative device, as a management host. An agent can use the network management protocol to inform the managing center of an unexpected event.

9.7.2 Structure of Management Information (SMI)

The *structure of management information* (SMI) language is used to define the rules for naming objects and to encode objects in a managed network center. In other words, SMI is a language by which a specific instance of the data in a managed network center is defined. For example, Integer32 means a 32-bit integer with a value between -2^{31} and $-2^{31} - 1$. The SMI language also provides higher-level language constructs, which typically specify the data type, status, and semantics of managed objects containing the management data. For example, the STATUS clause specifies whether the object definition is current or obsolete, ipInDelivers defines a 32-bit counter to trace the number of IP datagrams received at a managed device and then received at an upper-layer protocol.

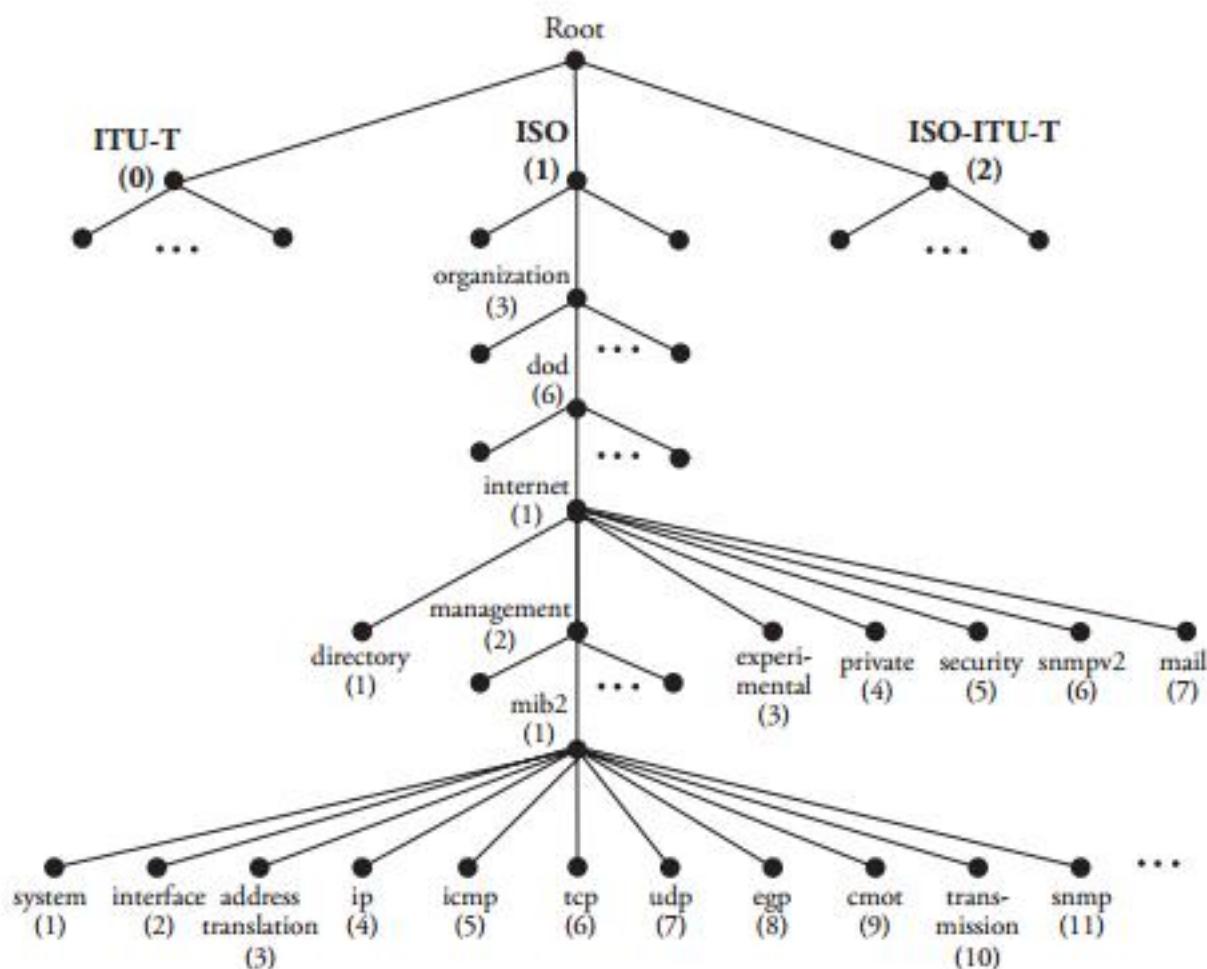


Figure 9.11 ASN.1 object identifier organized hierarchically

9.7.3 Management Information Base (MIB)

Management information base (MIB) is an information storage medium that contains managed objects reflecting the current status of the network. Because managed objects have associated pieces of information that are stored in a MIB, the MIB forms a collection of named objects, including their relationships to one another in a management center. The information pieces can be obtained by directing the managing center to do so.

Objects are organized in a hierarchical manner and are identified by the *abstract syntax notation one* (ASN.1) object definition language. The hierarchy of object names, known as *ASN.1 object identifier*, is an object identifier tree in which each branch has both a name and a number, as shown in Figure 9.11. Network management can then identify an object by a sequence of names or numbers from the root to that object.

On the root of the object identifier hierarchy are three entries: ISO (International Standardization Organization), ITU-T (International Telecommunication Union-Telecommunication) standardization sector, and ISO-ITU-T, the joint branch of these two organizations. Figure 9.11 shows only part of the hierarchy. Under the ISO entry are other branches. For example, the *organization* (3) branch is labeled sequentially from the root as 1.3. If we continue to follow the entries on this branch, we see a path over *dod* (6), *Internet* (1), *management* (2), *mib-2*(1), and *ip* (4). This path is identified by (1.3.6.1.2.1.4) to indicate all the labeled numbers from the root to the *ip* (4) entry. Besides that entry, MIB module represents a number of network interfaces and well-known Internet protocols at the bottom of this tree. This path clearly shows all the standards of “IP” associated with the “MIB-2” computer networking “management.”

9.7.4 Simple Network Management Protocol (SNMP)

The *Simple Network Management Protocol* (SNMP) is designed to monitor the performance of network protocols and devices. SNMP protocol data units (PDUs) can be carried in the payload of a UDP datagram, and so its delivery to a destination is not guaranteed. Managed devices, such as routers and hosts, are objects, and each object has a formal ASN.1 definition. For each object, MIB accommodates a database of information that describes its characteristics. With this protocol, a network manager can find the location of a fault. SNMP runs on top of UDP and uses client/server configurations. The commands of this protocol define how to query information from a server and forward information to a server or a client.

The task of SNMP is to transport MIB information among managing centers and agents executing on behalf of managing centers. For each managed MIB object, an SNMP request is used to retrieve or change its associated value. If an unsolicited message is received by an agent, or when an interface or device goes down, the protocol can also inform the managing center. The second version of this protocol, SNMPv2, runs on top of more protocols and has more messaging options, resulting in more effective network management. SNMPv3 has more security options.

SNMPv2 has seven PDUs, or messages, as follows.

1. GetRequest is used to obtain a MIB object value.
2. GetNextRequest is used to obtain the next value of a MIB object.
3. GetBulkRequest gets multiple values, equivalent to multiple GetRequests but without using multiple overheads.

Get or Set PDU

PDU Type	Request ID	Error Status	Error Index	Name	Value	...	Name	Value
----------	------------	--------------	-------------	------	-------	-----	------	-------

Trap PDU

PDU Type	Enterprise	Agent Address	Trap Type	Specific Code	Time Stamp	Name	Value	...	Name	Value
----- Header -----						----- Variable List -----				

Figure 9.12 SNMP PDU format

4. InformRequest is a manager-to-manager message that two communicating management centers are remote to each other.
5. SetRequest is used by a managing center to initiate the value of a MIB object.
6. Response is a reply message to a request-type PDU.
7. Trap notifies a managing center that an unexpected event has occurred.

Figure 9.12 shows the format of SNMP PDUs. Two types of PDUs are depicted: Get or Set and Trap. The Get or Set PDU format is as follows:

- *PDU type* indicates one of the seven PDU types.
- *Request ID* is an ID used to verify the response of a request. Thus, a managing center can detect lost requests or replies.
- *Error status* is used only by Response PDUs to indicate types of errors reported by an agent.
- *Error index* is a parameter indicating to a network administrator which *name* has caused an error.

If requests or replies are lost, SNMP does not mandate any method for retransmission. *Error status* and *Error index* fields are all zeros except for the one in a GetBulkRe-

quest PDU. Figure 9.12 also shows the format of the Trap PDU, whereby the *enterprise* field is for use in multiple networks; the *timestamp* field, for measuring up time; and the *agent address* field, for indicating that the address of the managed agent is included in the PDU header.

9.8 Summary

The top layer of the networking stack is known as the application layer. Services in this category allow users and programs to interact with services on remote machines and with remote users. The *application layer* offers network services and applications to users and runs certain applications. The *Domain Name System* (DNS) is a distributed hierarchical and global directory that translates machine or domain names to numerical IP addresses. An IP address can be assigned a *domain name*. Unique domain names assigned to hosts must be selected from a *name space* and are generally organized in a hierarchical fashion.

The TELNET remote login protocol uses a negotiation method to allow clients and servers to reconfigure the parameters controlling their interaction. The *Secure Shell* (SSH) remote login protocol is implemented on TCP for communications. SSH is more powerful and flexible than TELNET and allows the user to more easily execute a single command on a remote client.

The *Simple Mail Transfer Protocol* (SMTP) can transfer *e-mail* from the mail server of a source to mail servers of destinations. A user mailbox is a certain space the mail server allocates to the user to keep its e-mail. SMTP is designed to connect only the two mail servers of the associated parties, regardless of the distance between the two users. Consequently, this protocol involves only the two mail servers of the communicating users.

File transfer allows geographically distributed files and information to be shared among the members of a working group and for information sharing. A user can use a file transfer protocol to access a server and transfer the desired file. Two such protocols are FTP and SCP. The *World Wide Web* (WWW), or simply the *Web*, is a global network of servers linked together by a common protocol allowing access to all connected hypertext resources. HTTP requests are first directed to the network *proxy server* called *Web cache*. Once configured by the network, a browser's request for an object is directed to the Web cache.

The chapter ended with network management aspects of computer networks. Managed devices, such as routers and hosts, are managed objects, and each object has a formal ASN.1 definition. Another tool through which a database of information

and characteristics for objects can be accommodated is MIB. With SNMP, a network manager can find the location of a fault. SNMP runs on top of UDP and uses client/server configurations. SNMP commands define how to query information from a server and forward information to a server or a client.

The next chapter discusses the security aspects of computer networking. Types of network attacks, message encryption protocols, and message authentication techniques are covered.

9.9 Exercises

1. Find out about an IP address that is well known to you, such as that of your college or your work server. Set up an experiment in which you look up the domain name for this IP address.
2. Consider DNS servers.
 - (a) Compare two approaches, obtaining a name from a file in a remote machine and from a DNS server of the local ISP.
 - (b) Describe the relationship between a domain name taken from a DNS server and an IP address subnet.
 - (c) In part (b), do all hosts on the same subnet need to be identified by the same DNS server? Why?
3. Read all the RFCs associated with TELNET and SSH listed in Appendix B.
 - (a) What are the most significant differences between these two protocols?
 - (b) Compare the two protocols on the functionality given by “rlogin.”
4. Read all the RFCs associated with FTP listed in Appendix B.
 - (a) Does FTP compute any checksum for files it transfers? Why?
 - (b) FTP is based on TCP. Explain the status of file transfer if the TCP connection is shut down, and compare it with the same situation in which its control connection remains active.
 - (c) Find all the commands from RFCs set for clients.
5. Set up an experiment in which two computers across a defined network exchange the same defined file through FTP. Measure the file transfer delay
 - (a) On both directions when the network is in its best traffic state
 - (b) On both directions when the network is in its worst traffic state
 - (c) On one direction when you try FTP from one of the computers to itself

6. Do research on which letters, such as A, B, C, . . . , or signs, such as #, *, •, . . . , are not allowed to be sent on a URL to a Web server.
7. Read all the RFCs associated with HTTP listed in Appendix B.
 - (a) What is the purpose of GET command?
 - (b) What is the purpose of PUT command?
 - (c) Following part (a), explain why a GET command needs to use the name of the contacted server when it is applied.
8. Figure 9.11 shows the hierarchical ASN.1 object identifier of network management.
 - (a) Explain the role of ASN.1 in the protocol reference model, discussing it on either the five- or seven-layer protocol model.
 - (b) Find the impact of constructing a grand set of unique global ASN.1 names for MIB variables.
 - (c) Do research in the related RFCs listed in Appendix B, and find out where in the ASN.1 hierarchy a U.S. based organization must register its own developed MIB.
9. Consider the SNMP network management environment.
 - (a) Justify why UDP is preferable as TCP for this environment.
 - (b) MIB variables can be read in a local router the MIB belongs to. What would be the pros and cons if we let all managing centers access MIBs in all routers?
 - (c) Should MIB variables be organized in the local router memory the same way that SNMP expresses? Why?
10. *Computer simulation project.* Write a computer program to simulate a TCP connection for a client/server combination. The server receives a small file from the client. Set up another machine as the proxy server, and try to send messages from the proxy server to the main server. Display all possible messages in the server.

This page intentionally left blank

CHAPTER 10

Network Security

The invention of computers made the need for security of digital information a critical issue. Advances in the field of computer networks have made information security even more important. Computer systems have to be equipped with mechanisms for securing data. Computer networks need provisions that secure data from possible intrusions. Security is especially crucial in wireless networks, as a wireless medium, by its nature, is vulnerable to intrusions and attacks. This chapter focuses on *network security*. The following major topics in network security are covered:

- *Overview of network security: elements and threats*
- *Overview of security methods*
- *Secret-key encryption protocols*
- *Public-key encryption protocols*
- *Authentication: message digest and digital signatures*
- *Security of IP and wireless networks*
- *Firewalls*

We begin by identifying and classifying types of network threats, hackers, and attacks, including DNS hacking attacks and router attacks. Network security can be divided into two broad categories: *cryptographic techniques* and *authentication techniques* (verification). Both secret-key and public-key encryption protocols are presented. We

then discuss message authentication and digital signature methods, through which a receiver can be assured that an incoming message is from whom it says it is.

We then consider several standardized security techniques, such as IPsec, and the security of wireless networks and IEEE 802.11 standards, as well as firewalls, or devices designed to protect a network. Finally, we present a case study on the security of wireless networks.

10.1 Overview of Network Security

Network security is a top-priority issue in data networks. As communication networks are growing rapidly, security issues have pushed to the forefront of concern for end users, administrators, and equipment suppliers. Despite enormous joint efforts by various groups to develop effective security solutions for networks, hackers continue to pose new, serious threats by taking advantage of weaknesses present in the Internet infrastructure.

10.1.1 Elements of Network Security

Network security is concerned mainly with the following two elements:

1. *Confidentiality.* Information should be available only to those who have rightful access to it.
2. *Authenticity and integrity.* The sender of a message and the message itself should be verified at the receiving point.

In Figure 10.1, user 1 sends a message (“I am user 1”) to user 2. In part (a) of the figure, the network lacks any security system, so an intruder can receive the message, change its content to a different message (“Hi! I am user 1”) and send it to user 2. User 2 may not know that this falsified message is really from user 1 (authentication) and that the content of the message is what user 1 (confidentiality). In part (b) of the figure, a security block is added to each side of the communication, and a secret key that only users 1 and 2 would know about is included. Therefore, the message is changed to a form that cannot be altered by the intruder, who would be disabled in this communication transaction.

In general, no protocol or network architecture can ensure full security. Internet routing is based on a distributed system of many routers, switches, and protocols. These protocols have a number of points of vulnerabilities that can be exploited to cause such

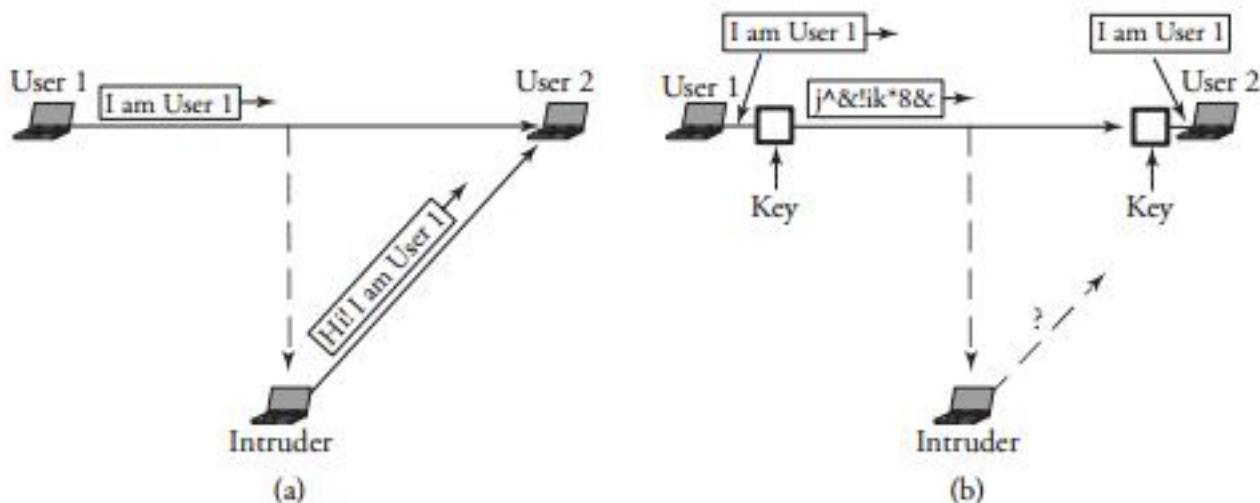


Figure 10.1 (a) Message content and sender identity falsified by intruder; (b) a method of applied security

problems as misdelivery or nondelivery of user traffic, misuse of network resources, network congestion and packet delays, and the violation of local routing policies.

10.1.2 Threats to Network Security

Internet infrastructure *attacks* are broadly classified into four categories, as follows:

1. DNS hacking
2. Routing table poisoning
3. Packet mistreatment
4. Denial of service

Among these threats, the first three attacks are related to network infrastructure; *denial-of-service attacks* are related to end systems.

DNS Hacking Attacks

As mentioned in Chapter 9, the *Domain Name System* (DNS) server is a distributed hierarchical and global directory that translates domain names into numerical IP address. DNS is a critical infrastructure, and all hosts contact DNS to access servers and start connections. In the normal mode of operation, hosts send UDP queries to the DNS server. Servers reply with a proper answer, or direct the queries to smarter servers. A DNS server also stores information other than host addresses.

Name-resolution services in the modern Internet environment are essential for e-mail transmission, navigation to Web sites, or data transfer. Thus, an attack on DNS

can potentially affect a large portion of the Internet. A DNS *hacking attack* may result in the lack of data authenticity and integrity and can appear in any of the following forms:

1. An *information-level attack* forces a server to correspond with other than the correct answer. With cache poisoning, a hacker tricks a remote name server into caching the answer for a third-party domain by providing malicious information for the domain's authorized servers. Hackers can then redirect traffic to a preselected site.
2. In a *masquerading attack*, the adversary poses as a trusted entity and obtains all the secret information. In this guise, the attacker can stop any message from being transmitted further or can change the content or redirect the packet to bogus servers. This action is also known as a *middle-man attack*.
3. The attacker normally sends queries to each host and receives in reply the DNS host name. In an *information leakage attack*, the attacker sends queries to all hosts and identifies which IP addresses are not used. Later on, the intruder can use those IP addresses to make other types of attacks.
4. Once a domain name is selected, it has to be registered. Various tools are available to register domain names over the Internet. If the tools are not smart enough, an invader might obtain secure information and use it to highjack the domain later. In the *domain highjacking attack*, whenever a user enters a domain address, she/he is forced to enter into the attacker's Web site. This can be very irritating and can cause a great loss of Internet usage ability.

Routing Table Poisoning Attacks

A *routing table poisoning attack* is the undesired modification of routing tables. An attacker can do this by maliciously modifying the routing information update packets sent by routers. This is a challenging and important problem, as a routing table is the basis of routing in the Internet. Any false entry in a routing table could lead to significant consequences, such as congestion, an overwhelmed host, looping, illegal access to data, and network partition. Two types of routing table poisoning attacks are the *link attack* and the *router attack*.

A *link attack* occurs when a hacker gets access to a link and thereby intercepts, interrupts, or modifies routing messages on packets. Link attacks act similarly on both the link-state and the distance-vector protocols discussed in Chapter 7. If an attacker succeeds in placing an attack in a link-state routing protocol, a router may send incorrect updates about its neighbors or remain silent even if the link state of its neighbor has changed. The attack through a link can be so severe that the attacker can program a

router to either drop packets from a victim or readdress packets to a victim, resulting in a lower throughput of the network. Sometimes, a router can stop an intended packet from being forwarded further. However, since more than one path to any destination exists, the packet ultimately reaches its destination.

Router attacks may affect the link-state protocol or even the distance-vector protocol. If link-state protocol routers are attacked, they become malicious. They may add a nonexisting link to a routing table, delete an existing link, or even change the cost of a link. This attack may cause a router to simply ignore the updates sent by its neighbors, leading to a serious impact on the operability of the network traffic flow.

In the distance-vector protocol, an attacker may cause routers to send wrong updates about any node in the network, thereby misleading a router and resulting in network problems.

Most unprotected routers have no way to validate updates. Therefore, both link-state and distance-vector router attacks are very effective. In the distance-vector protocol, for example, a malicious router can send wrong information in the form of a distance vector to all its neighbors. A neighbor may not be able to detect this kind of attack and thus proceeds to update its routing table, based on wrong distance vectors. The error can in turn be propagated to a great portion of the network before being detected.

Packet-Mistreatment Attacks

A *packet-mistreatment attack* can occur during any data transmission. A hacker may capture certain data packets and mistreat them. This type of attack is very difficult to detect. The attack may result in congestion, lowering throughput, and denial-of-service attacks. Similar to routing table poisoning attacks, packet-mistreatment attacks can also be subclassified into *link attacks* and *router attacks*. The link attack causes interruption, modification, or replication of data packets. A router attack can misroute all packets and may result in congestion or denial of service. Following are some examples of a packet-mistreatment attack:

- *Interruption.* If an attacker intercepts packets, they may not be allowed to be propagated to their destinations, resulting in a lower throughput of the network. This kind of attack cannot be detected easily, as even in normal operations, routers can drop some packets, for various reasons.
- *Modification.* Attackers may succeed in accessing the content of a packet while in transit and change its content. They can then change the address of the packet or

even change its data. To solve this kind of problem, a digital signature mechanism, discussed later in this chapter, can be used.

- *Replication.* An attacker might trap a packet and replay it. This kind of attack can be detected by using the sequence number for each packet.
- *Ping of death.* An attacker may send a *ping message*, which is large and therefore must be fragmented for transport. The receiver then starts to reassemble the fragments as the ping fragments arrive. The total packet length becomes too large and might cause a system crash.
- *Malicious misrouting of packets.* A hacker may attack a router and change its routing table, resulting in misrouting of data packets, causing a denial of service.

Denial-of-Service Attacks

A *denial-of-service attack* is a type of security breach that prohibits a user from accessing normally provided services. The denial of service does not result in information theft or any kind of information loss but can nonetheless be very dangerous, as it can cost the target person a large amount of time and money. Denial-of-service attacks affect the destination rather than a data packet or router.

Usually, a denial-of-service attack affects a specific network service, such as e-mail or DNS. For example, such an attack may overwhelm the DNS server in various ways and make it inoperable. One way of initiating this attack is by causing buffer overflow. Inserting an executable code inside memory can potentially cause a buffer overflow. Or, an adversary may use various tools to send large numbers of queries to a DNS server, which then is not able to provide services in a timely manner.

Denial-of-service attacks are easy to generate but difficult to detect. They take important servers out of action for few hours, thereby denying service to all users. There are yet a few other situations that can cause this kind of attack, such as UDP flood, a TCP flood and ICMP flood. In all these attacks, the hacker's main aim is to overwhelm victims and disrupt services provided to them.

Denial-of-service attacks are two types:

1. *Single-source.* An attacker sends a large number of packets to a target system to overwhelm and disable it. These packets are designed such that their real sources cannot be identified.
2. *Distributed.* In this type of attack, a large number of hosts are used to flood unwanted traffic to a single target. The target cannot then be accessible to other users in the network, as it is processing the flood of traffic.

The flood may be either a UDP flood or a TCP SYN flood. UDP flooding is used against two target systems and can stop the services offered by either system. Hackers link the UDP character-generating services of a system to another one by sending UDP packets with spoofed return addresses. This may create an infinite looping between the two systems, leading to system uselessness.

Normally, a SYN packet is sent by a host to a user who intends to establish a connection. The user then sends back an acknowledgment. In the TCP SYN flood, a hacker sends a large number of SYN packets to a target user. Since the return addresses are spoofed, the target user queues up a SYN/ACK packet and never processes it. Therefore, the target system keeps on waiting. The result may be a hard disk crash or reboot.

10.2 Overview of Security Methods

Common solutions that can protect computer communication networks from attacks are classified as *cryptographic techniques* or *authentication techniques* (verification).

10.2.1 Cryptographic Techniques

Cryptography has a long and fascinating history. Centuries ago, cryptography was used as a tool to protect national secrets and strategies. Today, network engineers focus on *cryptography* methods for computer communication networks. Cryptography is the process of transforming a piece of information or message shared by two parties into some sort of code. The message is scrambled before transmission so that it is undetectable by outside watchers. This kind of message needs to be decoded at the receiving end before any further processing.

The main tool that network security experts are using to encrypt a message M is a secret key K ; the fundamental operation often used to encrypt a message is the Exclusive-OR (\oplus). Suppose that we have one bit, M , and a secret bit, K . A simple encryption is carried out using $M \oplus K$. To decrypt this message, the second party—if he/she has the key, K —can easily detect M by performing the following:

$$(M \oplus K) \oplus K = M. \quad (10.1)$$

In computer communication networks, data can travel between two users while it is encrypted. In Figure 10.2, two servers are exchanging data while two types of encryption devices are installed in their communication network. The first encryption device is *end-to-end encryption*, whereby secret coding is carried out at both end systems.

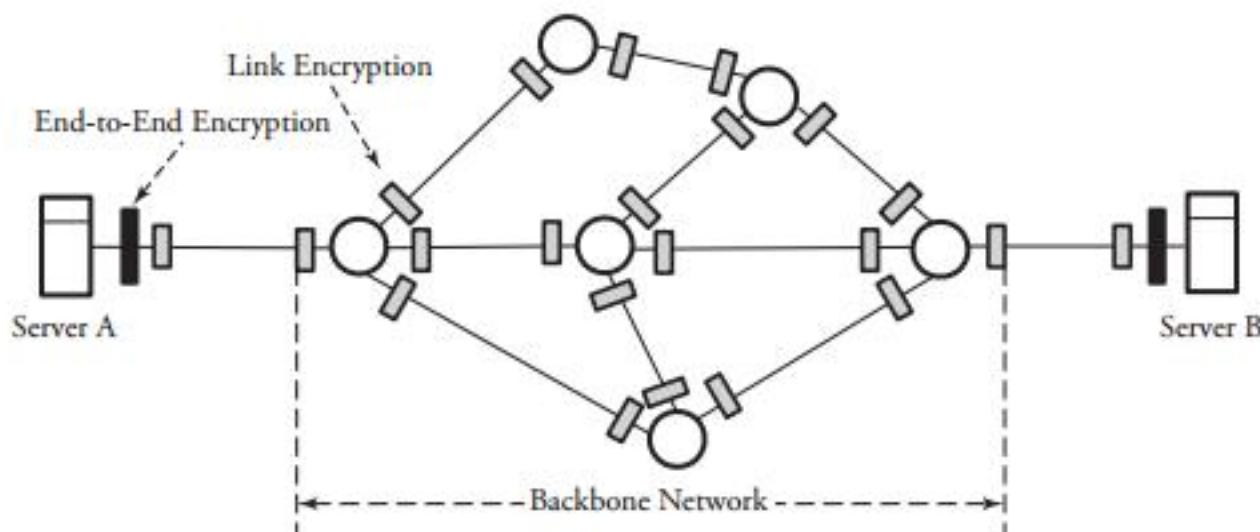


Figure 10.2 Overview of encryption points in a communication network

In this figure, server *A* encodes its data, which can be decoded only at the other end server. The second phase of encryption is *link encryption*, which secures all the traffic passing over that link.

The two types of encryption techniques are *secret-key encryption* and *public-key encryption*. In a secret-key model, both sender and receiver conventionally use the same key for an encryption process. In a public-key model, a sender and a receiver each use a different key. The public-key system is more powerful than the secret-key system and provides better security and message privacy. But the biggest drawback of public-key encryption is speed. The public-key system is significantly more complex computationally and may not be practical in many cases. Hence, the public-key system is used only to establish a session to exchange a session key. Then, this session key is used in a secret-key system for encrypting messages for the duration of the session.

10.2.2 Authentication Techniques

Encryption methods offer the assurance of message confidentiality. However, a networking system must be able to verify the authenticity of the message and the sender of the message. These forms of security techniques in computer networks are known as *authentication techniques* and are categorized as *authentication with message digest* and *authentication with digital signature*. Message authentication protects a user in a network

against data falsification and ensures data integrity. These methods do not necessarily use keys.

10.3 Secret-Key Encryption Protocols

Secret-key encryption protocols, sometimes known as *symmetric encryption*, or *single-key encryption* protocols, are conventional encryption models. They typically consist of an encryption algorithm, a key, and a decryption algorithm. At the end point, the encrypted message is called *ciphertext*. Several standard mechanisms can be used to implement a secret-key encryption algorithm. Here, we focus on two protocols: *Data Encryption Standard* (DES) and *Advanced Encryption Standard* (AES).

In these algorithms, a shared secret key between a transmitter and a receiver is assigned at the transmitter and receiver points. The encryption algorithm produces a different key at any time for a specific transmission. Changing the key changes the output of the algorithm. At the receiving end, the encrypted information can be transformed back to the original data by using a decryption algorithm and the same key that was used for encryption. The security of conventional encryption depends on the secrecy of the key, not on the secrecy of the encryption algorithm. Consequently, the algorithm need not be kept secret; only the key has to be secret.

10.3.1 Data Encryption Standard (DES)

With the *Data Encryption Standard* (DES), plaintext messages are converted into 64-bit blocks, each encrypted using a key. The key length is 64 bits but contains only 56 usable bits; thus, the last bit of each 8 byte in the key is a parity bit for the corresponding byte. DES consists of 16 identical rounds of an operation, as shown in Figure 10.3. The details of the algorithm on each 64-bit block of message at each round i of operation are as follows.

Begin DES Algorithm

1. **Initialize.** Before round 1 begins, all 64 bits of an incoming message and all 56 bits of the secret key are separately permuted (shuffled).
2. Each incoming 64-bit message is broken into two 32-bit halves denoted by L_i and R_i , respectively .
3. The 56 bits of the key are also broken into two 28-bit halves, and each half is rotated one or two bit positions, depending on the round.
4. All 56 bits of the key are permuted, producing version k_i of the key on round i .

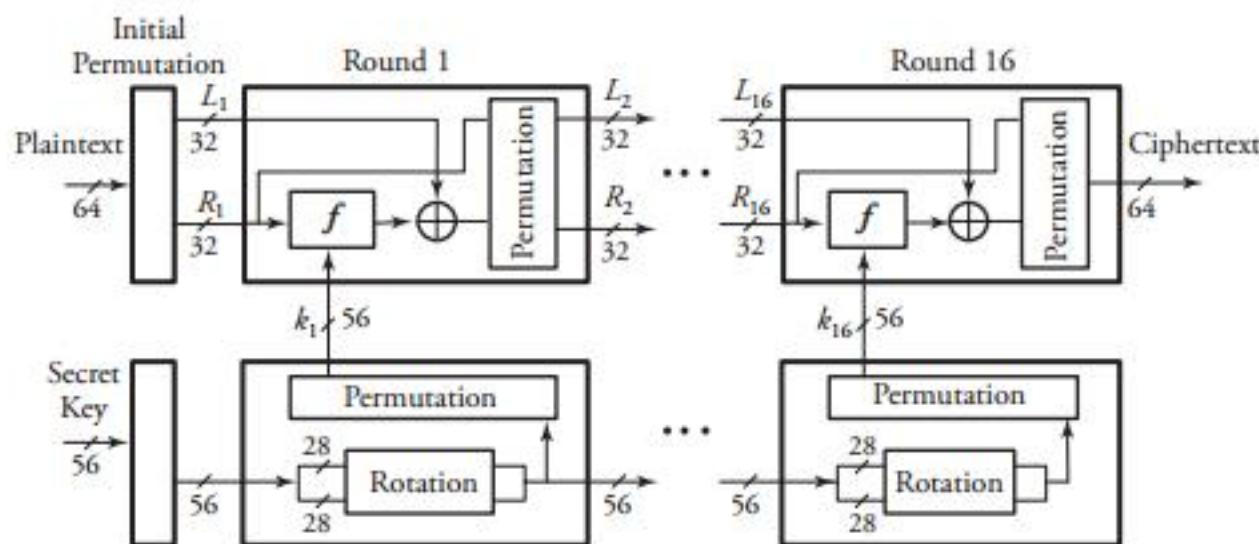


Figure 10.3 The Data Encryption Standard (DES)

5. In this step, \oplus is a logic Exclusive-OR, and the description of function $F()$ appears next. Then, L_i and R_i are determined by

$$L_i = R_{i-1} \quad (10.2)$$

and

$$R_i = L_{i-1} \oplus F(R_{i-1}, k_i). \quad (10.3)$$

6. All 64 bits of a message are permuted. ■

The operation of function $F()$ at any round i of DES is as follows.

1. Out of 52 bits of k_i , function $F()$ chooses 48 bits.
2. The 32-bit R_{i-1} is expanded from 32 bits to 48 bits so that it can be combined with 48-bit k_i . The expansion of R_{i-1} is carried out by first breaking R_{i-1} into eight 4-bit chunks and then expanding each chunk by copying the leftmost bit and the rightmost bit from left and right adjacent chunks, respectively.
3. Function $F()$ also partitions the 48 bits of k_i into eight 6-bit chunks.
4. The corresponding eight chunks of R_{i-1} and eight chunks of k_i are combined as follows:

$$R_{i-1} = R_{i-1} \oplus k_i. \quad (10.4)$$

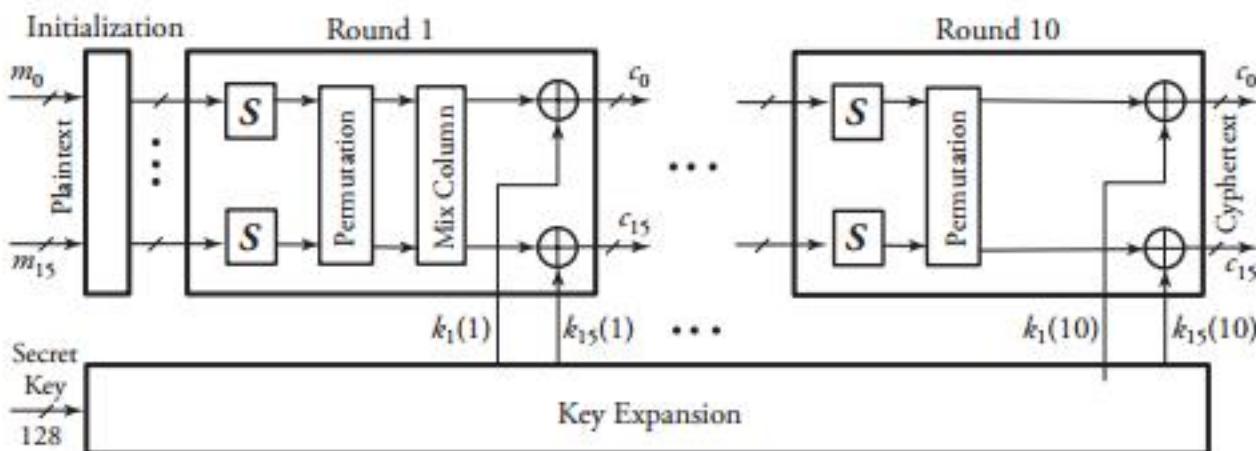


Figure 10.4 Overview of Advanced Encryption Standard (AES) protocol

At the receiver, the same steps and the same key are used to reverse the encryption. It is now apparent that the 56-bit key length may not be sufficient to provide full security. This argument is still controversial. Triple DES provides a solution for this controversy: three keys are used, for a total of 168 bits. It should also be mentioned that DES can be implemented more efficiently in hardware than in software.

10.3.2 Advanced Encryption Standard (AES)

The *Advanced Encryption Standard* (AES) protocol has a better security strength than DES. AES supports 128-bit symmetric block messages and uses 128-, 192-, or 256-bit keys. The number of rounds in AES is variable from 10 to 14 rounds, depending on the key and block sizes. Figure 10.4 illustrates the encryption overview of this protocol, using a 128-bit key. There are ten rounds of encryptions for the key size of 128 bits. All rounds are identical except for the last round, which has no mix-column stage.

A single block of 128-bit plaintext (16 bytes) as an input arrives from the left. The plaintext is formed as 16 bytes m_0 through m_{15} and is fed into round 1 after an initialization stage. In this round, substitute units—indicated by S in the figure—perform a byte-by-byte substitution of blocks. The ciphers, in the form of rows and columns, move through a *permutation stage* to shift rows to mix columns. At the end of this round, all 16 blocks of ciphers are Exclusive-ORed with the 16 bytes of round 1 key $k_0(1)$ through $k_{15}(1)$. The 128-bit key is expanded for ten rounds. The AES *decryption algorithm* is fairly simple and is basically the reverse of the encryption algorithm at each stage of a round. All stages of each round are reversible.

10.4 Public-Key Encryption Protocols

The introduction of *public-key encryption* brought a revolution to the field of cryptography. Public-key cryptography provided a very clever method for key exchange. In the public-key encryption model, a sender/receiver pair use different keys. This model is sometimes known as *asymmetric*, or *two-key, encryption*.

Public-key algorithm is based on mathematical functions rather than on substitution or permutation, although the security of any encryption scheme indeed depends on the length of the key and the computational work involved in breaking an encrypted message. Several public-key encryption protocols can be implemented. Among them, the following two protocols are the focus of our study:

- Rivert, Shamir, and Aldeman (RSA) protocol
- Diffie-Hillman key-exchange protocol.

In the public-key encryption methods, either of the two related keys can be used for encryption; the other one, for decryption. It is computationally infeasible to determine the decryption key given only the algorithm and the encryption key. Each system using this encryption method generates a pair of keys to be used for encryption and decryption of a message that it will receive. Each system publishes its encryption key by placing it in a public register or file and sorts out the key as a public one.

The companion key is kept private. If A wishes to send a message to B , A encrypts the message by using B 's public key. On receiving the message, B decrypts it with the private B key. No other recipients can decrypt the message, since only B knows its private key. This way, public-key encryption secures an incoming communication as long as a system controls its private key. However, public-key encryption has extra computational overhead and is more complex than the conventional one.

10.4.1 RSA Algorithm

Rivert, Shamir, and Aldeman developed the RSA public-key encryption and signature scheme. This was the first practical public-key encryption algorithm. RSA is based on the intractability of factoring large integers. Assume that a plaintext m must be encrypted to a ciphertext c . The RSA algorithm has three phases for this: *key generation*, *encryption*, and *decryption*.

Key Generation

In the RSA scheme, the key length is typically 512 bits, which requires an enormous computational power. A plaintext is encrypted in blocks, with each block having a

binary value less than some number n . Encryption and decryption are done as follows, beginning with the generation of a public key and a private key.

Begin Key Generation Algorithm

1. Choose two roughly 256-bit prime numbers, a and b , and derive $n = ab$. (A number is prime if it has factors of 1 and itself.)
2. Find x . Select encryption key x such that x and $(a - 1)(b - 1)$ are relatively prime. (Two numbers are relatively prime if they have no common factor greater than 1.)
3. Find y . Calculate decryption key y :

$$xy \bmod (a - 1)(b - 1) = 1. \quad (10.5)$$

4. At this point, a and b can be discarded.
5. The public key = $\{x, n\}$.
6. The private key = $\{y, n\}$. ■

In this algorithm, x and n are known to both sender and receiver, but only the receiver must know y . Also, a and b must be large and about the same size and both greater than 1,024 bits. The larger these two values, the more secure the encryption.

Encryption

Both sender and receiver must know the value of n . The sender knows the value of x , and only the receiver knows the value of y . Thus, this is a public-key encryption, with the public key $\{x, n\}$ and the private key $\{y, n\}$. Given $m < n$, ciphertext c is constructed by

$$c = m^x \bmod n. \quad (10.6)$$

Note here that if a and b are chosen to be on the order of 1,024 bits, $n \approx 2,048$. Thus, we are not able to encrypt a message longer than 256 characters.

Decryption

Given the ciphertext, c , the plaintext, m , is extracted by

$$m = c^y \bmod n. \quad (10.7)$$

In reality, the calculations require a math library, as numbers are typically huge. One can see easily how Equations (10.6) and (10.7) work.

Example. For an RSA encryption of a 4-bit message of 1,000, or $m = 9$, we choose $a = 3$ and $b = 11$. Find the public and the private keys for this security action, and show the ciphertext.

Solution. Clearly, $n = ab = 33$. We select $x = 3$, which is relatively prime to $(a - 1)(b - 1) = 20$. Then, from $xy \bmod (a - 1)(b - 1) = 3y \bmod 20 = 1$, we can get $y = 7$. Consequently, the public key and the private key should be $\{3, 33\}$ and $\{7, 33\}$, respectively. If we encrypt the message, we get $c = m^x \bmod n = 9^3 \bmod 33 = 3$. The decryption process is the reverse of this action, as $m = c^y \bmod n = 3^7 \bmod 33 = 9$.

10.4.2 Diffie-Hillman Key-Exchange Protocol

In the *Diffie-Hillman key-exchange* protocol, two end users can agree on a shared secret code without any information shared in advance. Thus, intruders would not be able to access the transmitted communication between the two users or discover the shared secret code. This protocol is normally used for *virtual private networks* (VPNs), explained in Chapter 16. The essence of this protocol for two users, 1 and 2, is as follows. Suppose that user 1 selects a prime a , a random integer number x_1 , and a generator g and creates $y_1 \in \{1, 2, \dots, a - 1\}$ such that

$$y_1 = g^{x_1} \bmod a. \quad (10.8)$$

In practice, the two end users agree on a and g ahead of time. User 2 performs the same function and creates y_2 :

$$y_2 = g^{x_2} \bmod a. \quad (10.9)$$

User 1 then sends y_1 to user 2. Now, user 1 forms its key, k_1 , using the information its partner sent as

$$k_1 = y_2^{x_1} \bmod a, \quad (10.10)$$

and user 2 forms its key, k_2 , using the information its partner sent it as

$$k_2 = y_1^{x_2} \bmod a. \quad (10.11)$$

It can easily be proved that the two Keys k_1 and k_2 are equal. Therefore, the two users can now encrypt their messages, each using its own key created by the other one's information.

10.5 Authentication

Authentication techniques are used to verify identity. Message authentication verifies the authenticity of both the message content and the message sender. Message content is authenticated through implementation of a hash function and encryption of the resulting message digest. The sender's authenticity can be implemented by use of a digital signature.

A common technique for authenticating a message is to implement a *hash function*, which is used to produce a “fingerprint” of a message. The hash value is added at the end of message before transmission. The receiver recomputes the hash value from the received message and compares it to the received hash value. If the two hash values are the same, the message was not altered during transmission. Once a hash function is applied on a message, m , the result is known as a message digest, or $h(m)$. The hash function has the following properties.

- Unlike the encryption algorithm, the authentication algorithm is not required to be reversible.
- Given a message digest $h(m)$, it is computationally infeasible to find m .
- It is computationally infeasible to find two different messages m_1 and m_2 such that $h(m_1) = h(m_2)$.

Message authentication can be implemented by two methods. In the first method, as shown in Figure 10.5 (a), a hash function is applied on a message, and then a process of encryption is implemented. Thus, a message digest can also be encrypted in this method. At this stage, the encryption can be a public key or a secret key. The authenticity of a message in this method is assured only if the sender and the receiver share the encryption key. At the receiver site, the receiving user 2 has to decrypt the received message digest and compare it with the one made locally at its site for any judgments on the integrity of the message.

In the second method, as shown in Figure 10.5 (b), no encryption is involved in the process of message authentication. This method assumes that the two parties share a secret key. Hence, at the receiving site, the comparison is made between the received $h(m)$ and the message digest made locally from the received message. This technique is more popular in the security infrastructure of the Internet Protocol. Among the message authentication protocols are the MD5 *hash algorithm* and the *Secure Hash Algorithm* (SHA). SHA is the focus of our discussion.

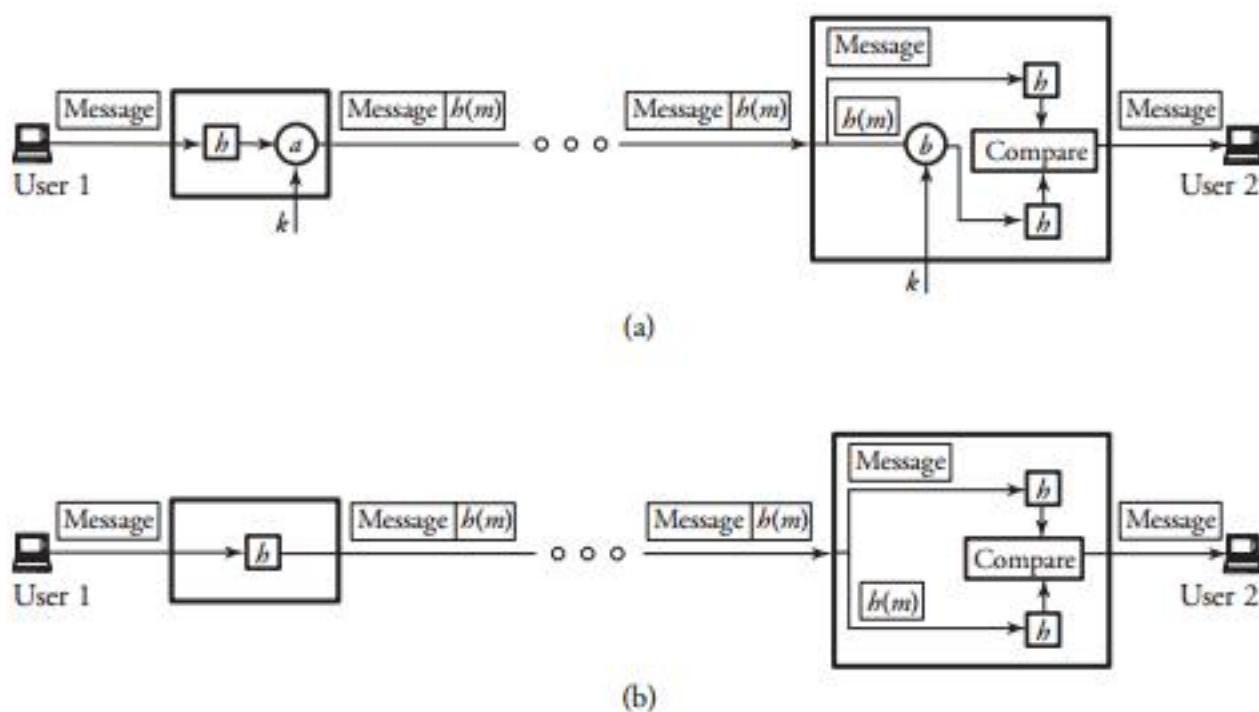


Figure 10.5 Message authentication: (a) combined with encryption; (b) use of the hash function

10.5.1 Secure Hash Algorithm (SHA)

The *Secure Hash Algorithm* (SHA) was proposed as part of the digital signature standard. SHA-1, the first version of this standard, takes messages with a maximum length of 2^{24} and produces a 160-bit digest. With this algorithm, SHA-1 uses five registers, R_1 through R_5 , to maintain a “state” of 20 bytes.

The first step is to pad a message m with length ℓ_m . The message length is forced to $\ell_m = 448 \bmod 512$. In other words, the length of the padded message becomes 64 bits less than the multiple of 512 bits. The number of padding bits can be as low as 1 bit and as high as 512 bits. The padding includes a 1 bit and as many 0 bits as required. Therefore, the least-significant 64 bits of the message length are appended to convert the padded message to a word with a multiple of 512 bits.

After padding, the second step is to expand each block of 512-bit (16 32 bits) words $\{m_0, m_1, \dots, m_{15}\}$ to words of 80 32 bits using:

$$w_i = m_i \text{ for } 0 \leq i \leq 15 \quad (10.12)$$

and

$$w_i = w_{i-3} \oplus w_{i-8} \oplus w_{i-14} \oplus w_{i-16} \leftrightarrow 1 \text{ for } 16 \leq i \leq 79, \quad (10.13)$$

where $\leftrightarrow j$ means left rotation by j bits. This way, bits are shifted several times if the incoming block is mixed with the state. Next, bits from each block of w_i are mixed into the state in four steps, each maintaining 20 rounds. For any values of a , b , and c , and bit number i , we define a function $F_i(a, b, c)$ as follows:

$$F_i(a, b, c) = \begin{cases} (a \cap b) \cup (\bar{a} \cap c) & 0 \leq i \leq 19 \\ a \oplus b \oplus c & 20 \leq i \leq 39 \\ (a \cap b) \cup (a \cap c) \cup (b \cap c) & 40 \leq i \leq 59 \\ a \oplus b \oplus c & 60 \leq i \leq 79 \end{cases}. \quad (10.14)$$

Then, the 80 steps ($i = 0, 1, 2, \dots, 79$) of the four rounds are described as follows:

$$\delta = (R_1 \leftrightarrow 5) + F_i(R_2, R_3, R_4) + R_5 + w_i + C_i \quad (10.15)$$

$$R_5 = R_4 \quad (10.16)$$

$$R_4 = R_3 \quad (10.17)$$

$$R_3 = R_2 \leftrightarrow 30 \quad (10.18)$$

$$R_2 = R_1 \quad (10.19)$$

$$R_1 = \delta, \quad (10.20)$$

where C_i is a constant value specified by the standard for round i . The message digest is produced by concatenation of the values in R_1 through R_5 .

10.6 Authentication and Digital Signature

A *digital signature* is one of the most important required security measures. Much like a person's signature on a document, a digital signature on a message is required for the authentication and identification of the right sender. The digital signature is supposed to be unique to an individual and serves as a means of identifying the sender. An electronic signature is not as easy as it was with the paper-based system. The digital signature requires a great deal of study, research, and skill. Even if these requirements are met, there is no guarantee that all the security requirements have been met.

The technical method of providing a sender's authentication is performed through cryptography. Many cryptographic mechanisms have been developed. Among them, the RSA algorithm implements both encryption and digital signature. When RSA is applied, the message is encrypted with the sender's private key. Thus, the entire encrypted message serves as a digital signature. This means that at the receiving end, the receiver can decrypt it, using the public key. This authenticates that the packet comes from the right user.

10.7 Security of IP and Wireless Networks

This section presents a case study of some of the security policies adopted for the Internet Protocol (IP) and basic wireless technologies. We start with IPsec, a standard for the IP layer.

10.7.1 IP Security and IPsec

Between any two users with a TCP/IP connection are multiple secure layers of security. A number of fields are appended to an IP packet when it is ready to undergo the security implementation. IP *security* (IPsec) is a set of protocols developed by the *Internet Engineering Task Force* (IETF) to support the secure exchange of packets at the IP layer. Figure 10.6 illustrates an encrypted and authenticated IP packet. An IPsec authentication header has the following fields.

- *Security parameters index* expresses a one-way relationship between two communicating users that can accept security.
- *Sequence number* is an increasing counter number.
- *Payload data* is an encryption-protected upper-layer segment.
- *Padding* is used if the plaintext is required by the encryption algorithm to be a certain multiple of 1 byte.
- *Pad length* specifies the number of bytes for the padding.
- *Next header* indicates the type of next header attached.
- *Authentication data* provides the integrity check value.

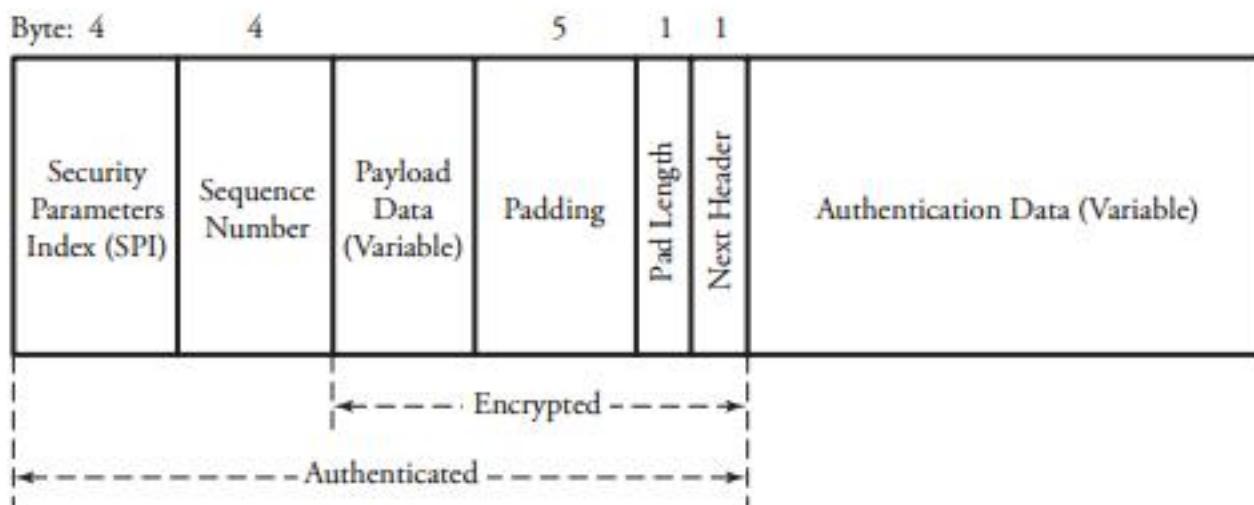


Figure 10.6 IPsec authentication header format

IPsec provides enhanced security features, such as better encryption algorithms and more comprehensive authentication. In order for IPsec to work, both sender and receiver must exchange public encryption keys. IPsec has two encryption modes: tunnel and transport. Tunnel mode encrypts the header and the payload of each packet; the transport mode encrypts the payload. IPsec can encrypt data between devices: router to router, security device to router, PC to router, and PC to server.

10.7.2 Security of Wireless Networks and IEEE 802.11

Wireless networks are particularly vulnerable because of their nonwired infrastructure. Intruders can access wireless networks by receiving radio waves carrying packets and frames propagated beyond the needed range of the network's base station and hosts. Our focus here is the security mechanisms for the wireless 802.11 standards known as *wired equivalent privacy* (WEP).

This section also describes types of security features desired for IEEE 802.11a, b, and i. WEP provides a level of security similar to that found in wired networks. It is a standard of security for IEEE 802.11a and b and offers authentication and data encryption between a host and a wireless base station, using a secret shared key. The essence of this protocol between a host and a base station (wireless access point) is as follows.

1. The host requests authentication from the base station.
2. The base station responds.
3. The host encrypts data by using secret-key encryption.
4. The base station decrypts the received encrypted data. If the decrypted data matches the original one sent to the host, the host is authenticated by the base station.

Figure 10.7 shows how data is encrypted. First, a 40-bit secret key, k , known by both the host and the base station, is created. A 24-bit initialization field to be used to encrypt a single frame is appended to this key. The initialization field is different for each frame.

As shown in the figure, a 4-byte CRC field is computed for the data payload. The payload and the CRC bytes are then encrypted. The encryption algorithm produces a stream of key values: $k_1, k_2, \dots, k_i, \dots, k_{n-1}, k_n$. Assume that the plaintext is partitioned into i bytes. Let c_i be the i th byte of the ciphertext and m_i be the i th byte of the plaintext; the encryption is done by using k_i , as follows:

$$c_i = m_i \oplus k_i. \quad (10.21)$$

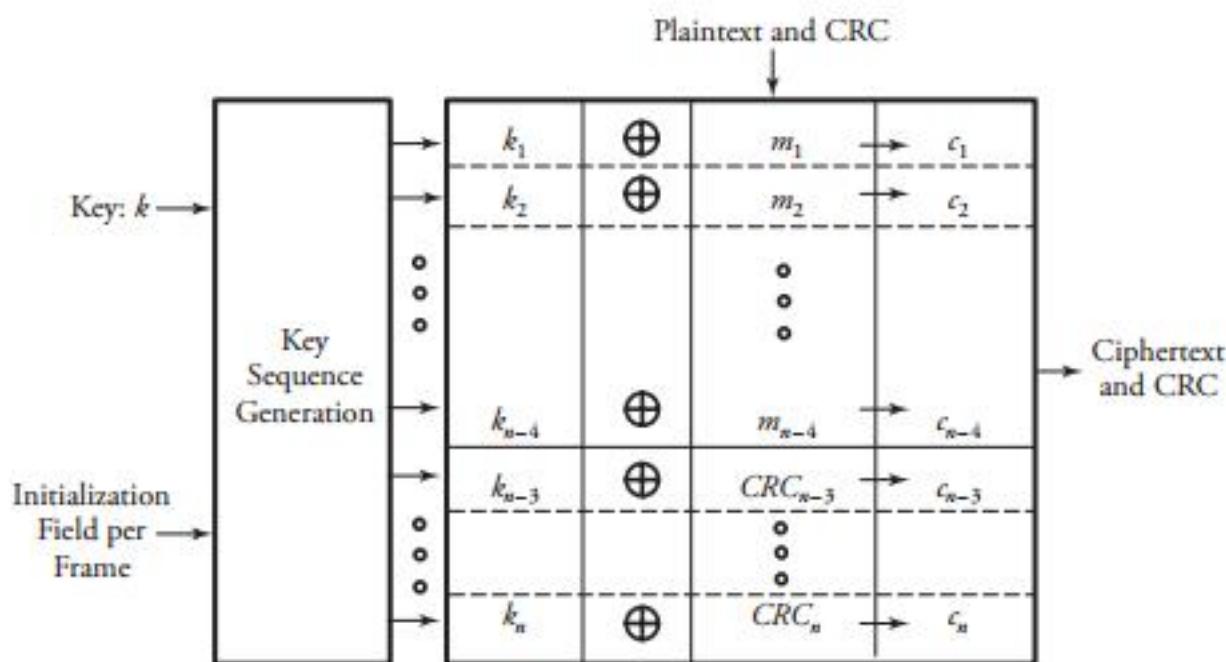


Figure 10.7 Security implementation in wireless IEEE 802.11

To decrypt the ciphertext, the receiver uses the same secret key as the sender used, appends the initialization field to it, and calculates

$$m_i = c_i \oplus k_i. \quad (10.22)$$

WEP is simple and relatively weak. The procedure for security of IEEE 802.11i is different because of its more sophisticated characteristics. This standard specifies an authentication server for the base-station communication. The separation of the authentication server from the base station means that the authentication server can serve many base stations. A new protocol, called *Extensible Authentication Protocol* (EAP), specifies the interaction between a user and an authentication server (IEEE 802.11i).

To summarize the IEEE 802.11i security mechanism: A base station first announces its presence and the types of security services it can provide to the wireless users. This way, users can request the appropriate type and level of encryption or authentication. EAP frames are encapsulated and sent over the wireless link. After decapsulation at the base station, the frames are encapsulated again, this time using a protocol called RADIUS for transmission over UDP to the authentication server. With EAP, public-key encryption is used, whereby a shared secret key known only to the user and the authentication server is created. The wireless user and the base station can also generate additional keys to perform the link-level encryption of data sent over the wireless link, which makes it much more secure than the one explained for 802.11a,b.

10.8 Firewalls

As the name suggests, a *firewall* protects data from the outside world. A firewall can be a software program or a hardware device. A firewall is a popular security mechanism for networks. A firewall is a simple router implemented with a special program. This unit is placed between hosts of a certain network and the outside world, as shown in Figure 10.8, and the rest of the network. The security issues faced by a smaller network like the one used at home are similar to larger networks. A firewall is used to protect the network from unwanted Web sites and potential hackers.

A firewall is placed on the link between a network router and the Internet or between a user and a router. The objective of such a configuration is to monitor and filter packets coming from unknown sources. Consequently, hackers do not have access to penetrate through a system if a firewall protects the system. For a large company with many small networks, the firewall is placed on every connection attached to the Internet. Companies can set rules about how their networks or particular systems need to work in order to maintain security. Companies can also set rules on how a system can connect to Web sites. These precautionary rules are followed in order to attain the advantage of having a firewall. Hence, the firewall can control how a network works with an Internet connection. A firewall can also be used to control data traffic.

Software firewall programs can be installed in home computers by using an Internet connection with these so-called gateways, the computer with such a software can access Web servers only through this software firewall. But hardware firewalls are more secure than software firewalls. Moreover, hardware firewalls are not expensive. Some firewalls also offer virus protection. The biggest security advantage of installing a firewall in a business network is to protect from any outsider logging on to the network under protection. Firewalls are preferred for use in almost all network security infrastructures,

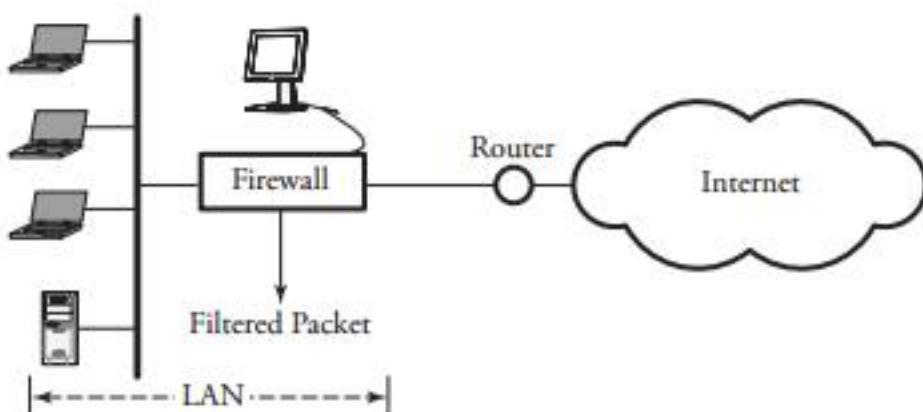


Figure 10.8 A simple configuration of a secured network using a firewall

as they allow the implementation of a security policy in one centralized place rather than end to end. Sometimes, a firewall is put where a large amount of data flow is normally expected.

A firewall controls the flow of traffic by one of the following three methods. The first method is *packet filtering*. Apart from forwarding packets between networks, a firewall filters those packets that pass through. If packets can get through the filter, they reach their destinations; otherwise, they are discarded. A firewall can be programmed to throw away certain packets addressed to a particular IP host or TCP port number. This condition is especially useful if a particular host does not want to respond to any access from an external source.

The second method is that a firewall filters packets based on the source IP address. This filtering is helpful when a host has to be protected from any unwanted external packets. The third method, denial of service, was explained earlier. This method controls the number of packets entering a network.

10.9 Summary

Network security is one of the top-priority issues in computer and communication networks. Security issues have pushed to the forefront of concern for end users, administrators, and equipment suppliers. Networking attacks can be classified as DNS *hacking*, *routing table poisoning*, *packet mistreatment*, and *denial of service*. Two major solutions for computer networking security are *cryptographic techniques* and *authentication techniques* (verification). The main tool that network security experts use to encrypt a message is a *key*, and the fundamental operation often used to encrypt a message is the Exclusive-OR operation.

Two secret-key encryption protocols are the *Data Encryption Standard* (DES), and the *Advanced Encryption Standard* (AES). In both methods, a secret key is shared between a transmitter and its receiver by assigning it to both the transmitter point and the receiver point. Public-key cryptography is more effective; a sender/receiver pair use different keys for encryption and decryption, and each party can publish its public (encryption) key so that anyone can use it to encrypt a message for that party. Two public-key protocols are the *Rivert, Shamir, and Aldeman* (RSA) protocol and the *Diffie-Hillman key-exchange* protocol.

A receiver can use *message authentication* methods to be assured that the incoming message is from its purported sender. Cryptographic hash functions are used in message authentication codes. Such codes can be generalized to produce a digital signature

that guarantees a document's *authenticity*. The *Secure Hash Algorithm* (SHA) has been proposed as part of the digital signature standard.

The IP security (IPsec) protocol requires both sender and receiver to exchange public encryption keys. IPsec has two encryption modes: tunnel and transport. The tunnel mode encrypts the header and the payload of each packet, and the transport mode encrypts the payload. IPsec can encrypt data between router and router, security device and router, PC and router, and PC and server. Another security mechanism is a firewall, which is placed on the link between a network router and the Internet or between a user and a router. The objective of such a configuration is to filter packets coming from unknown sources.

Part II of the book follows. Chapter 11 presents analytical methods for delay estimations of single queues of packets and networks of single queues.

10.10 Exercises

1. Assume that in round 4 of a DES process for a message, $L_4 = 4de5635d$ (in hex), $R_4 = 3412a90e$ (in hex) and $k_4 = be1142 7e6ac2$ (in hex). Find R_5 and L_5 .
2. Use DES to encrypt a 64-bit message, including all 1s with a 56-bit secret key consisting of 0101...01. Assuming a 1-bit rotation in the key process, find the outputs of the first round.
3. Check your system to find out about its encryption utility. If a DES algorithm is used:
 - (a) Use an experiment to measure the speed of encryption and decryption.
 - (b) Repeat part (a), using different keys.
4. Write a computer program to find the ciphertext for a message, using DES to encrypt a 64-bit message, including all 0s with a 56-bit secret key consisting of 1010...10. Assume that a 1-bit rotation in the key process.
5. In the RSA encryption algorithm, show how either of Equations (10.6) and (10.7) is concluded from the other one.
6. For an RSA encryption of a 4-bit message 1010, we choose $a = 5$, $b = 11$, and $x = 3$. Find the public and the private keys for this security action, and show the ciphertext.
7. Apply RSA and do the following.
 - (a) Encrypt $a = 5$, $b = 11$, $x = 7$, and $m = 13$.

- (b) Find the corresponding y .
 - (c) Decrypt the ciphertext
8. Normally the speed of the RSA encryption algorithm is much lower than secret-key encryption algorithms. To solve this issue, we can combine RSA with a secret-key encryption algorithm, such as AES. Assume that user 1 chooses an AES key of 256 bits and encrypts this key with the user 2's RSA public key and also encrypts the message with the AES secret key.
- (a) Prove that this AES key is too small to encrypt securely with RSA.
 - (b) Provide an example of a public key that can be recovered by an attacker.
9. Consider again the combination of the two encryption methods discussed in the previous problem. What solutions would be feasible to overcome the mentioned vulnerability of this method?
10. In the Diffie-Hillman key-exchange protocol, prove that the two keys k_1 and k_2 are equal.
11. *Computer simulation project.* Write a computer program to simulate the operation of packet filtering in firewalls. Your program must include the following features.
- (a) The firewall sets up to allow in only HTTP requests to a specific server.
 - (b) An outside intruder attempts to send packets with a forged internal address.
 - (c) All packets from a given remote server must be kept out.

PART II

ADVANCED CONCEPTS

This page intentionally left blank

CHAPTER 11

Packet Queues and Delay Analysis

Queueing models offer qualitative insights into the performance of communication networks and quantitative estimations of average packet delay. In many networking instances, a single buffer forms a *queue* of packets. A single queue of packets is a notion of packet accumulation at a certain router or even at an entire network. In the context of data communication, a *queueing buffer* is a physical system that stores incoming packets, and a *server* can be viewed as a switch or a similar mechanism to process and route packets to the desired ports or destinations. A *queueing system* generally consists of a queueing buffer of various sizes and one or more identical servers. This chapter focuses on delay analysis of single queueing units and queueing networks, including feedback. The following topics are covered:

- *Little's theorem*
- *Birth-and-death process*
- *Queueing disciplines*
- *Markovian FIFO queueing systems*
- *Non-Markovian and self-similar models*
- *Network of queues and delay models*

The chapter starts by analyzing two basic theorems: Little's theorem and birth-and-death processes. Next, various scenarios of queueing disciplines are presented: finite versus infinite queueing capacity, one server versus several servers, and Markovian

versus non-Markovian systems. Non-Markovian models are essential, as many network applications such as Ethernet, WWW, and multimedia traffic, cannot be modeled by Markovian patterns.

Networks of queues rely on two important and fundamental theorems: *Burke's theorem* and *Jackson's theorem*. Burke's theorem presents solutions to a network of several queues. Jackson's theorem is used when a packet visits a particular queue more than once. In such conditions, the network typically contains *loops* or *feedback*.

11.1 Little's Theorem

The fundamental concept of packet queueing is based on *Little's theorem*, or *Little's formula*. This formula states that for networks reaching steady state, the average number of packets in a system is equal to the product of the average arrival rate, λ , and the average time spent in the queueing system. For example, consider the communication system shown in Figure 11.1. Assume that the system starts with empty state at time $t = 0$. Let A_i and D_i be the i th arriving packet and i th departing packet, respectively, and let $A(t)$ and $D(t)$ be the total number of arriving packets and the total number of departing packets, respectively, up to a given time t .

Thus, the total time a packet i spends in the system is expressed by $T_i = D_i - A_i$, and the total number of packets in the system at time t is described by $K(t) = A(t) - D(t)$. The cumulative timing chart of packets in a first-come, first-served service discipline is shown in Figure 11.2. The total time that all packets spend in the system can be expressed by $\sum_{i=1}^{A(t)} T_i$. Thus, the average number of packets in the system, K_a , can be obtained by

$$K_a = \frac{1}{t} \sum_{i=1}^{A(t)} T_i. \quad (11.1)$$

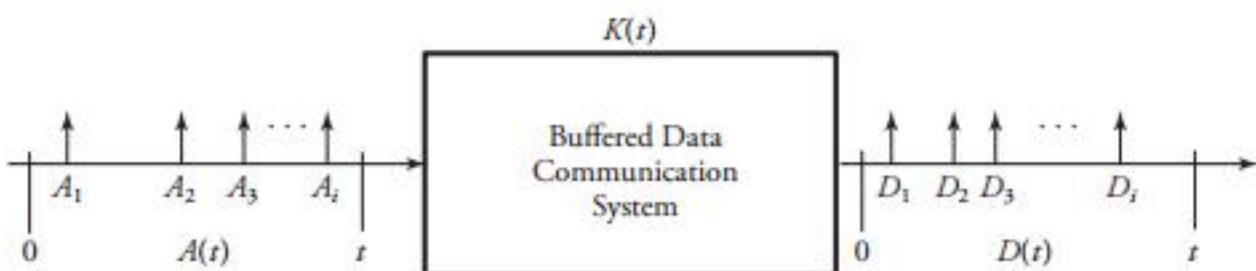


Figure 11.1 Overview of a buffered communication system with arriving and departing packets

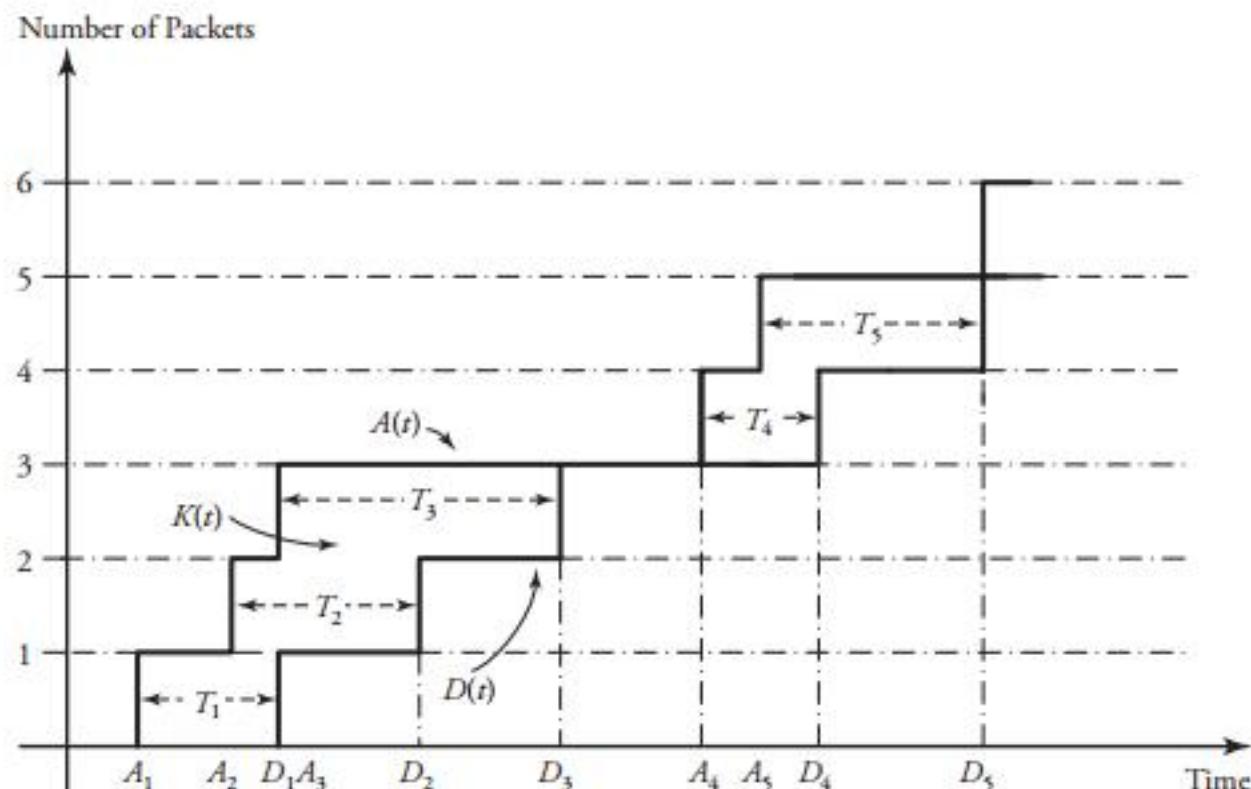


Figure 11.2 Accumulation of 5 packets in a queueing system

Similarly, the average time a packet spends in the system, T_a , can be represented by

$$T_a = \frac{1}{A(t)} \sum_{i=1}^{A(t)} T_i. \quad (11.2)$$

Also, the average number of arriving packets per second is given by λ as

$$\lambda = \frac{A(t)}{t}. \quad (11.3)$$

By combining Equations (11.1), (11.2), and (11.3), we obtain

$$K_a = \lambda T_a; \quad (11.4)$$

This important result leads to a final form: Little's formula. This formula makes the assumption that t is large enough and that K_a and T_a therefore converge to their expected values of corresponding random processes $E[K(t)]$ and $E[T]$, respectively. Thus:

$$E[K(t)] = \lambda E[T]. \quad (11.5)$$

Little's formula holds for most service disciplines with an arbitrary number of servers. Therefore, the assumption of first-come, first-served service discipline is not necessary.

Example. Consider a data-communication system with three transmission lines; packets arrive at three different nodes with arrival rates $\lambda_1 = 200$ packets/sec, $\lambda_2 = 300$ packets/sec, and $\lambda_3 = 10$ packets/sec, respectively. Assume that an average of 50,000 same-size packets float in this system. Find the average delay per packet in the system.

Solution. This delay is derived by

$$T_a = \frac{K_a}{\lambda_1 + \lambda_2 + \lambda_3} = \frac{50,000}{200 + 300 + 10} = 98.4 \text{ sec.}$$

This is a simple application of Little's formula in a communication system.

11.2 Birth-and-Death Process

If a packet arrives at or leaves from a queueing node of a computer network, the state of the node's buffer changes, and thus the state of the queue changes as well. In such cases, as discussed in Appendix C, if the system can be expressed by a *Markov process*, the activity of the process—in terms of the number of packets—can be depicted by a state machine known as the *Markov chain*. A particular instance of a Markov chain is the *birth-and-death process*.

In a *birth-and-death process*, any given state i can connect only to state $i - 1$ with rate μ_i or to state $i + 1$ with rate λ_i , as shown in Figure 11.3. In general, if $A(t)$ and $D(t)$ are the total number of arriving packets and the total number of departing packets, respectively, up to given time t , the total number of packets in the system at time t is described by $K(t) = A(t) - D(t)$. With this analysis, $A(t)$ is the total number of *births*, and $D(t)$ is the total number of *deaths*. Therefore, $K(t)$ can be viewed as a birth-and-death process representing the cumulative number of packets in a first-come, first-served service discipline.

Let p_i be the probability that the chain is in state i . The balance equation for the steady-state probability at state 0 denoted by p_0 is related to the one for State 1 denoted by p_1 as

$$\lambda_0 p_0 = \mu_1 p_1. \quad (11.6)$$

Similarly, for state 1, the balance equation is

$$\lambda_0 p_0 + \mu_2 p_2 = \lambda_1 p_1 + \mu_1 p_1, \quad (11.7)$$

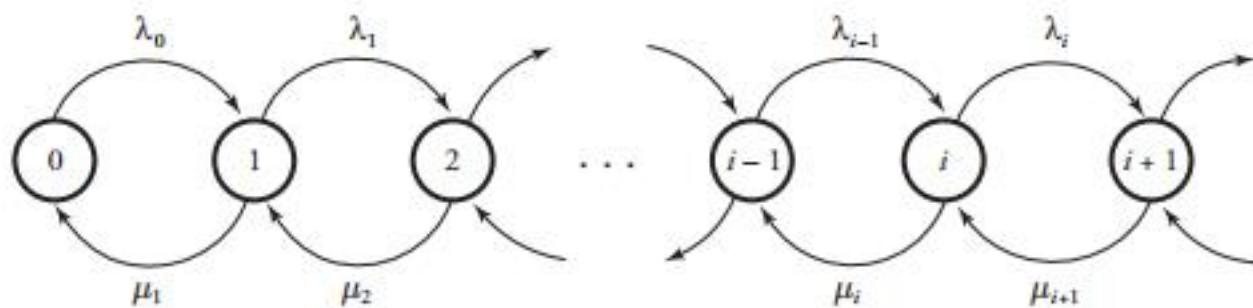


Figure 11.3 A general birth-and-death process and its transitions

and by using Equation (11.6), we derive the balance equation of state 1 as $\lambda_1 p_1 - \mu_2 p_2 = 0$. This development can continue up to state $i - 1$, where the balance equation can be derived as

$$\lambda_{i-1} p_{i-1} - \mu_i p_i = 0. \quad (11.8)$$

We define $\rho_i = \frac{\lambda_{i-1}}{\mu_i}$ as *state i utilization*. Thus, we can express the balance Equation (11.8) by a generic form as

$$p_i = \rho_i p_{i-1}. \quad (11.9)$$

If we apply Equation (11.9) for every state, starting with $p_1 = \rho_1 p_0$ up to $p_i = \rho_i p_{i-1}$, and combine all the equations, we obtain a generic equation

$$p_i = (\rho_1 \rho_2 \cdots \rho_i) p_0. \quad (11.10)$$

Since the sum of all the probabilities is always equal to 1, as $\sum_{i=0}^{\infty} p_i = \sum_{i=0}^{\infty} (\rho_1 \rho_2 \cdots \rho_i p_0) = 1$, p_i in Equation (11.10) can also be obtained from

$$p_i = \frac{\rho_1 \rho_2 \cdots \rho_i}{\sum_{j=0}^{\infty} (\rho_1 \rho_2 \cdots \rho_i)}. \quad (11.11)$$

The significance of this result is that a state probability p_i is expressed in terms of state utilizations, $\rho_1, \rho_2, \dots, \rho_i$ only.

11.3 Queueing Disciplines

Queueing systems are widely described by *Kendall's notations* as $A/B/m/K$, where A denotes the distribution of the interarrival times; B , the distribution of the service times; m , the number of servers; and K , the total capacity of the system. If a system

reaches its full capacity, the arriving packet number $K + 1$ is blocked away. A and B are normally represented by the following symbols:

M	Exponential distribution
E_k	Erlang distribution with k phases
H_k	Hyperexponential distribution with k phases
D	Deterministic distribution
G	General distribution
GI	General distribution with independent interarrival times
$MMPP$	Markov-modulated Poisson process
$BMAP$	Batch Markovian arrival process

In a queueing system, packets arrive at the buffer and are stored there. If other packets are waiting for service in the queue, the incoming packet is stored as long as all other packets are ahead of it. Thus, a server state can be either *idle* or *busy*. When a currently in-service packet departs from the system, one of the stored packets is selected for service according to the *scheduling discipline*. A commonly used arriving process assumes that the sequence of interarrival times is *independent and identically distributed* (IID).

Queueing systems can be classified as follows:

- *First come, first served* (FCFS). Packets are served in the order they arrive.
- *Last come, first served* (LCFS). Packets are served in reverse order of their arrival.
- *Random service*. Packets are selected at random for service.
- *Priority service*. The selection of a packet in the queue depends on priorities that are locally or permanently assigned to the packet.
- *Preemption*. The packet currently being serviced can be interrupted and preempted if a packet in the queue has a higher priority.
- *Round robin*. If the time allowed for the service of a packet is through and the process is not finished, the packet returns to the queue, and the action is repeated until the job service is completed.
- *Service sharing*. Servers can share their power with one another to provide services to the packets requiring more processing time.

This chapter focuses on FIFO models to show how the fundamental queueing in computer networks works. Chapter 12 looks at the applications of more advanced queuing models, such as priority, preemption, and round-robin queues.

11.4 Markovian FIFO Queueing Systems

In *Markovian* queueing systems, both arrival and service behaviors are based on Markovian-model queueing disciplines: $M/M/1$, $M/M/1/b$, $M/M/a$, $M/M/a/a$, and $M/M/\infty$.

11.4.1 $M/M/1$ Queueing Systems

Figure 11.4 shows a simple queueing system model. Packets arrive according to a Poisson model at rate λ so the interarrival times are IID exponential random variables with mean $1/\lambda$. When the server is prepared for service, packets from the queue enter the server. Consider the situation in which the service times are IID exponential random variables with mean $1/\mu$ and the interarrival and service times are independent. With the $M/M/1$ model, it is in fact assumed that the system can accommodate an unlimited number of packets.

Packet Arrival and Service Model

In an $M/M/1$ model, the *packet arrival* distribution is Poisson (see Appendix C) with rate λ , or the interarrival time distribution is exponential with mean time $1/\lambda$. Similarly, the service rate distribution is Poisson with rate μ , or the service time distribution is exponential with mean time $1/\mu$. Note that the interarrival time and the service time are independent. Figure 11.5 shows an example in which five packets, A_1 through A_5 , arrive in random and require service times S_1 through S_5 , respectively. In this situation,

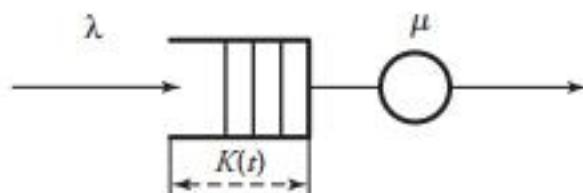


Figure 11.4 A simple queue/server model

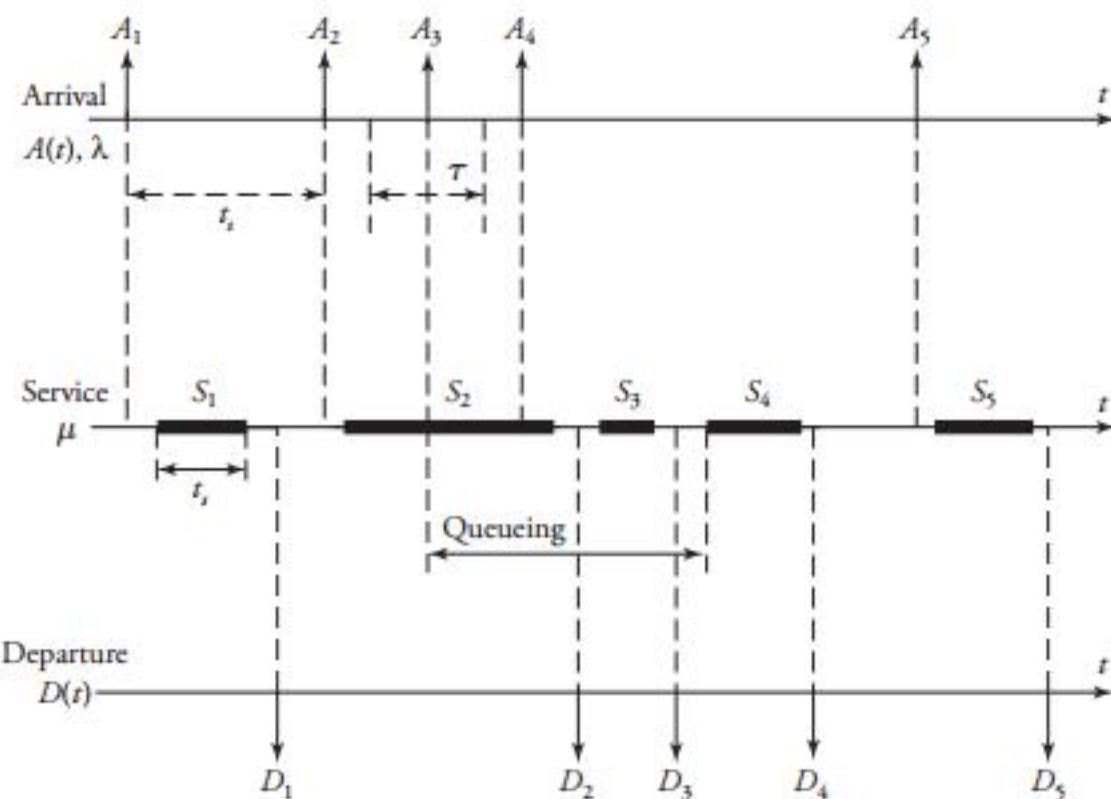


Figure 11.5 Packet arrival, service, and departure

queueing occurs while packets A_3 and A_4 wait in the buffer until the server becomes available. The probability that one packet arrives in an interval τ but no packet departs from the system is obtained using the Poisson formula:

$$\begin{aligned}
 P[1 \text{ packet arrives, } 0 \text{ packets depart}] &= \frac{\lambda \tau e^{-\lambda \tau}}{1!} \\
 &= \lambda \tau \left(1 - \frac{\lambda \tau}{1!} + \frac{(\lambda \tau)^2}{2!} - \dots\right) \\
 &= \lambda \tau + \left(-\frac{(\lambda \tau)^2}{1!} + \frac{(\lambda \tau)^3}{2!} - \dots\right) \\
 &\approx \lambda \tau.
 \end{aligned} \tag{11.12}$$

Equation (11.12) assumes that interval τ becomes very small and that therefore, all terms except the first one can be ignored. The same analysis can be derived for the case in which no packets arrive in an interval τ , but one packet departs from the

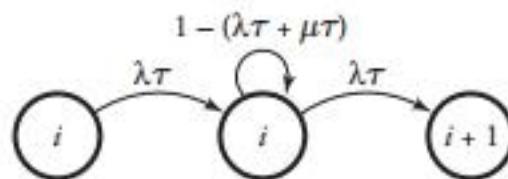


Figure 11.6 Packet arrival and departure

system. Because one packet is serviced in an interval τ , we can use rate μ instead of λ in Equation (11.12). Thus:

$$\begin{aligned}
 P[0 \text{ packets arrive, } 1 \text{ packet departs}] &= \mu\tau \frac{e^{-\mu\tau}}{1!} \\
 &= \mu\tau \left(1 - \frac{\mu\tau}{1!} + \frac{(\mu\tau)^2}{2!} - \dots\right) \\
 &= \mu\tau + \left(-\frac{(\mu\tau)^2}{1!} + \frac{(\mu\tau)^3}{2!} - \dots\right) \\
 &\approx \mu\tau.
 \end{aligned} \tag{11.13}$$

The last situation occurs when no change in the number of packets in the system is made. This means that there can be one arriving packet and one departing packet, or there can be no arriving packet and no departing packet. In either case, the probability can be derived by using Equations (11.12) and (11.13):

$$P[\text{no changes in the number of packets}] \approx 1 - (\lambda\tau + \mu\tau). \tag{11.14}$$

This analysis of the process is illustrated in Figure 11.6. In this process, a chain starts from a given state i , implying i packets in the system.

Number of Packets in the System

Properties obtained from Equations (11.12), (11.13), and (11.14) lead to the derivation of the number of packets in an $M/M/1$ system: $K(t)$. The fact is that $K(t)$ follows a birth-and-death process, and its Markov chain follows exactly as the generic one shown in Figure 11.3 but with equal rates as modified in Figure 11.7. The main property of the exponential random variable implies that the interarrival time is independent of the present and past status of $K(t)$. Thus, if the system has $K(t) > 0$ packets, the interdeparture time distribution is also an exponential random variable. This also implies that the past history of the system is irrelevant for the probabilities of future

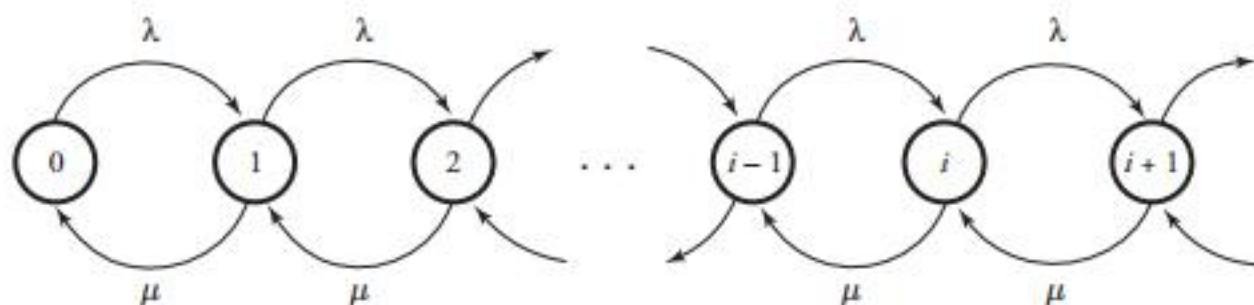


Figure 11.7 *M/M/1* queueing behavior and its transitions

states, and thus the process follows the property of a Markov chain. Note that *M/M/1* is a simple case of a birth-death process with:

$$\lambda_0 = \lambda_1 = \lambda_2 = \dots = \lambda \quad (11.15)$$

and

$$\mu_0 = \mu_1 = \mu_2 = \dots = \mu. \quad (11.16)$$

We define $\rho = \frac{\lambda}{\mu}$ as the system *utilization*. The preceding Markov chain has the state-transition rate diagram shown in Figure 11.7, and its global balance equations for the steady-state probabilities can be set by using the birth-death process results simplified as

$$p_i = (\rho\rho \cdots \rho)p_0 = \rho^i p_0. \quad (11.17)$$

Since the sum of all probabilities is always 1, we can write

$$\sum_{i=0}^{\infty} p_i = \sum_{i=0}^{\infty} \rho^i p_0 = 1. \quad (11.18)$$

A stable situation exists when the arrival rate is less than the service rate, or $\rho = \lambda/\mu < 1$, in which case $K(t)$ does not increase in value without bound, and therefore $\sum_{i=0}^{\infty} \rho^i \rightarrow \frac{1}{1-\rho}$. This fact simplifies Equation (11.17) for the derivation of the number of packets in the system:

$$p_i = P[K(t) = i] = (1 - \rho)\rho^i \quad i = 1, 2, \dots \quad (11.19)$$

The condition $\rho = \lambda/\mu < 1$ must be met if the system is designed to be stable; otherwise, packets arrive at the system more quickly than they can be processed. It is apparent that if the packet-arrival rate exceeds the processing-capacity rate of servers, the

number of packets in the queue grows without bound, leading to an unstable situation. Note that the distribution of $K(t)$ follows a *geometric distribution*.

Mean Delay and Queue Length

The result derived from Equation (11.19) can be used to calculate the mean delay a packet spends in a queue or the entire system and the mean length of a queue. The average number of packets in the system is, in fact, the expected value of $K(t)$:

$$\begin{aligned} E[K(t)] &= \sum_{i=0}^{\infty} i P[K(t) = i] = \sum_{i=0}^{\infty} i(1 - \rho)\rho^i = (1 - \rho) \sum_{i=0}^{\infty} i\rho^i \\ &= \frac{\rho}{1 - \rho}. \end{aligned} \quad (11.20)$$

Little's formula can be used to derive the mean packet delay in the system:

$$E[T] = \frac{1}{\lambda} E[K(t)] = \frac{1}{\lambda} \frac{\rho}{1 - \rho} = \frac{1}{\mu - \lambda}. \quad (11.21)$$

We can now show the mean waiting time of a packet in the queue, $E[T_q]$, in terms of $E[T]$ and the mean processing time of a packet in the server, $E[T_s]$:

$$E[T_q] = E[T] - E[T_s]. \quad (11.22)$$

The mean processing time of a packet in the server, $E[T_s]$, is in fact $\frac{1}{\mu}$; therefore:

$$\begin{aligned} E[T_q] &= \frac{1}{\lambda} \frac{\rho}{1 - \rho} - \frac{1}{\mu} \\ &= \frac{\lambda}{\mu} \frac{1}{\mu - \lambda}. \end{aligned} \quad (11.23)$$

Example. Use Little's formula to find the mean number of packets in the queue.

Solution. Little's formula can be applied in most stochastic systems, such as a queue defined solely as a system. If $E[K_q(t)]$ is the mean number of packets in the queue:

$$E[K_q(t)] = \lambda E[T_q] = \frac{\rho^2}{1 - \rho}.$$

Interestingly, the mean number of packets is a function only of utilization. In other words, the value of utilization in any single-queue system determines the average number of packets in the system.

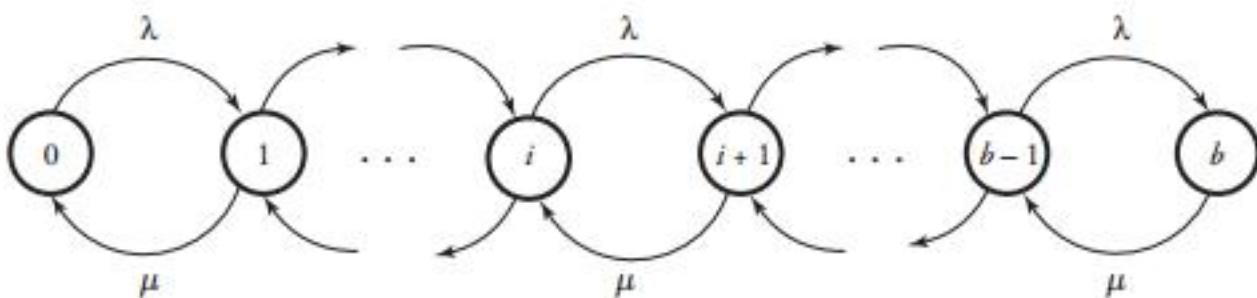


Figure 11.8 State-transition diagram for $M/M/1/b$ system

11.4.2 Systems with Limited Queueing Space: $M/M/1/b$

In a queue with exponential interarrival times and exponential service times, a single server and a maximum of b packets can be held in the system. Once such a system exceeds its capacity, arriving packet numbered $b + 1$ is turned away from the system, since there is no room for it. Figure 11.8 shows the state-transition diagram for the $M/M/1/b$ system. This diagram is a Markov chain and is very similar to the one illustrated in Figure 11.7, but the chain for the $M/M/1/b$ system stops at the maximum capacity of b . Thus, the global balance equations use the same results presented in Equation (11.18), where the sum of all probabilities is always 1 and thus

$$\sum_{i=0}^b p_i = \sum_{i=0}^b \rho^i p_0 = 1. \quad (11.24)$$

A stable situation exists when the arrival rate is less than the service rate, or $\rho = \lambda/\mu < 1$, in which case $K(t)$ does not increase in value without bound and therefore $\sum_{i=0}^b \rho^i \rightarrow \frac{1-\rho^{b+1}}{1-\rho}$. This fact simplifies Equation (11.17) for deriving the number of packets in the system, which is a continuous-time Markov chain taking on values from the set $\{0, 1, \dots, b\}$:

$$p_i = P[K(t) = i] = \begin{cases} \rho^i \frac{1-\rho}{1-\rho^{b+1}} & \text{if } \rho \neq 1 \text{ and } i = 1, 2, \dots, b \\ \frac{1}{1+b} & \text{if } \rho = 1 \end{cases}. \quad (11.25)$$

Note that the number of packets in the system is derived in two parts, depending on whether the utilization is maximized ($\rho = 1$) or not.

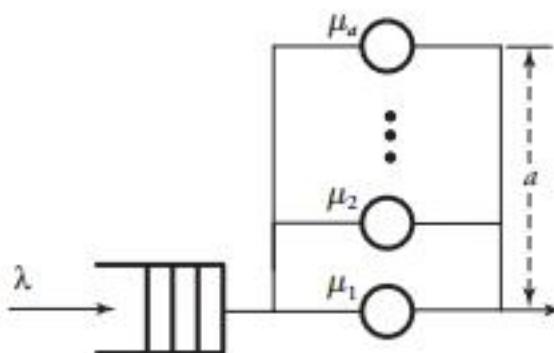


Figure 11.9 Queueing model of $M/M/a$ systems

Mean Number of Packets

The mean number of packets in the system, $E[K(t)]$, can be expressed by Formula C.20 in Appendix C:

$$E[K(t)] = \sum_{i=0}^b i P[K(t) = i] = \begin{cases} \frac{\rho}{1-\rho} - \frac{(b+1)\rho^{b+1}}{1-\rho^{b+1}} & \text{if } \rho \neq 1 \\ \frac{b}{2} & \text{if } \rho = 1 \end{cases}. \quad (11.26)$$

Clearly, results of the mean number of packets in the $M/M/1$ and $M/M/1/b$ systems are fundamentally different, owing to the difference in capacities available in these two systems.

11.4.3 $M/M/a$ Queueing Systems

Some stochastic systems are modeled by $M/M/a$, where a servers instead of one can be used to enhance system performance, as shown in Figure 11.9. The service rate in each server may be different from or equal to that in the others, as rates μ_1 through μ_a are indicated as service rates of the corresponding servers. The advantage of this discipline over systems with one server is its parallel-serving feature. A packet can be partitioned into $i < a$ different tasks and served in parallel in i different servers concurrently.

The resulting system can be modeled by a continuous-time Markov chain, as shown in Figure 11.10. The maximum number of servers is denoted by a . As long as the number of packets in the queue is less than a , the service rate is proportionally changing ($i\mu$) with the number of waiting packets while the arrival rate remains constant. However, when the number of packets in the queue reaches a , the arrival rate stays constant at its maximum of $a\mu$.

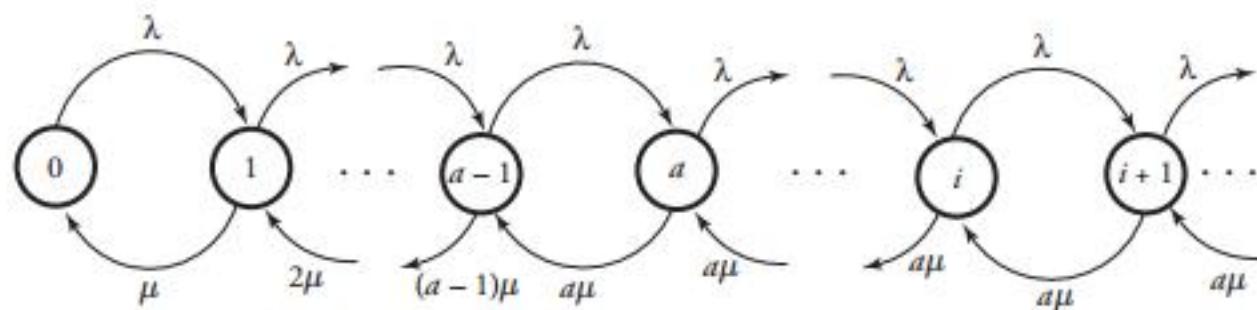


Figure 11.10 State-transition diagram for $M/M/a$ systems

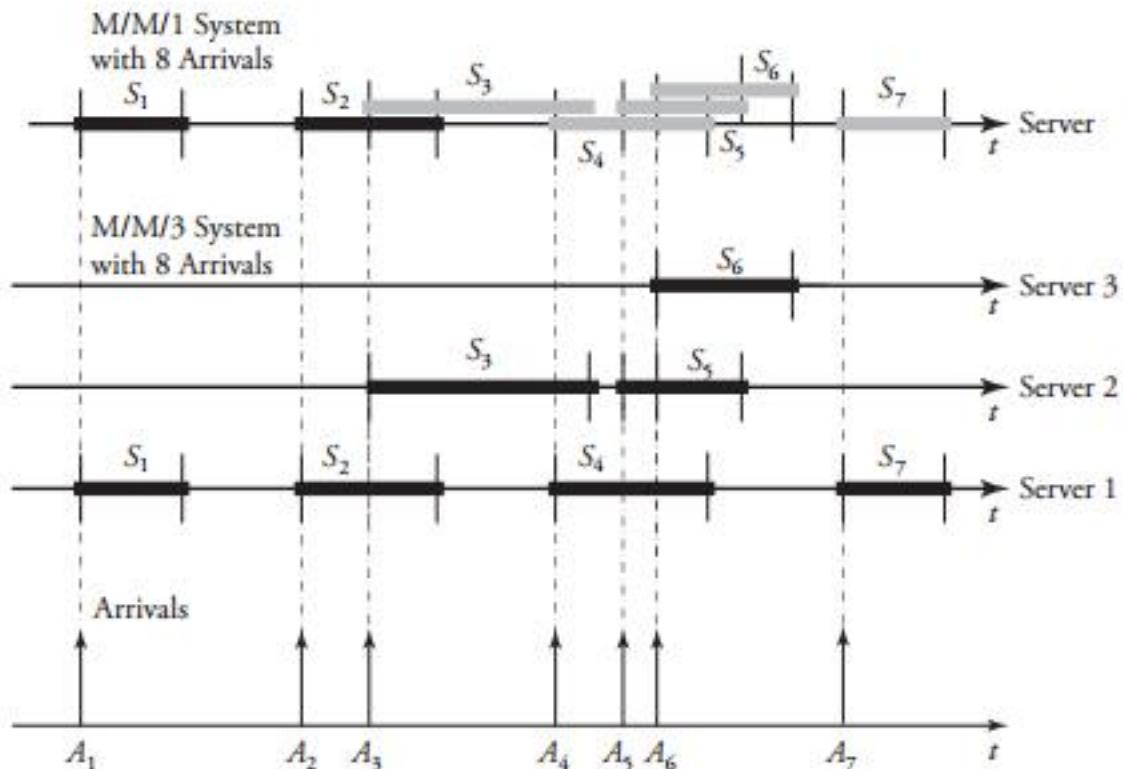


Figure 11.11 Comparison of $M/M/1$ and $M/M/3$ systems, each with seven packets

Example. Figure 11.11 shows a statistical comparison between $M/M/1$ and $M/M/3$ systems, each with seven packets. In the $M/M/1$ system, the third packet, S_3 , starts to accumulate in the queue. The queueing trend continues, as shown by gray, if the service rate stays low. On the contrary, in the $M/M/3$ system, packets are distributed over the three servers on arrival, and queueing has not happened.

Balance Equations for $i \leq a$

The steady-state probabilities for the $M/M/a$ system can be obtained by using the birth-and-death process. Balance equations for i servers being busy when $i \leq a$ begin with p_i :

$$p_i = \left(\frac{\lambda}{i\mu} p_{i-1} \right) \quad i = 1, \dots, a. \quad (11.27)$$

Recall that $\rho_i = \frac{\lambda}{i\mu}$ is the utilization and that $p_i = (\rho_1 \rho_2 \dots \rho_{i-1} \rho_i) p_0$. Thus, p_i can be rewritten as

$$p_i = \left(\frac{\lambda}{\mu} \frac{\lambda}{2\mu} \dots \frac{\lambda}{(i-1)\mu} \frac{\lambda}{i\mu} \right) p_0 = \frac{\left(\frac{\lambda}{\mu}\right)^i}{i!} p_0 = \left(\frac{\rho_1^i}{i!} \right) p_0. \quad (11.28)$$

It is easy to realize the probability that all servers are in use. Substitute $i = a$ in Equation (11.28):

$$p_a = \left(\frac{\rho_1^a}{a!} \right) p_0 = \frac{\left(\frac{\lambda}{\mu}\right)^a}{a!} p_0. \quad (11.29)$$

Balance Equations for $i \geq a$

To determine the balance equations for $i \geq a$, we again use the birth-and-death process. The second part of the transition diagram, starting from state a , is formulated over states $i = a, a + 1, a + 2, \dots$. As a result, the birth-and-death process formula is set up by

$$p_i = \rho^{i-a} p_a. \quad (11.30)$$

where $\rho = \frac{\lambda}{a\mu}$. In this equation, p_a can be substituted by the value obtained last from the case of $i \leq a$, presented in Equation (11.29). Therefore, by replacing p_a with $\frac{\rho_1^a}{a!} p_0$, we obtain

$$p_i = \frac{\rho^{i-a} \rho_1^a}{a!} p_0. \quad (11.31)$$

Equations (11.28) and (11.31) together cover all the cases in the state diagram. Since the sum of all probabilities is always 1, $\sum_{i=0}^{\infty} p_i = 1$:

$$\sum_{i=0}^{a-1} \frac{\rho_1^i}{i!} p_0 + \frac{\rho_1^a}{a!} \sum_{i=a}^{\infty} \rho^{i-a} p_0 = 1. \quad (11.32)$$

To have a stable system, we want $\rho < 1$, which causes $\sum_{i=a}^{\infty} \rho^{i-a}$ to converge to $\frac{1}{1-\rho}$. With this simplification, we can compute the probability of the first state from Equation 11.32:

$$p_0 = \frac{1}{\sum_{i=0}^{a-1} \frac{\rho_1^i}{i!} + \frac{\rho_1^a}{a!} \frac{1}{1-\rho}}. \quad (11.33)$$

Similar to previous cases, knowing p_0 is essential, as we can derive p_i from Equation (11.31).

In an $M/M/a$ system, we can use the *Erlang-C formula* to compute the blocking probability. This formula calculates the probability that an arriving packet finds all servers busy. This probability is the same as the probability that the waiting time for the packet in the queue is greater than zero, or the probability that the number of packets in the queueing system is greater than or equal to a , $P[K(t) \geq a]$:

$$\text{Erlang-C Blocking Prob.: } P[K(t) \geq a] = \sum_{i=a}^{\infty} p_i = \frac{p_a}{1-\rho}. \quad (11.34)$$

As with the $M/M/1$ queueing system, we can now estimate all the parameters for the $M/M/a$ discipline. The mean number of packets in the queue is

$$\begin{aligned} E[K_q(t)] &= \sum_{i=a}^{\infty} (i-a)p_i = \sum_{i=a}^{\infty} (i-a)\rho^{i-a} p_a = p_a \sum_{i-a=0}^{\infty} (i-a)\rho^{i-a} \\ &= \frac{p_a \rho}{(1-\rho)^2}. \end{aligned} \quad (11.35)$$

Accordingly, the mean packet delay in the queue is

$$E[T_q] = \frac{1}{\lambda} E[K_q(t)] = \frac{p_a}{a\mu(1-\rho)^2}. \quad (11.36)$$

Obviously, the total mean packet delay in a system is

$$E[T] = E[T_q] + E[T_s] = \frac{p_a}{a\mu(1-\rho)^2} + \frac{1}{\mu}. \quad (11.37)$$

The mean number of packets in the system is obtained again from Little's formula:

$$\begin{aligned} E[K(t)] &= \lambda E[T] = \lambda \left(\frac{p_a}{a\mu(1-\rho)^2} + \frac{1}{\mu} \right) \\ &= \frac{p_a \rho}{(1-\rho)^2} + \rho_1. \end{aligned} \quad (11.38)$$

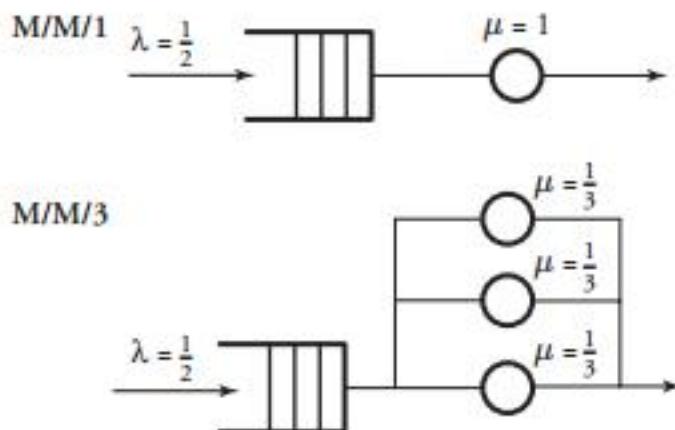


Figure 11.12 Two queueing disciplines with equal service powers

Example. A communication node receives packets at $\lambda = 1/2$ packet per nanosecond, and the switching server processes packets at the rate of one packet per nanosecond. Calculate the queueing delay and the total node's delay, and then compare the results for using queueing disciplines $M/M/1$ and $M/M/3$ with equal service powers as shown in Figure 11.12.

Solution. With an $M/M/1$ discipline, $\rho = \lambda/\mu = 0.5$. So the mean queueing delay is

$$E[T_q] = \frac{\rho/\mu}{1 - \rho} = 1 \text{ nsec}, \quad (11.39)$$

and the total node's delay is

$$E[T] = \frac{1/\mu}{1 - \rho} = 2 \text{ nsec}. \quad (11.40)$$

For the $M/M/3$ discipline, $a = 3$ servers are used; thus, $\rho_1 = \lambda/\mu = 1.5$, and $\rho = \lambda/a\mu = 0.5$. Thus:

$$p_0 = \frac{1}{\sum_{i=0}^2 \frac{\rho_1^i}{i!} + \frac{\rho_1^3}{3!} \frac{1}{1-\rho}}. \quad (11.41)$$

Since $p_a = \frac{\rho^a}{a!} p_0 = 0.095$, the queueing delay is

$$E[T_q] = p_a \frac{1}{a\mu(1-\rho)^2} = 0.38, \quad (11.42)$$

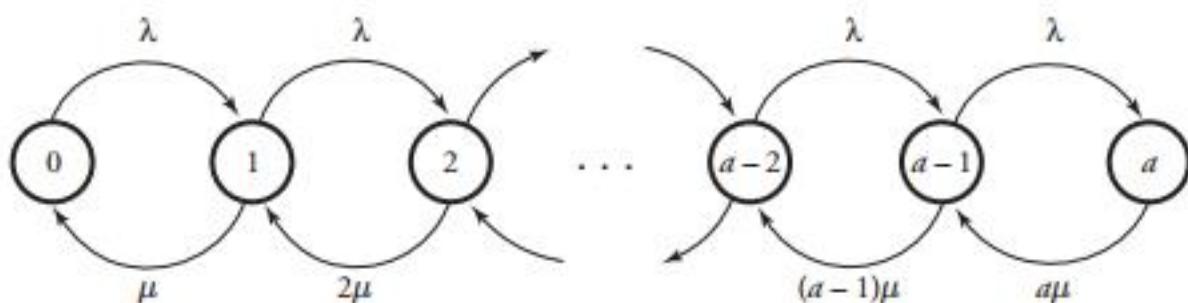


Figure 11.13 State-transition diagram for $M/M/a/a$ systems

and the total mean node's delay is

$$E[T] = E[T_q] + E[T_s] = 0.38 + \frac{1}{\mu} = 3.38. \quad (11.43)$$

This useful example expresses an important conclusion: *Increasing the number of servers lowers the queueing delay but increases the total system delay.* In summary, compared with the $M/M/a$ system ($a > 1$), the $M/M/1$ system has a smaller total delay but a larger queueing delay. However, note that the preceding statement should be made with an important condition; that is, the total service rates of a servers in the $M/M/a$ model must be equal to the service rate of the $M/M/1$ model.

11.4.4 Models for Delay-Sensitive Traffic: $M/M/a/a$

Some network nodes can be built with no buffering elements. One application of such systems is for cases in which *delay-sensitive traffic*, such as voice, needs a queueless service. These types of nodes can be shown by queueless models. One example of such models is the $M/M/a/a$ queueing system, which has a servers but no queueing line. If all servers are busy, an arriving packet is turned away, resulting in the transition-rate diagram shown in Figure 11.13. This transition rate partially resembles that for an $M/M/a$ system. Thus, we use the previously obtained equation for the probability that the system is in state $i \in \{0, \dots, a\}$:

$$p_i = \left(\frac{\rho_1^i}{i!} \right) p_0. \quad (11.44)$$

Combining Equation 11.44 and the fact that $\sum_{i=0}^a p_i = 1$ gives us

$$p_0 = \frac{1}{\sum_{i=0}^a \frac{\rho_1^i}{i!}}. \quad (11.45)$$

Erlang-B Blocking Probability

In an $M/M/a/a$ system, we can compute the blocking probability, this time using the *Erlang-B formula*. This formula calculates the probability that an arriving packet finds all servers busy but no waiting line for packets. The probability that all servers of the system are busy is referred to as the *Erlang-B formula*, using Equation (11.44) with $i = a$:

$$\begin{aligned} \text{Erlang-B Blocking Prob.: } P[K(t) = a] &= p_a = \frac{\rho_1^a}{a!} p_0 \\ &= \frac{\rho_1^a}{a! \left(\sum_{i=0}^a \frac{\rho_1^i}{i!} \right)}. \end{aligned} \quad (11.46)$$

When all servers are busy, an incoming packet is turned away. Let λ , λ_p , and $\lambda_b = p_a \lambda$ represent the arrival rate, the packet-passing rate, and the blocking rate, respectively. Therefore, $\lambda = \lambda_p + \lambda_b$. The packet-passing rate can then be rewritten as $\lambda_p = \lambda(1 - p_a)$. In such systems, it would be interesting to find the average number of packets in the system, since no queueing line exists. From Little's formula:

$$E[K(t)] = \lambda_p E[T_s] = \lambda(1 - p_a) \frac{1}{\mu}. \quad (11.47)$$

Example. When a mobile user in a wireless data network moves to a new region, a handoff process between the two regions needs to take place. If all the new region's channels are used, the handoff call is terminated or blocked. For real-time communications, this scenario is modeled by an $M/M/a/a$ system, where $a = c_i$ is the number of handoff servers of traffic type i . Assume that $c_i = 10, 50$, and 100 ; mean service time $1/\mu_i = 30$ msec; and handoff request rate $\lambda_i = 0 \dots 10,000$. Sketch a set of plots showing the performance of the handoff process.

Solution. The calculation effort is left to readers. Instead, the results of the computations in Figure 11.14 are given. The plots in this figure depict the results of an $M/M/a/a$ model for the wireless handoff. Equation (11.46) can be used to compute the handoff blocking probability.

11.4.5 $M/M/\infty$ Queueing Systems

In many high-speed network nodes, higher cost can be tolerated in exchange for better communication quality. One method of improving the quality of handling packets is to consider a large number of servers at a node. If we let a approach a real large number approximated to infinity, an $M/M/a/a$ system becomes an $M/M/\infty$ system. As a

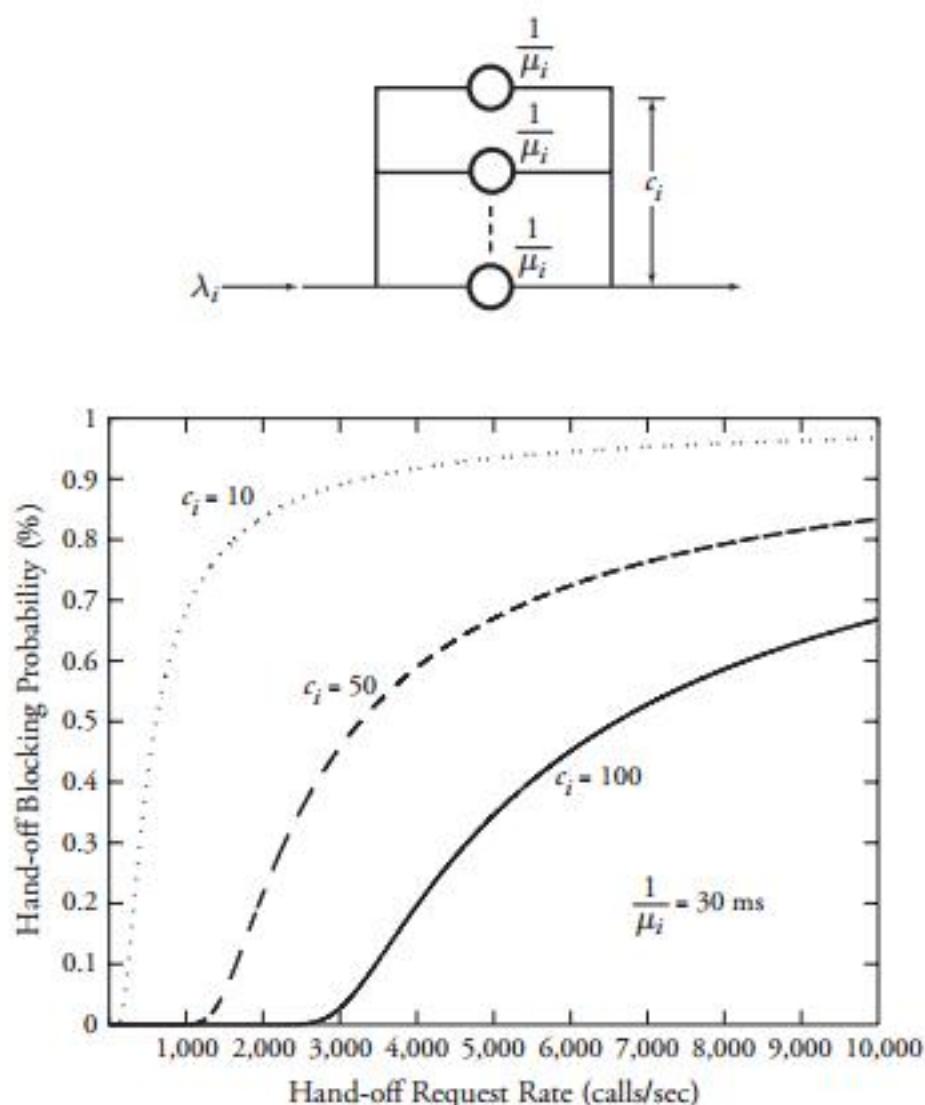


Figure 11.14 A wireless data network model as an $M/M/a/a$ system with c_i channel servers

result, the transition-rate diagram can continue to ∞ , as shown in Figure 11.15. To find steady-state probabilities, p_i , for this queueing system, we use Equation (11.46) of the $M/M/a/a$ system and let a be a number i varying up to ∞ :

$$p_i = \frac{\rho_1^i}{i! \left(\sum_{j=0}^i \frac{\rho_1^j}{j!} \right)}. \quad (11.48)$$

In the denominator of this equation, $\sum_{j=0}^i \frac{\rho_1^j}{j!}$ tends to converge to e^{ρ_1} when i approaches infinity:

$$p_i = \frac{\rho_1^i}{i! e^{\rho_1}}. \quad (11.49)$$

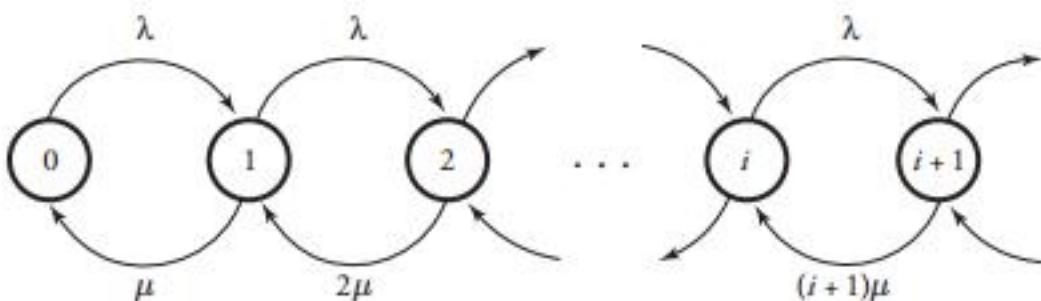


Figure 11.15 State-transition diagram for $M/M/\infty$ system

11.5 Non-Markovian and Self-Similar Models

Another class of queueing systems is either completely or partially *non-Markovian*. Partial non-Markovian means that either arrival or service behavior is not based on Markovian models. We first consider a single server in which packets arrive according to a Poisson process, but the service-time distribution is *non-Poisson* or *non-Markovian* as *general distribution*. One example of a non-Markovian system is $M/G/1$ queues with priorities. Another is $M/D/1$, where the distribution of service time is deterministic, not random. Yet another example is reservation systems in which part of the service time is occupied with sending packets—serving packets—and part with sending control information or making reservations for sending the packets.

11.5.1 Pollaczek-Khinchin Formula and $M/G/1$

Let us consider a single-server scenario in which packets arrive according to a Poisson process with rate λ , but packet service times have a *general distribution*. This means that the distribution of service times cannot necessarily be exponential, as shown in Figure 11.16. Suppose that packets are served in the order they are received, with T_{si} the service time of the i th packet. In our analysis, we assume that random variables T_{s1}, T_{s2}, \dots are identically distributed, mutually independent, and independent of the interarrival times. Let T_{qi} be the queueing time of the i th packet. Then:

$$T_{qi} = \pi_i + \sum_{j=i-K_{qi}}^{i-1} T_{sj}, \quad (11.50)$$

where K_{qi} is the number of packets found by the i th packet waiting in the queue on arrival, and π_i is the partial service time seen by the i th packet if a given packet j is already being served when packet i arrives. If no packet is in service, π_i is zero. In order

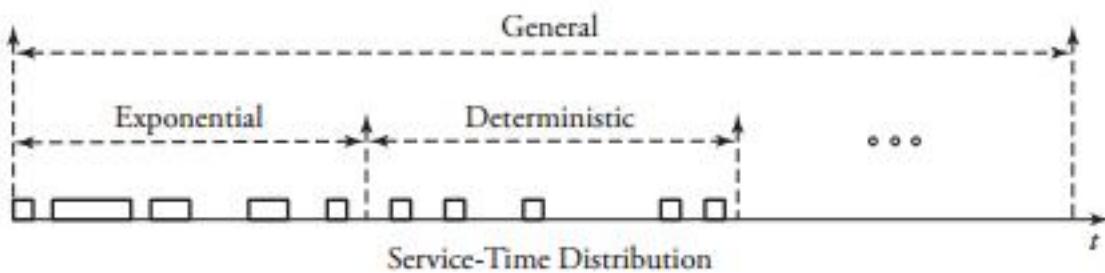


Figure 11.16 A general distribution of service times

to understand the averages of the mentioned times, we take expectations of two sides in Equation (11.50):

$$E[T_{qi}] = E[\pi_i] + E\left[\sum_{j=i-N_i}^{i-1} T_{sj}\right] = E[\pi_i] + E\left[\sum_{j=i-K_{qi}}^{i-1} E[T_{sj}|K_{qi}]\right]. \quad (11.51)$$

With the independence induction of the random variables K_{qi} and T_{qj} , we have

$$E[T_{qi}] = E[\pi_i] + E[T_{si}]E[K_{qi}]. \quad (11.52)$$

It is clear that we can apply the average service time as $E[T_{si}] = \frac{1}{\mu}$ in Equation (11.52). Also, by forcing Little's formula, where $E[K_{qi}] = \lambda E[T_{qi}]$, Equation (11.52) becomes

$$E[T_{qi}] = E[\pi_i] + \frac{1}{\mu} \lambda E[T_{qi}] = E[\pi_i] + \rho E[T_{qi}], \quad (11.53)$$

where $\rho = \lambda/\mu$ is the utilization. Thus,

$$E[T_{qi}] = \frac{E[\pi_i]}{1 - \rho}. \quad (11.54)$$

Figure 11.17 shows an example of mean π_i representing the remaining time for the completion of an in-service packet as a function of time. If a new service of T_{si} seconds begins, π_i starts at value T_{si} and decays linearly at 45 degrees for T_{si} seconds. Thus, the mean π_i in the interval $[0, t]$ with n triangles is equal to the area of all triangles ($\sum_{i=1}^n \frac{1}{2} T_{si}^2$) divided by total time (t):

$$E[\pi_i] = \frac{1}{t} \sum_{i=1}^n \frac{1}{2} T_{si}^2 = \frac{1}{2} \frac{n}{t} \frac{\sum_{i=1}^n T_{si}^2}{n}. \quad (11.55)$$

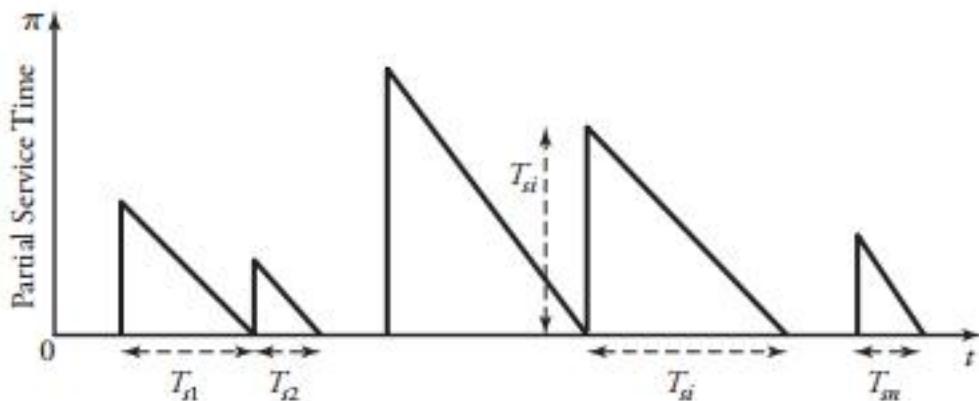


Figure 11.17 Mean partial service time

If n and t are large enough, $\frac{n}{t}$ becomes λ , and $\sum_{i=1}^n T_{si}^2/n$ becomes $E[T_{si}^2]$, the second moment of service time. Then, Equation (11.55) can be rewritten as

$$E[\pi_i] = \frac{1}{2}\lambda E[T_{si}^2]. \quad (11.56)$$

Combining Equations (11.55) and (11.54) forms the *Pollaczek-Khinchin* (P-K) formula:

$$E[T_{qi}] = \frac{\lambda E[T_{si}^2]}{2(1 - \rho)}. \quad (11.57)$$

Clearly, the total queueing time and service time is $E[T_{qi}] + E[T_{si}]$. By inducing Little's formula, the expected number of packets in the queue, $E[K_{qi}]$, and the expected number of packets in the system, $E[K_i]$, we get

$$E[K_{qi}] = \frac{\lambda^2 E[T_{si}^2]}{2(1 - \rho)} \quad (11.58)$$

and

$$E[K_i] = \rho + \frac{\lambda^2 E[T_{si}^2]}{2(1 - \rho)}. \quad (11.59)$$

These results of the Pollaczek-Khinchin theorem express the behavior of a general distribution of service time modeled as $M/G/1$. For example, if an $M/M/1$ system needs to be expressed by these results, and since service times are exponentially distributed, we have $E[T_{si}^2] = 2/\mu^2$, and the Pollaczek-Khinchin formula reduces to

$$E[T_{qi}] = \frac{\rho}{\mu(1 - \rho)}. \quad (11.60)$$

Note that the queueing metrics as the expected value of queuing delay obtained in Equation (11.60) is still a function of the system utilization. However, the notion of service rate μ has shown up in the equation and is a natural behavior of the non-Markovian service model.

11.5.2 M/D/1 Models

The distribution of service time in a computer network is deterministic. In the *asynchronous transfer mode*, discussed in Chapter 2, packets are equal sized, and may be so ATM fits well fits in the *M/D/1* model. When service times are identical for all packets, we have $E[T_{si}^2] = 1/\mu^2$. Then:

$$E[T_{qi}] = \frac{\rho}{2\mu(1-\rho)}. \quad (11.61)$$

Note that the value of $E[T_{qi}]$ for an *M/D/1* queue is the lower bound to the corresponding value for an *M/G/1* model of the same λ and μ . This is true because the *M/D/1* case yields the minimum possible value of $E[T_{si}^2]$. It is also worth noting that $E[T_{qi}]$ in an *M/M/1* queue is twice as much as the one in an *M/D/1* queue. The reason is that the mean service time is the same in the two cases, and for small ρ s, most of the waiting occurs in the service unit, whereas for large ρ s, most of the waiting occurs within the queue.

11.5.3 Self-Similarity and Batch-Arrival Models

The case of video streaming traffic, which obeys a model in the form of *batch arrival*, not Poisson, is analyzed in Chapter 18, so our analysis on non-Poisson arrival models is presented there. Here, we give some descriptive and intuitive examples of such models, which give rise to *self-similar* traffic.

Non-Markovian arrival models, or basically *non-Poisson arrival models*, have a number of important networking applications. The notion of self-similarity, for example, applies to WAN and LAN traffic. One example of such arrival situations is self-similarity traffic owing to the World Wide Web (WWW). The self-similarity in such traffic can be expressed based on the underlying distributions of WWW document sizes, the effects of caching and user preference in file transfer, and even user think time. Another example of self-similar traffic is video streaming mixed up with regular random traffic backgrounds. In such a case, batches of video packets are generated and are directed into a pool of mixed traffic in the Internet.

Traffic patterns indicate significant burstiness, or variations on a wide range of time scales. Bursty traffic can be described statistically by using *self-similarity patterns*. In a self-similar process, the distribution of a packet, frame, or object remains unchanged when viewed at varying scales. This type of traffic pattern has observable bursts and thus exhibits long-range dependence. The dependency means that all values at all times are typically correlated with the ones at all future instants. The long-range dependence in network traffic shows that packet loss and delay behavior are different from the ones in traditional network models using Poisson models.

11.6 Networks of Queues

Networks of queueing nodes can be modeled and their behavior, including feedback, studied. Two important and fundamental theorems are *Burke's theorem* and *Jackson's theorem*. Burke's theorem presents solutions to a network of several queues. Jackson's theorem is relevant when a packet visits a particular queue more than once.

11.6.1 Burke's Theorem

When a number of queueing nodes are connected, they all may be either in series or in parallel. In such circumstances, the entire system requires a careful and justified procedure for modeling. Burke's theorem presents solutions to a network of m queues: m queueing nodes in series and m queueing nodes in parallel.

Burke's Theorem on Cascaded Nodes

One common case is cascaded queueing nodes, as shown in Figure 11.18. In this network, the departure of a packet from a queueing node i is the arrival for node $i + 1$. The utilization for any node i is $\rho_i = \lambda / \mu_i$, where λ and μ_i are the arrival rate and service rate of that node, respectively.

The packet-movement activity of this network is modeled by an m -dimensional Markov chain. For simplicity, consider a two-node network of Nodes 1 and 2. Figure 11.19 shows a two-dimensional Markov chain for this simplified network. The chain starts at state 00. When the first packet arrives at rate λ , the chain goes to state 01. At this point, the chain can return to state 10 if it gets serviced at rate μ_1 , since the packet is now in node 2. If the packet is sent out of the second node at service rate μ_2 , the chain returns to state 00, showing the empty status of the system. The Markov chain can be interpreted similarly for a larger number of packets.

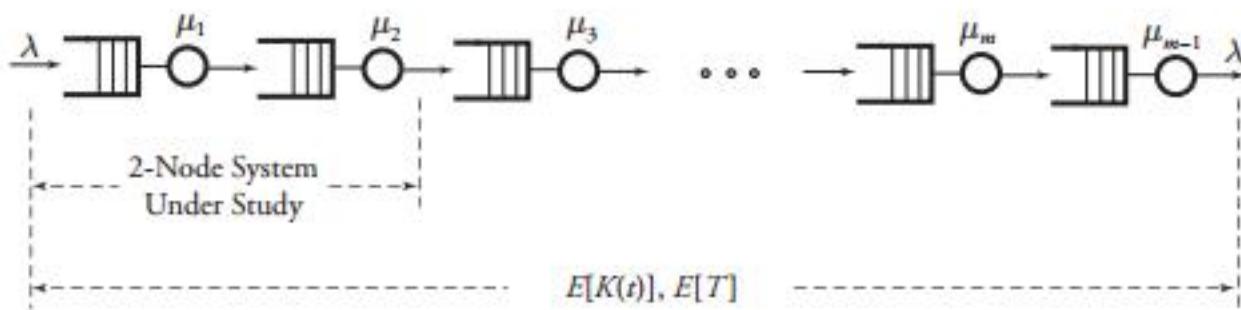


Figure 11.18 Burke's theorem applied on m cascaded queues

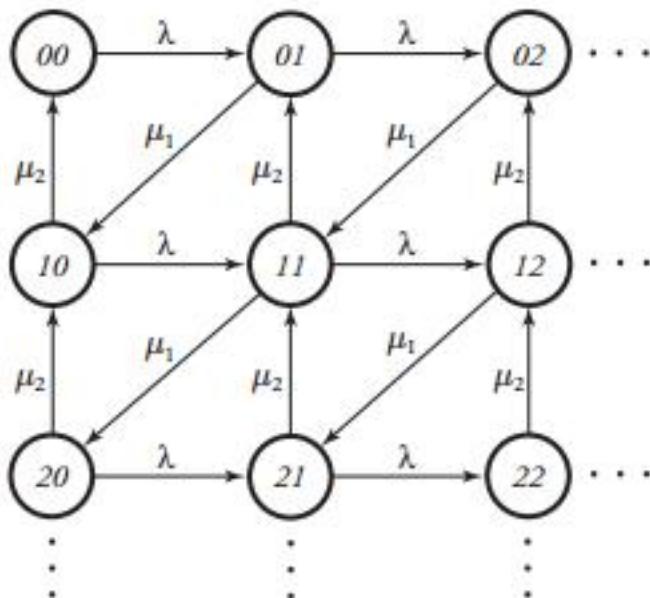


Figure 11.19 Two-dimensional Markov chain of the first two cascaded queues in Figure 11.18

In our system with m nodes, assume an $M/M/1$ discipline for every node; the probability that k packets are in the system, $P[K(t) = k]$, is:

$$\begin{aligned} P[K(t) = k] &= P[K_1(t) = k_1] \times P[K_2(t) = k_2] \times \cdots \times P[K_m(t) = k_m] \\ &= \prod_{i=1}^m (1 - \rho_i) \rho_i^{k_i}. \end{aligned} \quad (11.62)$$

The expected number of packets in each node, $E[K_i(t)]$, can easily be estimated by using the $M/M/1$ property of each node:

$$E[K_i(t)] = \frac{\rho_i}{1 - \rho_i}. \quad (11.63)$$

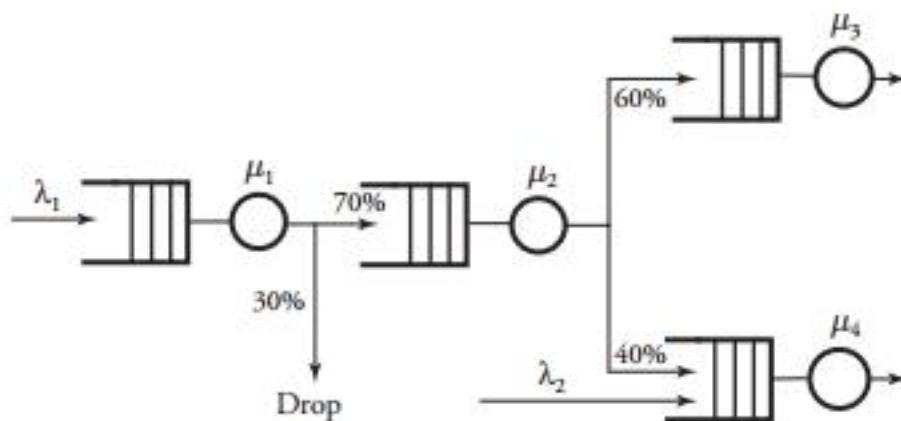


Figure 11.20 A Four-queueing-node system solved using Burke's theorem

Therefore, the expected number of packets in the entire system is derived by

$$E[K(t)] = \sum_{i=1}^m E[K_i(t)] = \sum_{i=1}^m \frac{\rho_i}{1 - \rho_i}. \quad (11.64)$$

We can now use Little's formula to estimate the total queueing delay that a packet should expect in the entire system:

$$E[T] = \frac{1}{\lambda} \sum_{i=1}^m E[K_i(t)] = \frac{1}{\lambda} \sum_{i=1}^m \frac{\rho_i}{1 - \rho_i}. \quad (11.65)$$

This last important result can help estimate the speed of a particular system modeled by a series of queueing nodes.

Example. Figure 11.20 shows four queueing nodes. There are two external sources at rates: $\lambda_1 = 2$ million packets per second and $\lambda_2 = 0.5$ million packets per second. Assume that all service rates are identical and equal to $\mu_1 = \mu_2 = \mu_3 = \mu_4 = 2.2$ million packets per second. Find the average delay on a packet passing through nodes 1, 2, and 4.

Solution. The figure shows the utilizations as

$$\rho_1 = \frac{\lambda_1}{\mu_1},$$

$$\rho_2 = \frac{0.7\lambda_1}{\mu_2},$$

and

$$\rho_4 = \frac{0.4(0.7)\lambda_1 + \lambda_2}{\mu_4} = \frac{0.28\lambda_1 + \lambda_2}{\mu_4}.$$

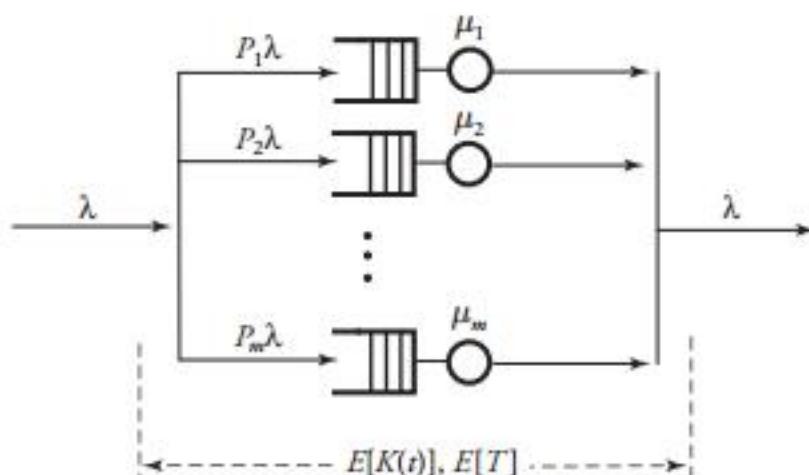


Figure 11.21 Application of Burke's theorem on parallel nodes

The expected numbers of packets in nodes 1, 2 and 4 are, respectively:

$$E[K_1(t)] = \frac{\rho_1}{1 - \rho_1} = \frac{\lambda_1}{\mu_1 - \lambda_1},$$

$$E[K_2(t)] = \frac{\rho_2}{1 - \rho_2} = \frac{0.7\lambda_1}{\mu_2 - 0.7\lambda_1},$$

and

$$E[K_4(t)] = \frac{\rho_4}{1 - \rho_4} = \frac{0.28\lambda_1 + \lambda_2}{\mu_4 - 0.28\lambda_1 - \lambda_2}.$$

The expected overall delay, then, is

$$E[T] = \frac{E[K_1(t)]}{\lambda_1} + \frac{E[K_2(t)]}{0.7\lambda_1} + \frac{E[K_4(t)]}{0.28\lambda_1 + \lambda_2}.$$

Burke's Theorem on Parallel Nodes

We can now easily apply Burke's theorem in cases of in-parallel queueing nodes, as shown in Figure 11.21. Suppose that the incoming traffic with rate λ is distributed over m parallel queueing nodes with probabilities P_1 through P_m , respectively. Also assume that the queuing nodes' service rates are μ_1 through μ_m , respectively. In this system, the utilization for any node i is expressed by

$$\rho_i = \frac{P_i\lambda}{\mu_i}. \quad (11.66)$$

Assuming that each node is represented by an $M/M/1$ queueing model, we calculate the expected number of packets in the queue of a given node i by

$$E[K_i(t)] = \frac{\rho_i}{1 - \rho_i}. \quad (11.67)$$

But the main difference between a cascaded queueing system and a parallel queueing system starts here, at the analysis on average delay. In the parallel system, the overall average number of packets, $\sum_{i=1}^m E[K_i(t)]$, is an irrelevant metric. The new argument is that a packet has a random chance of P_i to choose only one of the m nodes (node i) and thus is not exposed to all m nodes. Consequently, we must calculate the average number of queued packets that a new arriving packet faces in a branch i . This average turns out to have a perfect stochastic pattern and is obtained by $E[K(t)]$:

$$E[K(t)] = E[E[K_i(t)]] = \sum_{i=1}^m P_i E[K_i(t)] = \sum_{i=1}^m P_i \left(\frac{\rho_i}{1 - \rho_i} \right). \quad (11.68)$$

Using Little's formula, we can first estimate the delay incurred in branch i :

$$E[T_i] = \frac{E[K_i(t)]}{P_i \lambda}. \quad (11.69)$$

Now, the total queueing delay that a packet should expect in the entire system, $E[T]$, is the average delay over all m parallel branches. This delay is, in fact, a stochastic mean delay of all parallel branches, as follows:

$$\begin{aligned} E[T] &= E[E[T_i]] = \sum_{i=1}^m P_i E[T_i] = \sum_{i=1}^m P_i \frac{E[K_i(t)]}{P_i \lambda} \\ &= \frac{1}{\lambda} \sum_{i=1}^m \frac{\rho_i}{1 - \rho_i}. \end{aligned} \quad (11.70)$$

Obviously, $E[T]$ can help us estimate the speed of a particular m parallel-node system. An important note here is that Equations (11.64) and (11.68) may look the same. But the fact is that since the utilization in the case of cascaded nodes, $\rho_i = \lambda_i / \mu_i$, and the one for parallel nodes, $\rho_i = P_i \lambda_i / \mu_i$, are different, the results of $E[K(t)]$ in these cases too are different. We can argue identically on $E[T]$ for the two cases. Although Equations (11.65) and (11.70) are similar, we should expect the overall delay in the parallel-node case to be far less than the one for in-series case, owing to the same reason.

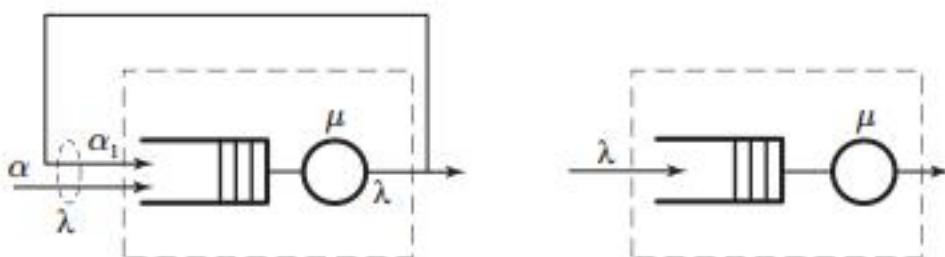


Figure 11.22 A feedback system and its simplified equivalent

11.6.2 Jackson's Theorem

One of the most frequently applied theorems in analyzing a network of queues is known as *Jackson's theorem*, which has several segments. Here, we emphasize the most applicable phase, known as open networks of queues. The essence of Jackson's theorem is applied to situations in which a packet visits a particular queue more than once. In such conditions, the network typically contains *loops*, or *feedback*.

Modeling Feedback in Networks

Figure 11.22 shows a basic cascaded queueing network, including a simple $M/M/1$ queueing element with a server in which simple feedback is implemented. Packets arrive at the left at rate α , passing through a queue and server with service rate μ and are partially fed back to the system at rate α_1 . Thus, this queueing architecture allows packets to visit the queue more than once by circulating them back to the system. This system is a model of many practical networking systems. One good example is switching systems that multicast packets step by step, using a recycling technique discussed in later chapters. In Figure 11.22, if the probability that a packet is fed back to the input is p , it is obvious that the total arrival rate to the queue, or λ , is

$$\lambda = \alpha + p\lambda. \quad (11.71)$$

Equation (11.71) demonstrates that the equivalent of the system surrounded by the dashed line can be configured as shown in Figure 11.22. Since $\alpha_1 = p\lambda$, we can combine this equation and Equation 11.71 and derive a relationship between λ and α as

$$\lambda = \frac{\alpha}{1 - p}. \quad (11.72)$$

The utilization of the simplified system denoted by $\rho = \frac{\lambda}{\mu}$ is, clearly:

$$\rho = \frac{\alpha}{\mu(1 - p)}. \quad (11.73)$$

By having the utilization of the system, we can derive the probability that k packets are in the system, $P[K(t) = k]$, knowing that the system follows the $M/M/1$ rule:

$$P[K(t) = k] = (1 - \rho)\rho^k = \frac{\mu(1 - p) - \alpha}{\mu(1 - p)} \left(\frac{\alpha}{\mu(1 - p)} \right)^k. \quad (11.74)$$

The expected number of packets in the system, $E[K(t)]$, can be derived by using the $M/M/1$ property:

$$E[K(t)] = \frac{\rho}{1 - \rho} = \frac{\alpha}{\mu(1 - p) - \alpha}. \quad (11.75)$$

Using Little's formula, we can now estimate the total queueing and service delay that a packet should expect. Let $E[T_u]$ be the expected delay for the equivalent unit shown in Figure 11.22 with lump sum arrival rate λ . Let $E[T_f]$ be the expected delay for the system with arrival rate α :

$$E[T_u] = \frac{E[K(t)]}{\lambda} = \frac{\alpha}{\lambda(\mu(1 - p) - \alpha)} \quad (11.76)$$

and

$$E[T_f] = \frac{E[K(t)]}{\alpha} = \frac{1}{\mu(1 - p) - \alpha}. \quad (11.77)$$

Note that we should always see the inequity $E[T_u] < E[T_f]$ as expected.

Open Networks

Figure 11.23 shows a generic model of Jackson's theorem when the system is open. In this figure, unit i of an m -unit system is modeled with an arrival rate sum of λ_i and a service rate μ_i . Unit i may receive an independent external traffic of α_i . The unit can receive feedback and feed-forward traffic from $i - 1$ other units. At the output of this unit, the traffic may proceed forward with probability P_{ii+1} or may leave the system with probability $1 - P_{ii+1}$. The total arrival rate, λ_i , is then

$$\lambda_i = \alpha_i + \sum_{j=1}^m P_{ji} \lambda_j. \quad (11.78)$$

The instantaneous total number of packets in all m nodes, $\mathbf{K}(t)$, is a vector consisting of the number of packets in every individual node and is expressed by

$$\mathbf{K}(t) = \{K_1(t), K_2(t), \dots, K_m(t)\}. \quad (11.79)$$

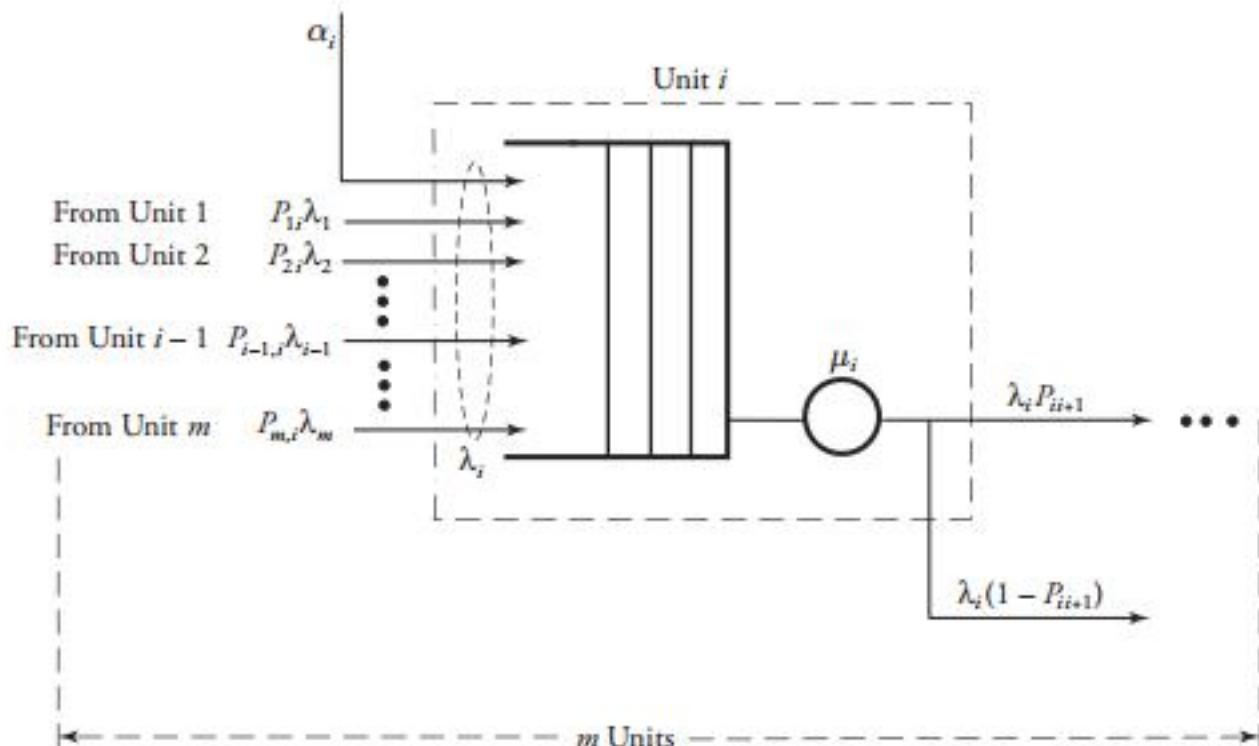


Figure 11.23 Generalization of a node model for Jackson's theorem

The probability that k packets are in the system is the joint probabilities of numbers of packets in m units:

$$P[K(t) = k] = \prod_{i=1}^m P[K_i(t) = k_i]. \quad (11.80)$$

The expected number of packets in all units is obtained by

$$E[K(t)] = E[K_1(t)] + E[K_2(t)] + \dots + E[K_m(t)]. \quad (11.81)$$

Finally, the most significant factor used in evaluating a system of queueing nodes, total delay at stage i , is calculated by applying Little's formula:

$$E[T_i] = \frac{\sum_{i=1}^m E[K_i(t)]}{\alpha_i}. \quad (11.82)$$

We assume that α_i is the only input to the entire system. In practice, there may be other inputs to the system, in which case the total delay can be obtained by using *superposition*. Equations (11.80) through (11.82) are the essence of Jackson's theorem on open queueing networks.

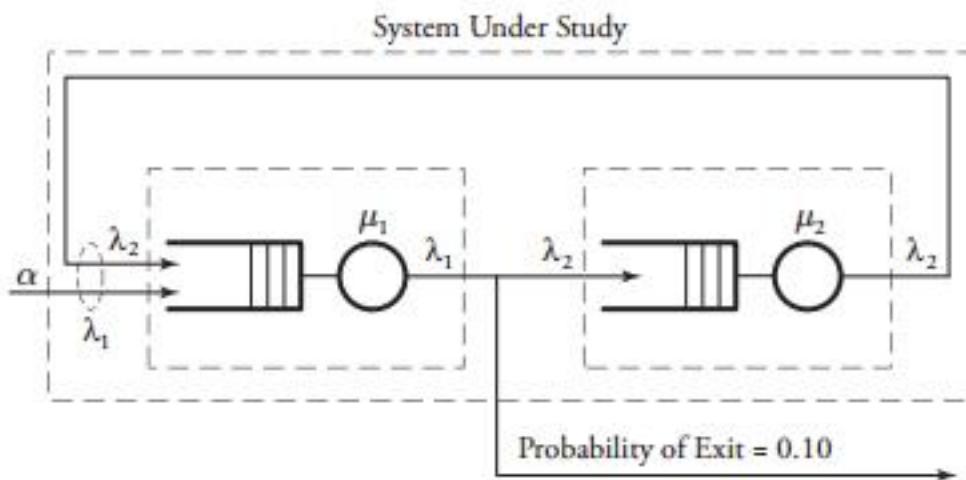


Figure 11.24 Two communication nodes with feedback

Example. Figure 11.24 shows a cascade of two $M/M/1$ nodes in which packets arrive at rate $\alpha = 10,000$ packets per second. Service rates of both nodes are $\mu_1 = \mu_2 = 200,000$ packets per second. At the output of node 1, packets exit with probability of $p = 0.10$; otherwise, they are driven into node 2 and then fed back to node 1. Find the estimated delay incurred on a packet, including the delay from circulations.

Solution. We first calculate the arrival rates to two nodes, λ_1 and λ_2 :

$$\lambda_1 = \lambda_2 + \alpha$$

and

$$\lambda_2 = (1 - p)\lambda_1.$$

By applying the values of $\alpha = 10,000$ and $p = 0.10$, this set of equations results in $\lambda_1 = 100,000$ and $\lambda_2 = 90,000$. The utilization for each system can now be derived as $\rho_1 = \lambda_1/\mu_1 = 0.5$ and $\rho_2 = \lambda_2/\mu_2 = 0.45$. According to $M/M/1$ discipline results obtained in the previous chapter, the average number of packets in each system is

$$E[K_1(t)] = \frac{\rho_1}{1 - \rho_1} = 1$$

and

$$E[K_2(t)] = \frac{\rho_2}{1 - \rho_2} = 0.82.$$

The delay is

$$E[T_{u1}] = \frac{E[K_1(t)]}{\lambda_1} = 0.01 \text{ msec},$$

$$E[T_{u2}] = \frac{E[K_2(t)]}{\lambda_2} = 0.009 \text{ msec},$$

and

$$E[T_f] = \frac{E[K_1(t)] + E[K_2(t)]}{\alpha} = 0.18 \text{ msec.}$$

11.7 Summary

This chapter presented the mathematical foundation for basic analysis of computer networks developing packet queues, using Markovian and non-Markovian cases. Single queues are the simplest models for the analysis of a single node in a communication network. *Little's theorem* relates the average number of packets and the average time per packet in the system to the arrival rate.

Birth-and-death processes can help identify the activity within a packet queue. Queueing node models with several scenarios were presented: finite versus infinite queueing capacity, one server versus several servers, and Markovian versus non-Markovian systems. The non-Markovian models are further classified into systems with non-Markovian services and systems with non-Markovian traffic arrivals. On a non-Markovian service model, we assume independent interarrivals and service times, which leads to the $M/G/1$ system.

Burke's theorem is the fundamental tool analyzing networks of packet queues. By Burke's theorem, we can present solutions to a network of several queues. We applied Burke's theorem in-series queueing nodes and in-parallel queueing nodes. *Jackson's theorem* provides a practical solution to situations in which a packet visits a particular queue more than once through *loops* or *feedback*. Open Jackson networks have a product-form solution for the joint queue length distribution in the steady state. We can derive several performance measures of interest, because nodes can be treated independently. Jackson networks with no external arrivals or departures forming a closed network also have a product-form solution for the steady-state joint queue length distribution.

The next chapter presents quality-of-service (QoS) in networks, another important issue of networking. Recall that QoS is a significant parameter in routing packets requiring performance.

11.8 Exercises

1. Each buffered output line of a data demultiplexer receives a block of packets every $20 \mu s$ and keeps them in the buffer. A sequencer checks each block for any packet misordering and corrects it, if necessary. It takes $10 \mu s$ to identify any packet misordering at each block and $30 \mu s$ to make the right sequence if there is any packet misordering. Suppose that a buffer is initially empty and that the numbers of misorderings in the first 15 blocks are $2, 4, 0, 0, 1, 4, 3, 5, 2, 4, 0, 2, 5, 2, 1$.
 - (a) Illustrate a plot of queueing behavior for the number of packet blocks in the demultiplexer's output line as a function of time.
 - (b) Find the mean number of packet blocks in each output line's queue.
 - (c) Determine the percentage of the time that a buffer is not empty.
2. A buffered router presented by an $M/M/1$ model receives Markovian traffic with a mean packet-arrival rate of $\lambda = 40/\text{sec}$ and an overall utilization of $\rho = 0.9$.
 - (a) Calculate the average number of packets in the queueing system.
 - (b) Determine the average time a packet remains in the queueing system.
 - (c) Sketch the Markov chain of the node's process.
3. For an $M/M/1$ system with service rate μ , the distribution of service time is exponential and is equal to $e^{-\mu t}$. In other words, by letting T be a random variable to represent the time until the next packet departure, $P[T > t] = e^{-\mu t}$. Now, consider an $M/M/a$ system with i servers busy at a time, equal service rate μ per server, and a similar definition of random variable T .
 - (a) Show T in terms of T_1, T_2, \dots , where T_i is a random variable representing the time until the next packet departure in server i .
 - (b) Find the distribution of service time, $P[T > t]$.
4. Consider a network node represented by an $M/M/1$ model.
 - (a) Find the probability that the number of packets in the node is less than a given number k , $P[K(t) < k]$. This is normally used to estimate the threshold time of the network node to reach a certain load.
 - (b) If we require that $P[K(t) < 20] = 0.9904$, determine the switching service rate if the arrival rate to this node is 300 packets per second.
5. For an $M/M/1$ queueing system:
 - (a) Find $P[K(t) \geq k]$, where k is a constant number.
 - (b) Determine the maximum allowable arrival rate in a system with service rate μ , if $P[K(t) \geq 60] = 0.01$ is required.

6. Consider a network node modeled by an $M/M/1$ queue in which a packet's willingness to join the queue is impacted by the queue size. A packet that finds $i \in \{0, 1, \dots\}$ other packets in the system joins the queue with probability $\frac{1}{i+1}$ and otherwise leaves immediately. If the arrival rate is λ and the mean service rate is μ :
- Sketch the state-transition diagram for this system.
 - Develop and solve the balance equations for the equilibrium probabilities p_i .
 - Show that a steady-state distribution exists.
 - Find the utilization of the server.
 - Find the throughput, the average number of packets in the system.
 - Find the average response time for a packet that decides to join.
7. In an $M/M/2$ communication node, packets arrive according to a Poisson process of rate 18 per second. The system has two parallel crossbar switches, and each switch spends an exponentially distributed amount of time with mean 100 ms to process a packet.
- Find the probability that an arriving packet must wait to be processed.
 - Find the mean number of packets in the system.
 - Find the mean time a packet spends in the system.
 - Find the probability that more than 50 packets are in the system.
8. Consider a high-speed node of a data network having parallel-plane switching fabrics, in which six packets can be processed at a time without any prior waiting time ($M/M/6/6$). Assume that the arrival rate is 100 packets/sec and that the mean service rate is 20 packets/sec.
- What is the probability of blocking a packet?
 - How many more switches are required to reduce the blocking probability to 50 percent more?
9. When a wireless user moves to a new cell, a handoff request for a new channel is needed. A handoff can be modeled with traffic type i assumption. When all the channels are used or none are available, the handoff call has to be terminated or blocked in the new base station. Classify the traffic into k types. Each call uses only one channel, and each channel uses only one radio channel. The handoff process at the new base station is modeled using an $M/M/c/c$ system: random interarrival call time, exponential holding time of a channel, c channels, and c handoff calls. Let λ_i be the handoff request rate for traffic type $i \in \{0, 1, \dots, k\}$, and let $1/\mu_i$ be the mean channel-exchange time for traffic type i . When j channels are busy,

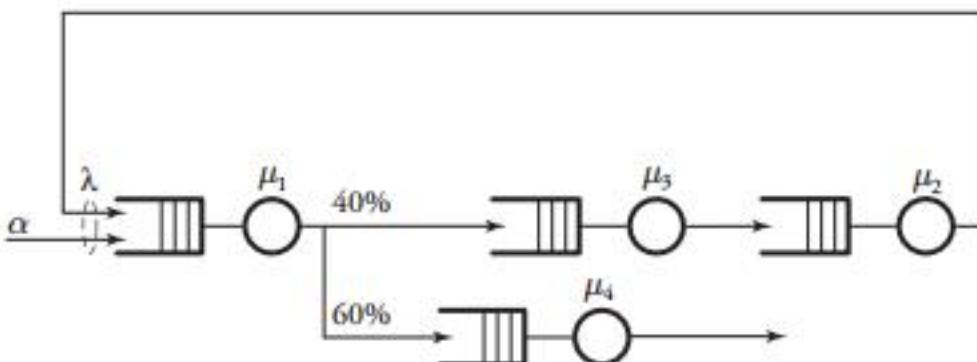


Figure 11.25 Exercise 11 network example

handoff calls depart at rate $j\mu_i$. When all c_i channels are in use, the channel-exchange rate is $c_i\mu_i$. In this case, any new arriving handoff calls are blocked.

- (a) Show a continuous-time Markov chain for the model.
 - (b) Derive a set of global balance equations.
 - (c) If P_0 is the probability that no channel exchange is requested for traffic type i , find P_j , the probability that j channel exchanges are requested for traffic type i .
 - (d) Find the handoff blocking probability, P_{c_i} .
10. Continuing the previous exercise, we want to obtain some statistical performance-evaluation results for the handoff.
- (a) Assume that the total available numbers of channels (c_i) are 10, 50, and 100, respectively; plot the handoff blocking probability with a choice of three mean holding times of 10 ms, 20 ms, and 30 ms.
 - (b) Discuss the results obtained in the plots.
11. Consider a router containing four networked queueing nodes, each represented by an $M/M/1$ model, as shown in Figure 11.25. Any percentage number shown on each branch expresses the share of the branch from the traffic coming to its branching point. The arrival rate to the system is $\alpha = 20$ packets per ms. The service rates in these nodes are $\mu_1 = 100$, $\mu_2 = 100$, $\mu_3 = 20$, and $\mu_4 = 30$ packets per ms, respectively.
- (a) Find the arrival rate to each of the four queueing units.
 - (b) Find the average number of packets in each unit.
 - (c) Find the average delay for a packet from its arrival to the system until it exits the system.

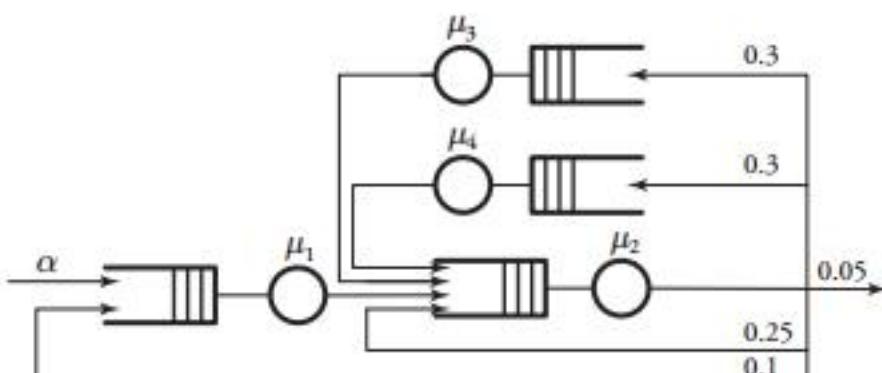


Figure 11.26 Exercise 12 network example

12. Figure 11.26 shows a network of four data switching nodes modeled by four $M/M/1$ systems. The arrival rate to this network is λ . The outgoing packets get distributed over the outgoing link and feedback paths with the probabilities indicated on the figure. The arrival rate to the system is $\alpha = 200$ packets per second, and the service rates are $\mu_1 = 4$, $\mu_2 = 3$, $\mu_3 = 5$, and $\mu_4 = 2$ packets per ms.
- Find the arrival rate to each of the four queueing units.
 - Find the average number of packets in each unit.
 - Find the average system delay for a packet.
13. A network of four switching nodes is modeled by four $M/M/1$ systems, as shown in Figure 11.27. The arrival rate to the network is $\alpha = 100$ packets per second. The outgoing packets get distributed over the outgoing link and feedback paths with probabilities given on the figure.
- Find the arrival rate to each of the four queueing units.
 - Find the average number of packets in each unit.
 - Find the average system delay for a packet.
14. A network of six switching nodes is modeled by six $M/M/1$ systems, as shown in Figure 11.28. Each of the five parallel nodes receives a fair share of traffic. The arrival rate to the system is $\alpha = 200$ packets per second, the service rate for the single node is $\mu_0 = 100$ packets per ms, and the service rate for each of the five nodes is $\mu_i = 10$ packets per ms.
- Find the arrival rate to each of the six queueing units.
 - Find the average number of packets in each unit.
 - Find the average system delay for a packet.

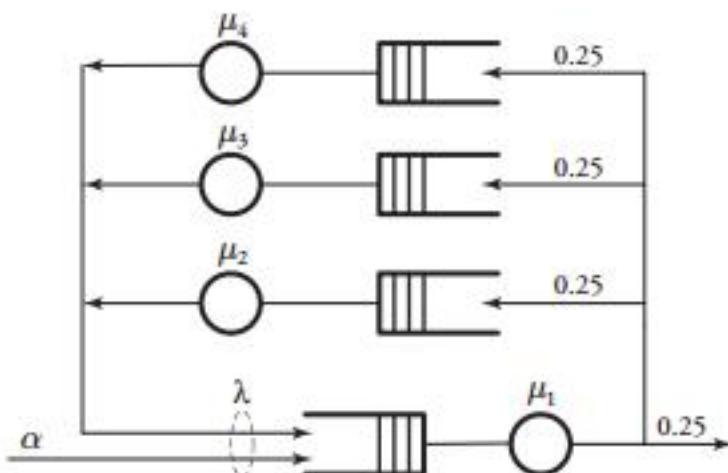


Figure 11.27 Exercise 13 network example

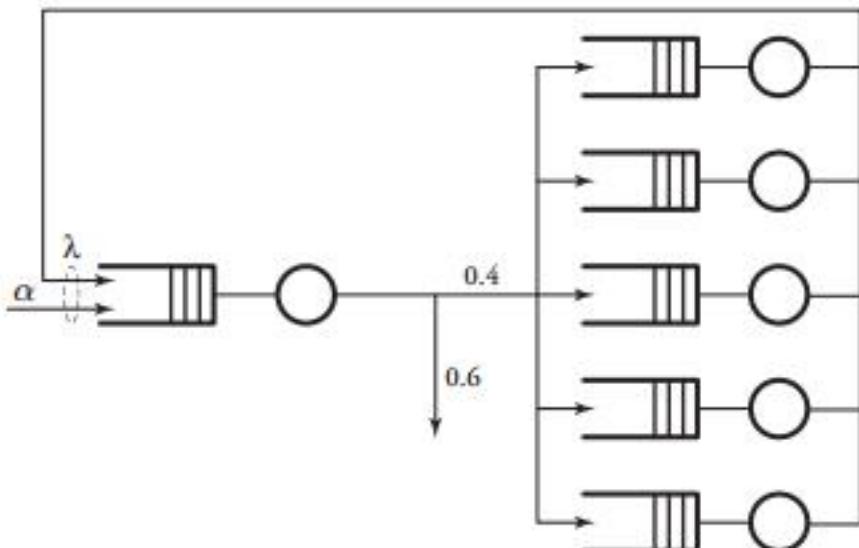


Figure 11.28 Exercise 14 network example

15. A network of four routers is modeled by four $M/M/1$ systems, as shown in Figure 11.29. Outgoing packets get distributed over the outgoing link and feedback paths with the probabilities indicated on the figure. The arrival rate to the system is $\alpha = 800$ packets per second, and the service rates are $\mu_1 = 10$, $\mu_2 = 12$, $\mu_3 = 14$, and $\mu_4 = 16$ packets per ms.
- Find the arrival rate to each of the four queueing units.
 - Find the average number of packets in each unit.
 - Find the average system delay for a packet.

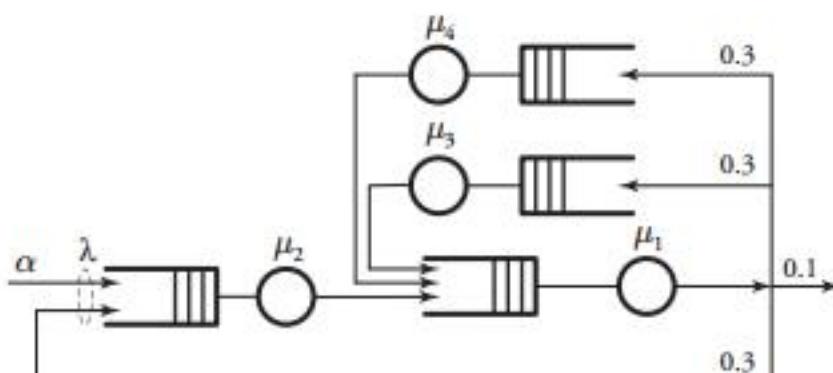


Figure 11.29 Exercise 16 network example

16. *Computer simulation project.* Write a computer program to simulate an input buffer of a router. Consider $K = 64$ buffer slots; each slot can fit only in a packet of size 1,000 bytes.
 - (a) Construct and simulate 1 bit of memory.
 - (b) Extend your program to construct 1,000 bytes of memory.
 - (c) Further extend your program to simulate all 64 buffer slots.
 - (d) Dynamically assign packets of different sizes every 1 ms, and send out the packet every t seconds. Measure the performance metrics of this buffer given different values of t
17. *Computer simulation project.* Carry the program you developed for a single buffer in the previous project and extend it to simulate two connected buffers: one with $K_1 = 64$ buffer slots and the other one with $K_2 = 128$ buffer slots. Each slot can fit only in a packet of size 1,000 bytes. Dynamically assign packets of different size every 1 ms, and send out the packet every t_1 seconds from the first buffer and every t_2 seconds from the second buffer:
 - (a) Construct and simulate each of the two buffers individually.
 - (b) Integrate these two buffers and measure the performance metrics of both buffers together, given different values of t_1 and t_2 .

CHAPTER 12

Quality of Service and Resource Allocation

Recall from Chapter 3 that the main portion of the port processor interface in routers supports *quality of service* (QoS). This chapter explores all angles of QoS and the numerous methods of providing it at various protocol stack layers. This chapter covers the following topics:

- *Overview of QoS*
- *Integrated services QoS*
- *Traffic shaping*
- *Admission control*
- *Reservation protocol (RSVP)*
- *Differentiated services QoS*
- *Resource allocation*

Two broad approaches to QoS are *integrated services* and *differentiated services*. Integrated services provide QoS to individual applications and flow records. Providing QoS requires certain features to be maintained in switching nodes. QoS protocols govern traffic shaping and packet scheduling. Traffic shaping regulates the spacing between incoming packets. Other advanced QoS protocols covered are admission control and RSVP.

The differentiated services provide QoS support to a broad class of applications. The basics of resource allocation for packet-switched networks are reviewed, as is resource

allocation for all possible layers of the protocol stack. Resource-allocation algorithms can be used to avoid possible ATM congestion.

12.1 Overview of QoS

Communication networks face a variety of quality-of-service demands. The main motivation for a QoS unit in a data network port processor is to control access to available bandwidth and to regulate traffic. Traffic regulation traffic is always necessary in WANs in order to avoid congestion. A network must be designed to support both real-time and non-real-time applications. Voice and video transmissions over IP must be able to request a higher degree of assurance from the network. A network that can provide these various levels of services requires a more complex structure.

The provision of QoS to a network either does or does not come with a guarantee. Nonguaranteed QoS is typically based on the *best-effort model*, whereby a network provides no guarantees on the delivery of packets but makes its best effort to do so. In a non-real-time application, a network can use the retransmit strategy for successful data delivery. However, in a real-time application, such as voice or video networking, where timely delivery of packets is required, the application requires a low-latency communication. Consequently, the network must be able to handle packets of such applications more carefully.

Approaches to providing quality support can be further divided into *integrated services* or *differentiated services*. The details of these two approaches are discussed in Sections 12.2 and 12.3, respectively.

12.2 Integrated Services QoS

The *integrated services approach*, consisting of two service classes, defines both the service class and mechanisms that need to be used in routers to provide the services associated with the class. The first service class, the *guaranteed service class*, is defined for applications that cannot tolerate a delay beyond a particular value. This type of service class can be used for real-time applications, such as voice or video communications. The second, *controlled-load service class*, is used for applications that can tolerate some delay and loss. Controlled-load service is designed such that applications run very well when the network is not heavily loaded or congested. These two service classes cover the wide range of applications on the Internet.

A network needs to obtain as much information as possible about the flow of traffic to provide optimum services to the flow. This is true especially when real-time services to

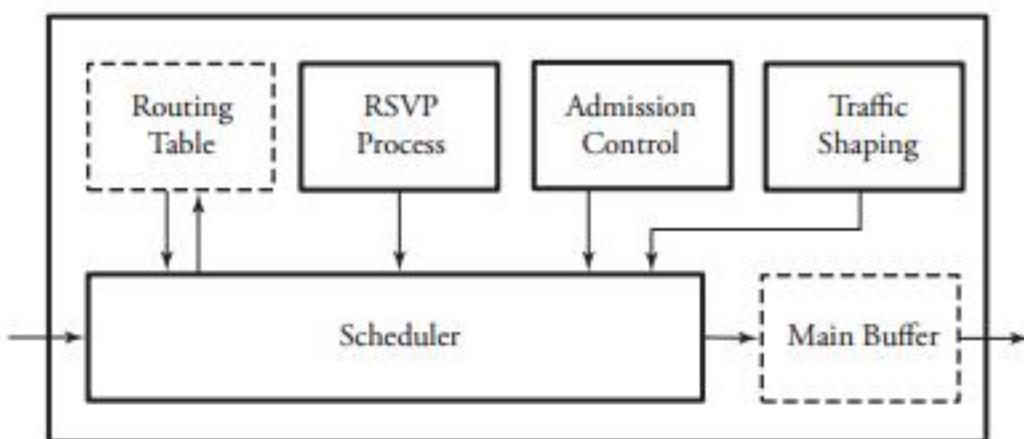


Figure 12.1 Overview of QoS methods in integrated services

an application are requested. Any application request to a network has to first specify the type of service required, such as controlled load or guaranteed service. An application that requires guaranteed service also needs to specify the maximum delay that it can tolerate. Once the delay factor is known to a service provider, the QoS unit of the node must determine the necessary processes to be applied on incoming flows. Figure 12.1 shows four common categories of processes providing quality of service.

1. *Traffic shaping* regulates turbulent traffic.
2. *Admission control* governs whether the network, given information about an application's flow, can admit or reject the flow.
3. *Resource allocation* lets network users reserve bandwidth on neighboring routers.
4. *Packet scheduling* sets the timetable for the transmission of packet flows. Any involving router needs to queue and transmit packets for each flow appropriately.

The widespread deployment of the integrated services approach has been deterred owing to scalability issues. As the network size increases, routers need to handle larger routing tables and switch larger numbers of bits per second. In such situations, routers need to refresh information periodically. Routers also have to be able to make admission-control decisions and queue each incoming flow. This action leads to scalability concerns, especially when networks scale larger.

12.2.1 Traffic Shaping

Realistically, spacing between incoming packets has an irregular pattern, which in many cases causes congestion. The goal of *traffic shaping* in a communication network is to control access to available bandwidth to regulate incoming data to avoid congestion,

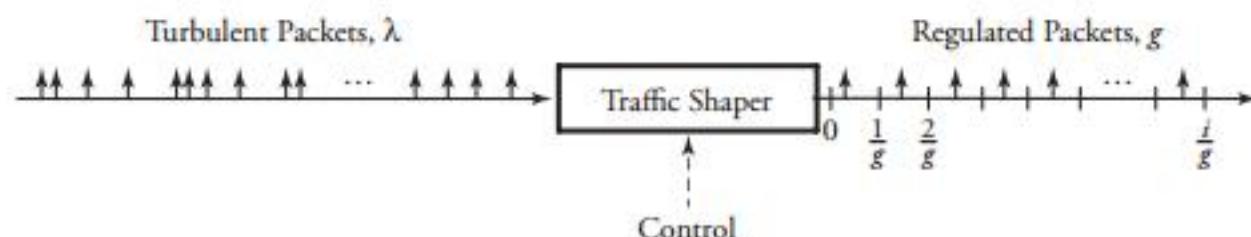


Figure 12.2 Traffic shaping to regulate any incoming turbulent traffic

and to control the delay incurred by packets (see Figure 12.2). Turbulent packets at rate λ and with irregular arrival patterns are regulated in a traffic shaper over equal-sized $1/g$ intervals.

If a policy dictates that the packet rate cannot exceed a specified rate even though the network node's access rates might be higher, a mechanism is needed to smooth out the rate of traffic flow. If different traffic rates are applied to a network node, the traffic flow needs to be regulated. (Monitoring the traffic flow is called *traffic policing*.) Traffic shaping also prevents packet loss by preventing the sudden increased usage of system bandwidth. The stochastic model of a traffic shaper consists of a system that converts any form of traffic to a deterministic one. Two of the most popular traffic-shaping algorithms are *leaky bucket* and *token bucket*.

Leaky-Bucket Traffic Shaping

This algorithm converts any turbulent incoming traffic into a smooth, regular stream of packets. Figure 12.3 shows how this algorithm works. A leaky-bucket interface is connected between a packet transmitter and the network. No matter at what rate packets enter the traffic shaper, the outflow is regulated at a constant rate, much like the flow of water from a leaky bucket. The implementation of a leaky-bucket algorithm is not difficult.

At the heart of this scheme is a finite queue. When a packet arrives, the interface decides whether that packet should be queued or discarded, depending on the capacity of the buffer. The number of packets that leave the interface depends on the protocol. The packet-departure rate expresses the specified behavior of traffic and makes the incoming bursts conform to this behavior. Incoming packets are discarded once the bucket becomes full.

This method directly restricts the maximum size of a burst coming into the system. Packets are transmitted as either fixed-size packets or variable-size packets. In the fixed-size packet environment, a packet is transmitted at each clock tick. In the variable-size packet environment, a fixed-sized block of a packet is transmitted. Thus, this algorithm

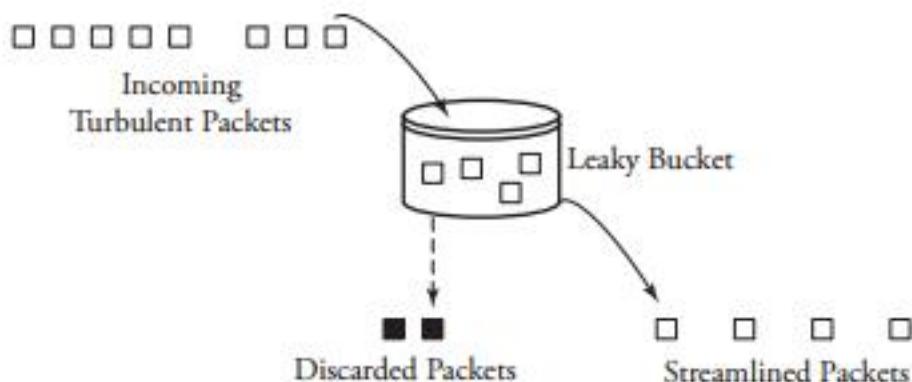


Figure 12.3 The leaky-bucket traffic-shaping algorithm

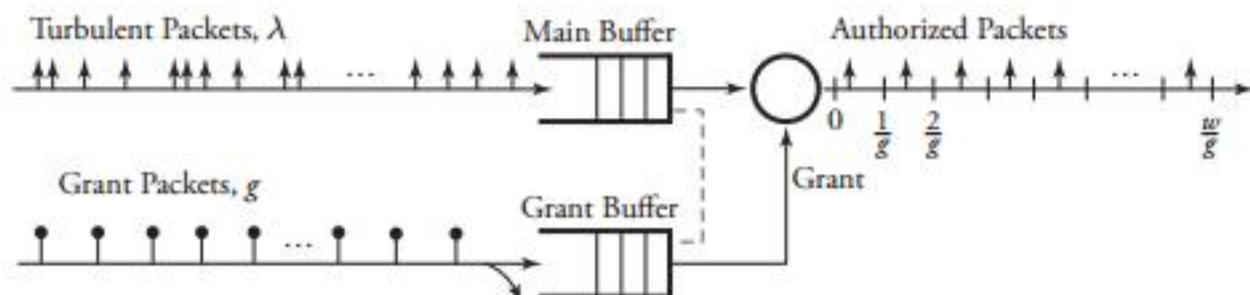


Figure 12.4 Queueing model of leaky-bucket traffic-shaping algorithm

is used for networks with variable-length packets and also equal-sized packet protocols, such as ATM.

The leaky-bucket scheme is modeled by two main buffers, as shown in Figure 12.4. One buffer forms a queue of incoming packets, and the other one receives authorizations. The leaky-bucket traffic-shaper algorithm is summarized as follows.

Begin Leaky-Bucket Algorithm

1. Define for the Algorithm:

λ = rate at which packets with irregular rate arrive at the *main buffer*

g = rate at which *authorization grants* arrive at the *grant buffer*

w = size of the grant buffer and can be dynamically adjusted

2. Every $1/g$ seconds, a grant arrives.

3. Over each period of $1/g$ seconds, i grants can be assigned to the first i incoming packets, where $i \leq w$, and packets exit from the queue one at a time every $1/g$ seconds, totaling i/g seconds.

4. If more than w packets are in the main buffer, only the first w packets are assigned grants at each window time of $1/g$, and the rest remain in the main queue to be examined in the next $1/g$ interval.
5. If no grant is in the grant buffer, packets start to be queued.

With this model, w is the bucket size. The bucket in this case is the size of the window that the grant buffer opens to allow w packets from the main buffer to pass. This window size can be adjusted, depending on the rate of traffic. If w is too small, the highly turbulent traffic is delayed by waiting for grants to become available. If w is too large, long turbulent streams of packets are allowed into the network. Consequently, with the leaky-bucket scheme, it would be best for a node to dynamically change the window size (bucket size) as needed. The dynamic change of grant rates in high-speed networks may, however, cause additional delay through the required feedback mechanism.

Figure 12.5 shows the Markov chain state diagram (see Section C.5) depicting the activity of grant generation in time. At the main buffer, the mean time between two consecutive packets is $1/\lambda$. At the grant buffer, a grant arrives at rate g . Hence, every $1/g$ seconds, a grant is issued to the main buffer on times $0, 1/g, 2/g, \dots$. If the grant buffer contains w grants, it discards any new arriving grants. A state $i \in \{0, 1, \dots, w\}$ of the Markov chain refers to the situation in which i grants are allocated to i packets in the main buffer, and thus $w - i$ grants are left available. In this case, the i packets with allocated grants are released one at a time every $1/g$ seconds. When the packet flow is slow, the grant queue reaches its full capacity, and thus grants ($w + 1$) and beyond are discarded.

We define the following main variables for delay analysis:

- $P_X(x)$ = The probability of x packet arrivals in $1/g$ seconds
- P_i = The probability that a grant has arrived or that the Markov chain is in state i
- P_{ji} = The transition probability from any state j to a state i on the Markov chain

As shown in Figure 12.5, the chain starts at state 0, implying that w grants are available in the grant buffer and that no packet is in the main buffer. In this state, the first arriving packet to the main buffer with probability $P_X(1)$ gets a grant while a new grant arrives in the same period. This creates no changes in the state of the chain. Therefore, the transition probability at state 0 has two components, as follows:

$$P_{00} = P_X(0) + P_X(1). \quad (12.1)$$

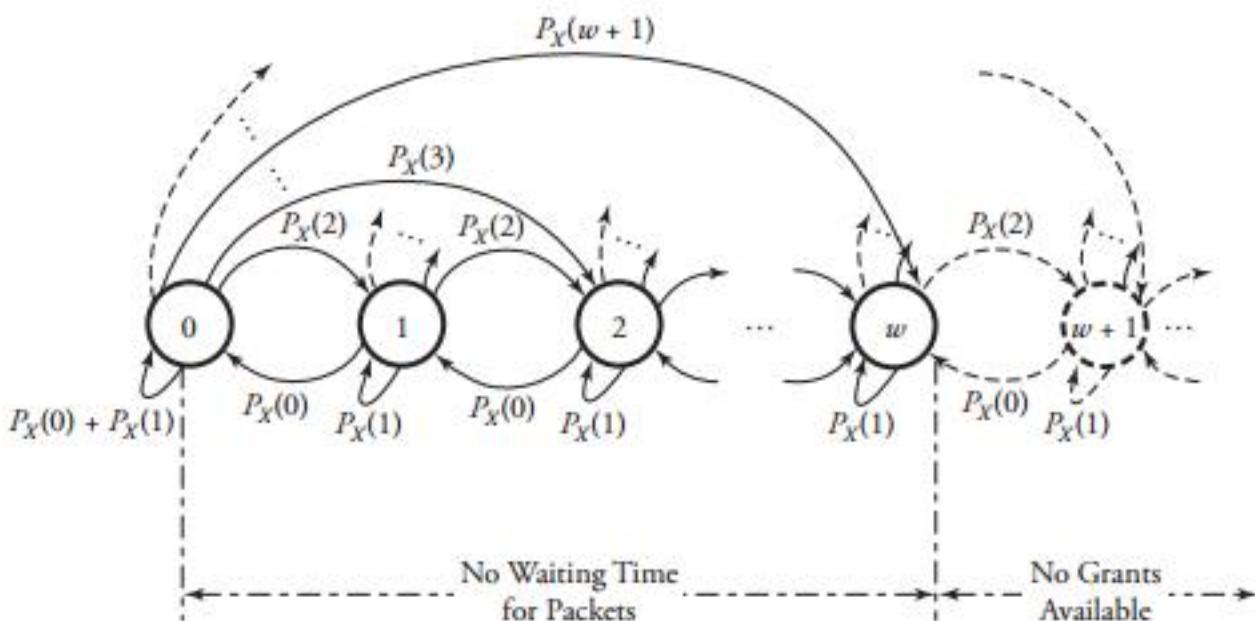


Figure 12.5 State diagram modeling the leaky-bucket traffic-shaping algorithm

This means that P_{00} is equal to the probability that no packet or only one packet arrives ($P_X(0) + P_X(1)$). As long as $i \geq 1$, state 0 can be connected to any state $i \leq w$, inducing the property of the queueing system in Figure 12.5 by

$$P_{0i} = P_X(i+1) \quad \text{for } i \geq 1. \quad (12.2)$$

For example, two new packets arriving during a period $1/g$ with probability $P_X(2)$ change state 0 to state 1 so that one of the two packets is allocated a grant, and thus the chain moves to state 1 from state 0, and the second packet is allocated the new arriving grant during that period of time. The remaining transition probabilities are derived from

$$P_{ji} = \begin{cases} P_X(i-j+1) & \text{for } 1 \leq j \leq i+1 \\ 0 & \text{for } j > i+1 \end{cases}. \quad (12.3)$$

Now, the global balance equations can be formed. We are particularly interested in the probability of any state i denoted by P_i . The probability that the chain is in state 0, P_0 , implying that no grant has arrived, is the sum of incoming transitions:

$$P_0 = P_X(0)P_1 + [P_X(0) + P_X(1)]P_0. \quad (12.4)$$

For P_1 , we can write

$$P_1 = P_X(2)P_0 + P_X(1)P_1 + P_X(0)P_2. \quad (12.5)$$

For all other states, the probability of the state can be derived from the following generic expression:

$$P_i = \sum_{j=0}^{i+1} P_X(i-j+1)P_j \quad \text{for } i \geq 1. \quad (12.6)$$

The set of equations generated from Equation (12.6) can be recursively solved. Knowing the probability of each state at this point, we can use Little's formula (see Section 11.1) to estimate the average waiting period to obtain a grant for a packet, $E[T]$, as follows:

$$E[T] = \frac{\sum_{i=w+1}^{\infty} (i-w)P_i}{g}. \quad (12.7)$$

It is also interesting to note that the state of the Markov chain can turn into "queueing of packets" at any time, as shown by dashed lines in Figure 12.5. For example at state 0, if more than $w + 1$ packets arrive during $1/g$, the state of the chain can change to any state after w , depending on the number of arriving packets during $1/g$. If this happens, the system still assigns $w + 1$ grants to the first $w + 1$ packets and assigns no grant to the remaining packets. The remaining packets stay pending to receive grants.

Example. The probability that x packet arrives in $1/g$ seconds is obtained by

$$P_X(x) = \frac{(\frac{\lambda}{g})^x e^{-\frac{\lambda}{g}}}{x!}.$$

Find the probability that the system has not issued any grants.

Solution. We give the first final result, which starts at P_0 , and leave the rest of this recursive effort to the reader. The final form of P_0 , using Equation (12.4) and $P_X(x)$, starts at:

$$P_0 = \frac{g - \lambda}{g P_X(0)},$$

where this equation can be rearranged to $g = \lambda / (1 - P_0 P_X(0))$. We can then state that the system is considered stable if $\lambda < g$ or $P_0 P_X(0) > 1$.

Token-Bucket Traffic Shaping

Traffic flow for most applications varies, with traffic bursts occurring occasionally. Hence, the bandwidth varies with time for these applications. However, the network must be informed about the bandwidth variation of the flow in order to perform

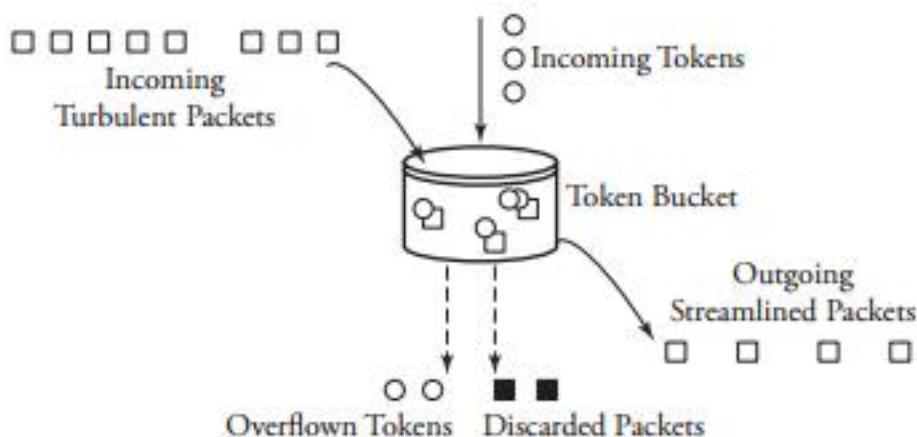


Figure 12.6 The token-bucket traffic-shaping method

admission control and to operate more efficiently. Two parameters—the *token arrival rate*, v , and the *bucket depth*, b —together describe the operation of the token-bucket traffic shaper. The token rate, v , is normally set to the average traffic rate of the source. The bucket depth, b , is a measure of the maximum amount of traffic that a sender can send in a burst.

With the *token-bucket traffic-shaping* algorithm, flow traffic is shaped efficiently as shown in Figure 12.6. The token-bucket shaper consists of a buffer, similar to a water bucket, that accepts fixed-size *tokens* of data generated by a token generator at a constant rate every clock cycle. Packets with any unpredictable rate must pass through this token-bucket unit. According to this protocol, a sufficient number of tokens are attached to each incoming packet, depending on its size, to enter the network. If the bucket is full of tokens, additional tokens are discarded. If the bucket is empty, incoming packets are delayed (buffered) until a sufficient number of tokens is generated. If the packet size is too big, such that there are not enough tokens to accommodate it, a delay of input packets is carried over.

The best operation of this algorithm occurs when the number of tokens is larger than the number of incoming packet sizes. This is not a big flaw, as tokens are periodically generated. The output rate of packets attached with tokens obeys the clock, and so the corresponding flow becomes a regulated traffic stream. Thus, by changing the clock frequency and varying the token-generation rate, we can adjust the input traffic to an expected rate.

Comparison of Leaky-Bucket and Token-Bucket Approaches

Both algorithms have their pros and cons. The token-bucket algorithm enforces a more flexible output pattern at the average rate, no matter how irregular the incoming

traffic is. The leaky-bucket method enforces a more rigid pattern. Consequently, the token bucket is known as a more flexible traffic-shaping method but has greater system complexity. The token-bucket approach has an internal clock that generates tokens with respect to the clock rate. A queueing regulator then attaches incoming packets to tokens as packets arrive, detaches tokens as packets exit, and, finally, discards the tokens. This process adds considerable complexity to the system. In the leaky-bucket approach, no virtual tokens are seen, greatly increasing the speed of operation and enhancing the performance of the communication node. Our primary goal of coming up with a high-speed switching network would thus be easily attained.

12.2.2 Admission Control

The *admission-control* process decides whether to accept traffic flow by looking at two factors:

1. r_s = type of service requested
2. t_s = required bandwidth information about the flow

For controlled-load services, no additional parameters are required. However, for guaranteed services, the maximum amount of delay must also be specified. In any router or host capable of admission control, if currently available resources can provide service to the flow without affecting the service to other, already admitted flows, the flow is admitted. Otherwise, the flow is rejected permanently. Any admission-control scheme must be aided by a policing scheme. Once a flow is admitted, the policing scheme must make sure that the flow conforms to the specified t_s . If not, packets from these flows become obvious candidates for packet drops during a congestion event. A good admission-control scheme is vital to avoid congestion.

12.2.3 Resource Reservation Protocol (RSVP)

The *Resource Reservation Protocol* (RSVP) is typically used to provide real-time services over a connectionless network. RSVP is a soft-state protocol so that it can handle link failure efficiently. The protocol adopts a receiver-oriented approach to resource reservation. This process is required to provide the desired QoS for an application. RSVP supports both unicast and multicast flows. Unlike connection-oriented resource set-up mechanisms, RSVP can adapt to router failures by using an alternative path.

In order for a receiver to make a reservation at intermediate routers, the sender initially sends the t_s message, which passes through each intermediate router before reaching the receiver. This way, the receiver becomes aware of the information on the flow and the path and makes a reservation at each router. This message is sent

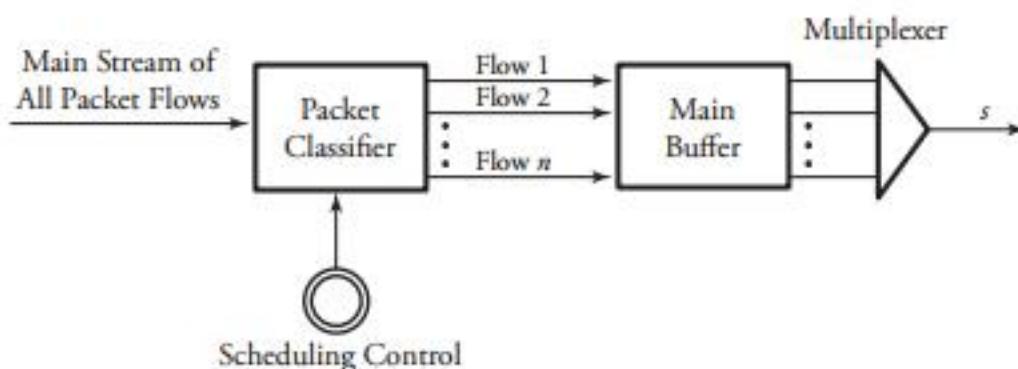


Figure 12.7 Overview of a packet scheduler

periodically to maintain the reservation. The router can accept or deny the reservation, based on its available resources. Also, the t_s message is periodically refreshed to adapt to link failures. When a particular link fails, the receiver receives this message over a different path. The receiver can then use this new path to establish a new reservation; consequently, the network operation continues as usual.

Reservation messages from multiple receivers can be merged if their delay requirements are similar. If a receiver needs to receive messages from multiple senders, it requests a reservation meeting the total of t_s requirements of all senders. Thus, RSVP uses a receiver-oriented approach to request resources to meet the QoS requirements of various applications.

12.2.4 Packet Scheduling

RSVP enables network users to reserve resources at the routers. Once the resources have been reserved, a *packet-scheduling* mechanism has to be in place to provide the requested QoS. Packet scheduling involves managing packets in queues to provide the QoS associated with the packet, as shown in Figure 12.7. A *packet classifier* is the heart of scheduling scheme and performs on the basis of the header information in the packet. Packet classifying involves identifying each packet with its reservation and ensuring that the packet is handled correctly.

Packets are classified according to the following parameters:

- Source/destination IP address
- Source/destination port
- Packet flow priority
- Protocol type
- Delay sensitivity

Based on that information, packets can be classified into those requiring guaranteed services and those requiring controlled-load services. Designing optimal queueing mechanisms for each of these categories is important in providing the requested QoS. For provision of a guaranteed service, *weighted-fair queueing* is normally used. This type of queueing is very effective. For controlled-load services, simpler queueing mechanisms can be used.

Packet scheduling is necessary for handling various types of packets, especially when *real-time packets* and *non-real-time packets* are aggregated. Real-time packets include all delay-sensitive traffic, such as video and audio data, which have a low delay requirement. Non-real-time packets include normal data packets, which do not have a delay requirement. Packet scheduling has a great impact on the QoS guarantees a network can provide. If a router processes packets in the order in which they arrive, an aggressive sender can occupy most of the router's queueing capacity and thus reduce the quality of service. Scheduling ensures fairness to different types of packets and provides QoS. Some of the commonly used scheduling algorithms that can be implemented in the input port processors (IPPs) and output port processors (OPPs) of routers or main hosts are discussed in the next sections.

First-In, First-Out Scheduler

Figure 12.8 shows a simple scheduling scheme: *first in, first out* (FIFO). With this scheme, incoming packets are served in the order in which they arrive. Although FIFO is very simple from the standpoint of management, a pure FIFO scheduling scheme provides no fair treatment to packets. In fact, with FIFO scheduling, a higher-speed user can take up more space in the buffer and consume more than its fair share of bandwidth. FIFO is simple to implement from the hardware standpoint and is still the most commonly implemented scheduling policy, owing to its simplicity. With smart buffer-management schemes, it is possible to control bandwidth sharing among different classes and traffic. Delay bound of this scheduler, T_q , is calculated based on the queueing buffer size, as follows:

$$T_q \leq \frac{K}{s}, \quad (12.8)$$

where K is the maximum buffer size, and s is the outgoing link speed. Certain control algorithms, RSVP, can be used to reserve the appropriate link capacity for a particular traffic flow. However, when high-speed links are active, tight delay bounds are required by real-time applications.

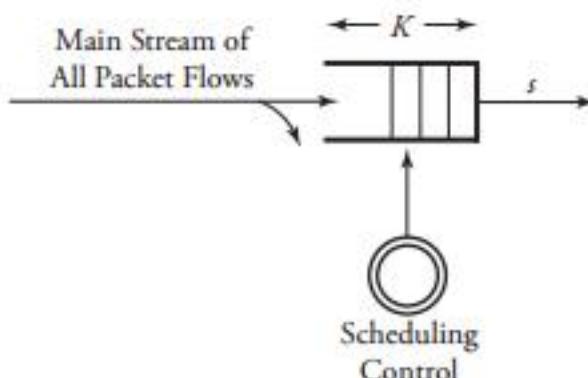


Figure 12.8 A typical FIFO queueing scheduler

Priority Queueing Scheduler

The *priority queueing* (PQ) scheduler combines the simplicity of FIFO schedulers with the ability to provide service classes. With various priority queueing, packets are classified on the priority of service indicated in the packet headers. Figure 12.9 shows a simple model of this scheduler. Lower-priority queues are serviced only after all packets from the higher-priority queues are serviced. Only a queue with the highest priority has a delay bound similar to that with FIFO. Lower-priority queues have a delay bound that includes the delays incurred by higher-priority queues. As a result, queues with lower priorities are subject to bandwidth starvation if the traffic rates for higher-priority queues are not controlled.

In this scheduling scheme, packets in lower-priority queues are serviced after all the packets from higher-priority queues are transmitted. In other words, those queues with lower priorities are subject to severely limited bandwidth if traffic rates for the higher-priority queues are not controlled.

Priority queueing is either *nonpreemptive* or *preemptive*. In order to analyze these two disciplines, we need to define a few common terms. For a queue with flow i (class i queue), let λ_i , $1/\mu_i$, and $\rho_i = \lambda_i/\mu_i$ be the arrival rate, mean service rate, and mean offered load (utilization), respectively. We also express a lower value of i for a higher-priority class.

Non-Preemptive Priority Queues. In nonpreemptive priority-queueing schemes, the process of lower-priority packets cannot be interrupted under any condition. Every time an in-service packet terminates from its queue, the waiting packet from the highest-priority queue enters service. This fact makes the system simple from the

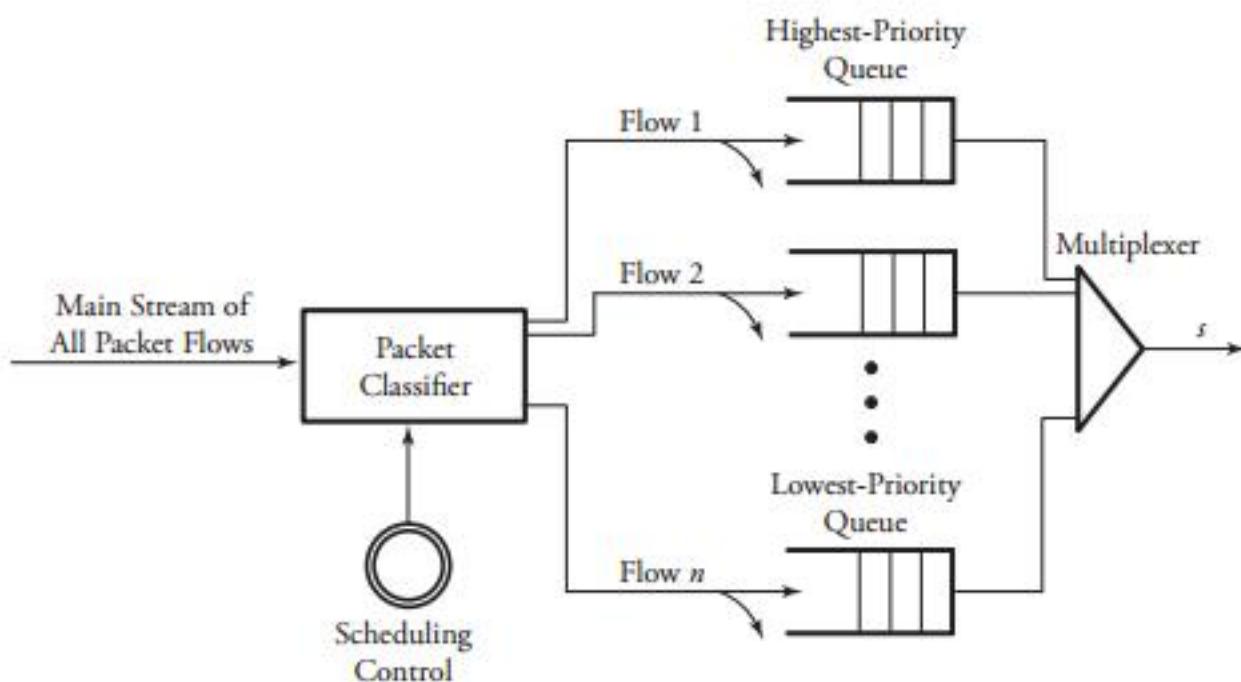


Figure 12.9 A typical priority-queueing scheduler

implementation standpoint. Let $E[T_i]$ be the mean waiting time for a packet in flow i (class i) queue. This total queuing delay has three components, as follows:

1. The mean waiting time for any class i packet until any in-service class j packet among n classes terminates. We denote this time by W_x .
2. The mean time until a packet from a class i or lower (higher-priority) waiting ahead is serviced. We denote this time by $E[T_{q,i}]_2$.
3. The mean time owing to higher-priority packets arriving while the current packet is waiting and being serviced before the current packet. We denote this time by $E[T_{q,i}]_3$.

Hence, $E[T_{q,i}]$ can be the sum of all these three components:

$$E[T_{q,i}] = W_x + E[T_{q,i}]_2 + E[T_{q,i}]_3. \quad (12.9)$$

The first component is computed by using the mean residual service time, r_j , for a class j packet currently in service as:

$$W_x = \sum_{j=1}^n \rho_j r_j. \quad (12.10)$$

Using Little's formula, we can derive the second component, $E[T_{q,i}]_2$, by

$$E[T_{q,i}]_2 = \sum_{j=1}^i \frac{E[K_{q,j}]}{\mu_j}, \quad (12.11)$$

where $E[K_{q,j}]$ is the mean number of waiting packets and μ_j is the mean service rate for class j packets. Since we can also write $E[T_{q,i}] = E[K_{q,i}]/\mu_i$, we can rewrite Equation (12.11) as

$$E[T_{q,i}]_2 = \sum_{j=1}^i \rho_j E[T_{q,j}]. \quad (12.12)$$

Similarly, the third component, $E[T_{q,i}]_3$, can be developed by

$$E[T_{q,i}]_3 = \sum_{j=1}^{i-1} E[T_{q,i}] \left(\frac{\lambda_j}{\mu_j} \right) = E[T_{q,i}] \sum_{j=1}^{i-1} \rho_j. \quad (12.13)$$

By incorporating Equations (12.10), (12.12), and (12.13) into Equation (12.9), we obtain

$$E[T_{q,i}] = \sum_{j=1}^n \rho_j r_j + \sum_{j=1}^i \rho_j E[T_{q,j}] + E[T_{q,i}] \sum_{j=1}^{i-1} \rho_j. \quad (12.14)$$

Note that the total system delay for a packet passing queue i and the server—in this case, the server is the multiplexer—is

$$E[T_i] = E[T_{q,i}] + \frac{1}{\mu_i}. \quad (12.15)$$

Recall that the $E[T_i]$ expression corresponds to Equation (11.22).

Preemptive Priority Queues. A preemptive-priority queue is a discipline in which service in a lower-priority packet can be interrupted by an incoming higher-priority packet. Let $E[T_i]$ be the mean waiting time for a packet in flow i (class i) queue. Figure 12.10 shows an example in which a class i packet has been interrupted three times by higher-priority packets. This total queueing delay has four components. The first three are identical to those for the nonpreemptive case. The fourth component, θ_i , is the total mean completion time for the current class i packet when it is preempted in the server by higher-priority packets (classes $i - 1$ to 1). Therefore, by including θ_i into

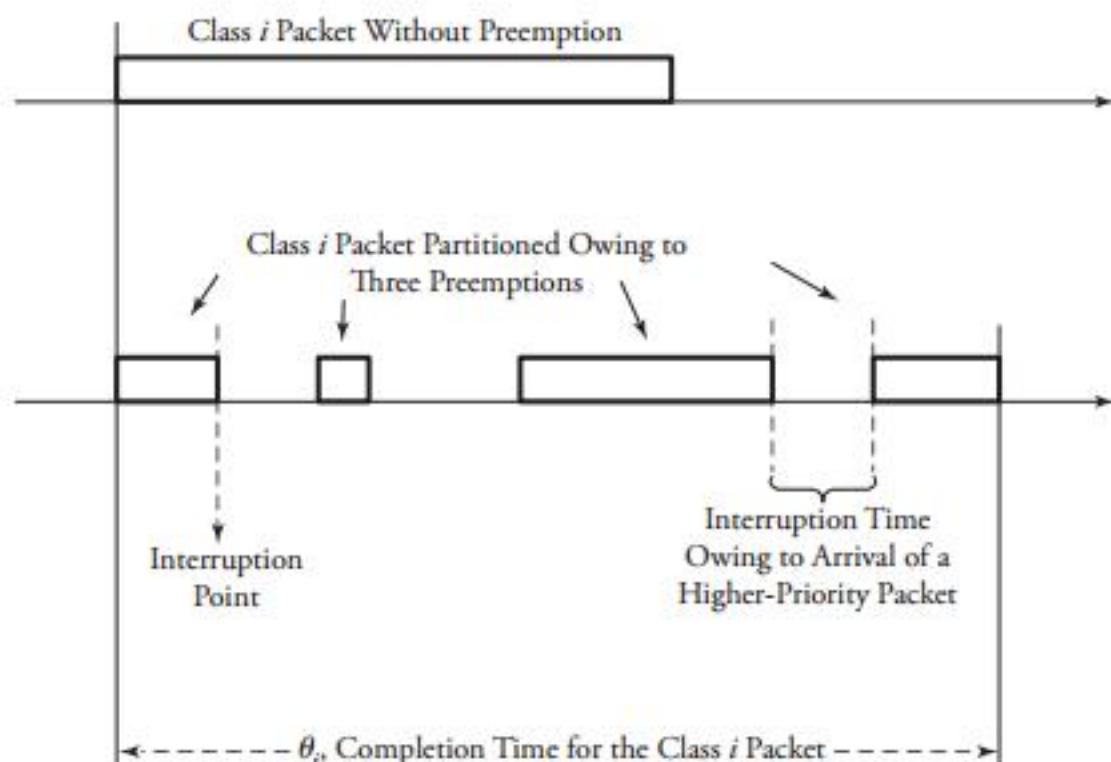


Figure 12.10 Preemptive-priority queueing. A class i packet has been interrupted three times by higher-priority packets.

Equation (12.15), we calculate the total system delay for a packet passing queue i and the server as

$$E[T_i] = E[T_{q,i}] + \left(\frac{1}{\mu_i} + \theta_i \right). \quad (12.16)$$

During θ_i , as many as $\lambda_j \theta_i$ new higher-priority packets of class j arrive, where $j \in \{1, 2, \dots, i-1\}$. Each of these packets delays the completion of our current class i packet for $1/\mu_j$. Therefore, θ_i , which also includes the service time of the class i packet, as seen in Figure 12.10, can be clearly realized as

$$\theta_i = \frac{1}{\mu_i} + \sum_{j=1}^{i-1} \frac{1}{\mu_j} (\lambda_j \theta_i). \quad (12.17)$$

From Equation (12.17), we can compute θ_i as

$$\theta_i = \frac{1}{\mu_i \left(1 - \sum_{j=1}^{i-1} \rho_j \right)}. \quad (12.18)$$

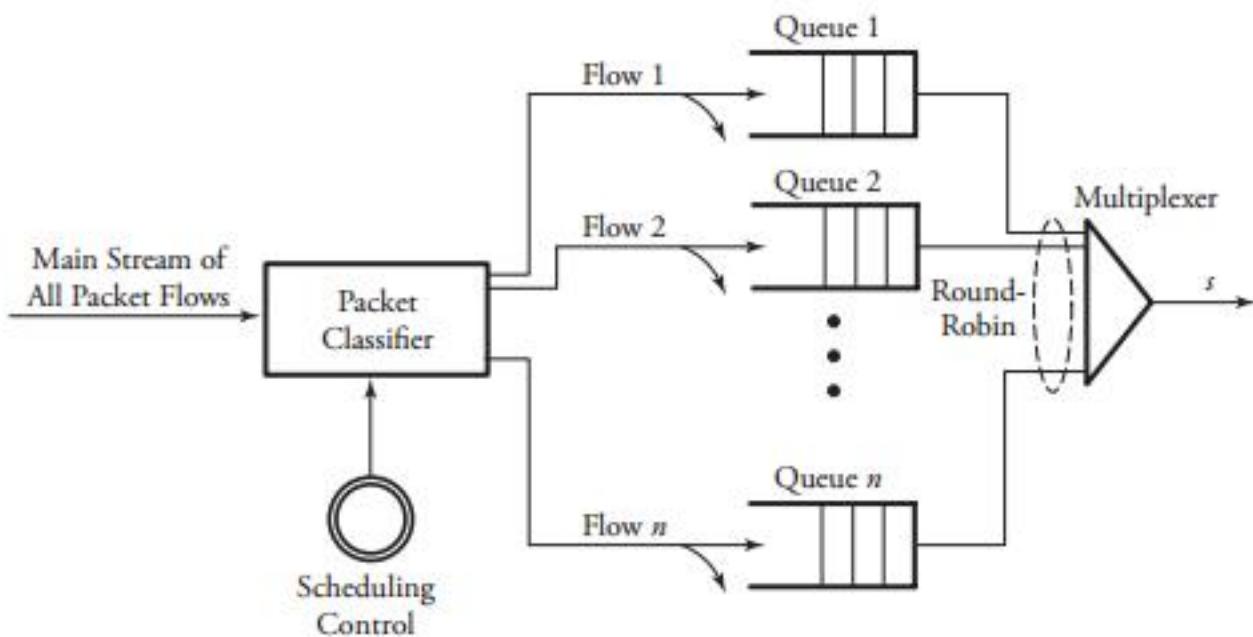


Figure 12.11 Overview of fair-queueing scheduler

If we substitute θ_i of Equation (12.18) into Equation (12.16), we obtain the total system delay for a packet passing queue i and the server:

$$E[T_i] = E[T_{q,i}] + \frac{1}{\mu_i} + \frac{1}{\mu_i \left(1 - \sum_{j=1}^{i-1} \rho_j\right)}, \quad (12.19)$$

where $E[T_{q,i}]$ is the same as the one computed for the nonpreemptive case in Equation (12.14).

Fair Queueing Scheduler

The *fair-queueing* (FQ) scheduler is designed to better and more fairly treat servicing packets. Consider multiple queues with different priorities. The fair queueing shown in Figure 12.11 eliminates the process of packet priority sorting. This improves the performance and speed of the scheduler significantly. As a result, each flow i is assigned a separate queue i . Thus, each flow i is guaranteed a minimum fair share of s/n bits per second on the output, where s is the transmission bandwidth, and n is the number of flows (or number of queues). In practice, since all the inputs (flows) may not necessarily have packets at the same time, the individual queue's bandwidth share will be higher than s/n .

Assume that a_j is the arriving time of packet j of a flow. Let s_j and f_j be the start and ending times, respectively, for transmission of the packet. Then, c_j , as the virtual

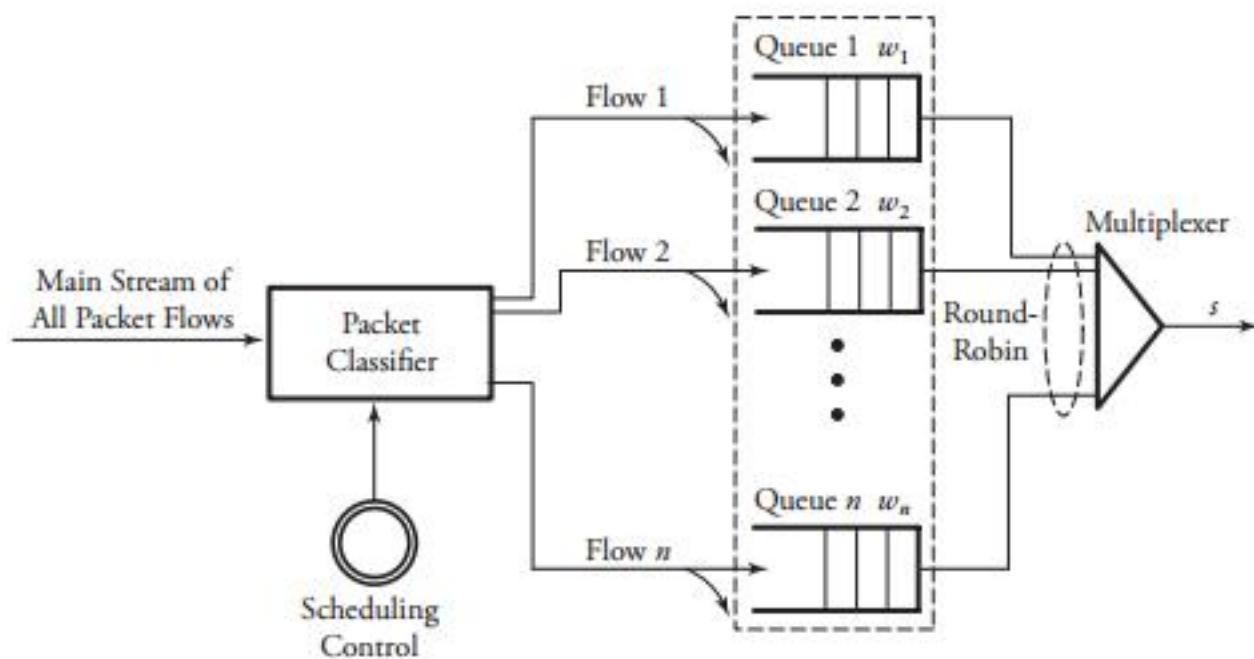


Figure 12.12 Overview of weighted fair-queueing scheduler

clock count needed to transmit the packet, can be obtained from $c_j = f_j - s_j$, since $s_j = \max(f_{j-1}, a_j)$. Then:

$$c_j = f_j - \max(f_{j-1}, a_j). \quad (12.20)$$

We can calculate f_j for every flow and use it as a timestamp. A timestamp is used for maintaining the order of packet transmissions. For example, the timestamp is each time the next packet is to be transmitted. In such a case, any packet with the lowest timestamp finishes the transmission before all others.

Weighted Fair Queueing Scheduler

The *weighted fair-queueing* scheduler (WFQ) scheduler is an improvement over the fair-queueing scheduler. Consider the weighted fair-queueing scheme shown in Figure 12.12. For an n -queue system, queue $i \in \{1 \dots n\}$ is assigned a weight w_i . The outgoing link capacity s is shared among the flows with respect to their allocated weights. Thus, each flow i is guaranteed to have a service rate of at least

$$r_i = \left(\frac{w_i}{\sum_{j=1}^n w_j} \right) s. \quad (12.21)$$

Given a certain available bandwidth, if a queue is empty at a given time, the unused portion of its bandwidth is shared among the other active queues according to their

respective weights. WFQ provides a significantly effective solution for servicing real-time packets and non-real-time packets, owing to its fairness and ability to satisfy real-time constraints. Each weight, w_i , specifies the fraction of the total output port bandwidth dedicated to flow i . Assigning a higher weight to the real-time queues reduces delay for the real-time packets, so real-time guarantees can be ensured. These weights can be tuned to specific traffic requirements, based on empirical results. As mentioned, any unused bandwidth from inactive queues can be distributed among the active queues with respect to their allocated weights, again with the same fair share of

$$r_i = \left(\frac{w_i}{\sum_{j \in b(t)} w_j} \right) s, \quad (12.22)$$

where $b(t)$ is a set of active queues at any time t . The delay bound in this scheme is independent of the maximum number of connections n , which is why WFQ is considered one of the best queueing schemes for providing tight delay bounds. As a trade-off to the effectiveness of this scheme, the implementation of a complete weighted fair-queueing scheduler scheme is quite complex.

A version of this scheduler, the so-called *weighted round-robin* (WRR) scheduler, was proposed for asynchronous transfer mode. In WRR, each flow is served in a round-robin fashion with respect to a weight assigned for each flow i , without considering packet length. This is very similar to the deficit round-robin scheduler discussed next for fixed-size packets. Since the system deals only with same-size packets, it is very efficient and fast. The service received by an active flow during its round of service is proportional to its fair share of the bandwidth specified by the weight for each flow.

Example. Figure 12.13 shows a set of four flows—A, B, C, and D—to be processed at one of the outputs of a router. Each packet's arrival time and label are indicated. Three schedulers are compared: priority queueing, fair queueing, and weighted fair queueing. All schedulers scan the flows, starting from flow A. With priority queueing, priorities decrease from line A to line D. With fair queueing, if the arrival times of two packets are the same, the smaller flow number is selected. Using weighted fair queueing, we assumed that flows A, B, C, and D are given 20 percent, 10 percent, 40 percent, and 20 percent of the output capacity, respectively.

Deficit Round-Robin Scheduler

In weighted fair queueing the dependency on a sorting operation that grows with the number of flows is a concern for scalability as speed increases. In the *deficit round-robin* (DRR) scheduler, each flow i is allocated b_i bits in each round of service, and

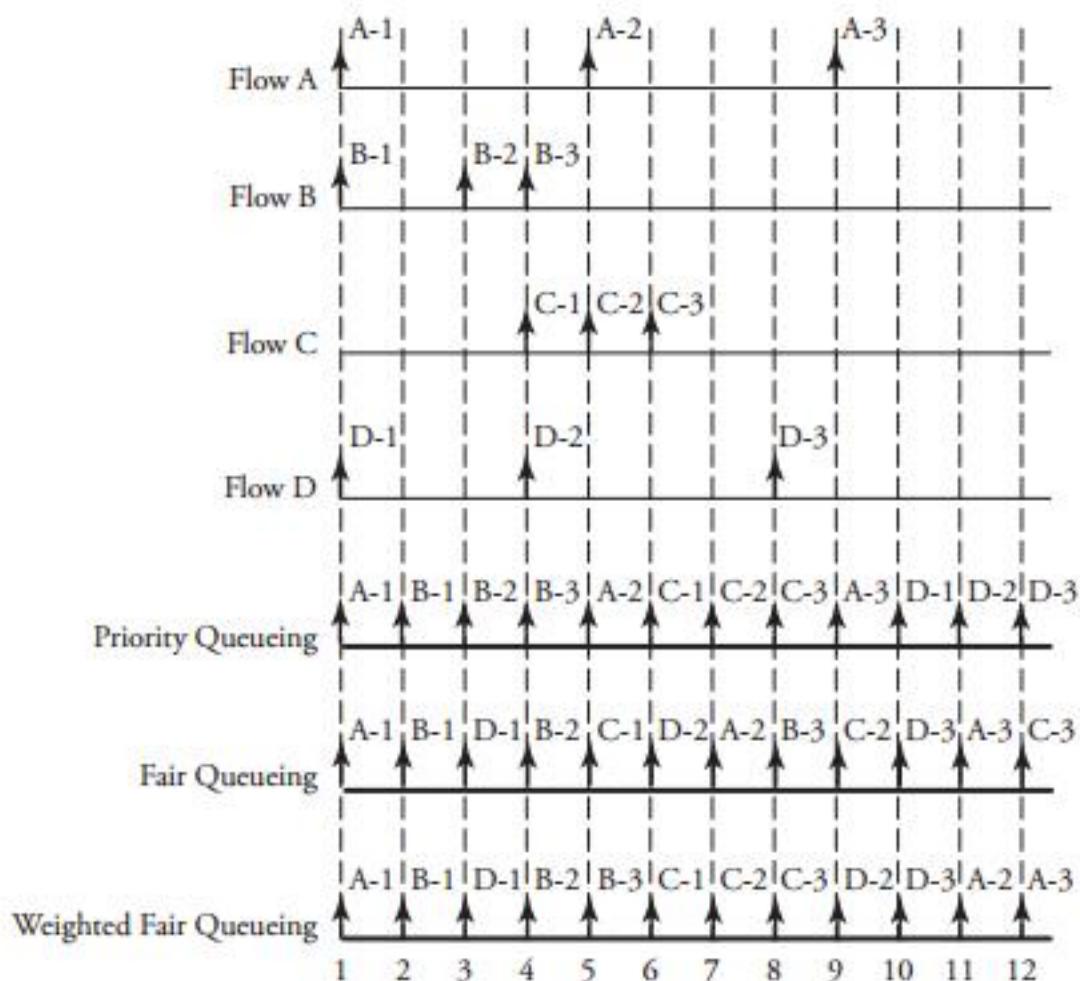


Figure 12.13 Comparison of priority queueing, fair queueing, and weighted fair queueing on processing of four flows

b_m is defined as the minimum bit-allocation value among all flows. Thus, we have $b_m = \min\{b_i\}$. In every cycle time, active flows are placed in an active list and serviced in round-robin order. If a packet cannot be completely serviced in a round of service without exceeding b_i , the packet is kept active until the next service round, and the unused portion of b_i is added to the next round. If a flow becomes inactive, the deficit is not accumulated to the next round of service and is eventually dropped from the node.

Although DRR is fair in terms of throughput, it lacks any reasonable delay bound. Another major disadvantage of this technique is that the delay bound for a flow with a small share of bandwidth can be very large. For example, consider a situation in which all flows are active. A low-weighted flow located at the end of the active list has to wait for all the other flows to be serviced before its turn, even if it is transmitting a minimum-sized packet.

Earliest Deadline First Scheduler

The *earliest deadline first* (EDF) scheduler computes the departure deadline for incoming packets and forms a *sorted deadline list* of packets to ensure the required transmission rate and maximum delay guarantees. The key lies in the assignment of the deadline so that the server provides a delay bound for those packets specified in the list. Thus, the deadline for a packet can be defined as

$$D = t_a + T_s, \quad (12.23)$$

where t_a is the expected arrival time of a packet at the server, and T_s is the delay guarantee of the server associated with the queue that the packet arrives from.

12.3 Differentiated Services QoS

The *differentiated services* (DS), or DiffServ, approach provides a simpler and more scalable QoS. DS minimizes the amount of storage needed in a router by processing traffic flows in an aggregate manner, moving all the complex procedures from the core to the edge of the network. A *traffic conditioner* is one of the main features of a DiffServ node to protect the DiffServ domain. As shown in Figure 12.14, the traffic conditioner includes four major components: meter, marker, shaper, and dropper. A *meter* measures the traffic to make sure that packets do not exceed their traffic profiles. A *marker* marks or unmarks packets in order to keep track of their situations in the DS node. A *shaper* delays any packet that is not compliant with the traffic profile. Finally, a *dropper* discards any packet that violates its traffic profile.

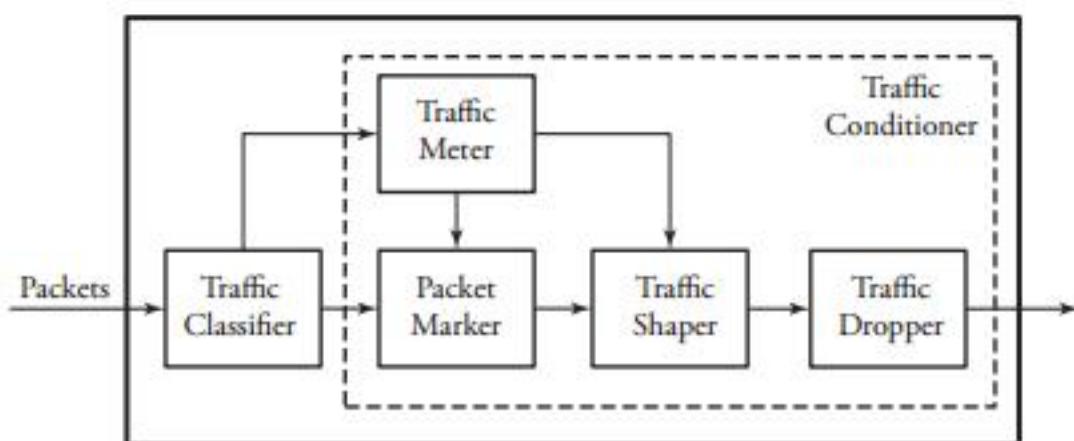


Figure 12.14 Overview of DiffServ operation

When users request a certain type of service, a service provider must satisfy the user's need. In order to allocate and control the available bandwidth within the DS domain, a bandwidth broker is needed to manage the traffic. The *bandwidth broker* operates in its own DS domain and maintains contact with other bandwidth brokers at neighboring domains. This is a way to confirm that a packet's requested service is valid throughout all domains within the routing path of the packet.

In order to process traffic flows in an aggregate manner, a packet must go through a *service-level agreement* (SLA) that includes a traffic-conditioning agreement (TCA). An SLA indicates the type of forwarding service, and a TCA presents all the detailed parameters that a customer receives. An SLA can be either static or dynamic. A static SLA is a long-term agreement, and a dynamic SLA uses the bandwidth broker that allows users to make changes more frequently. A user preferring packet-dependent quality-of-service can simply mark different values in the *type of service* (ToS) field at either the host or its access router. Routers in the DS model then detect the value in the DS field in *per hop behaviors* (PHBs). The quality-of-service can then be performed in accordance with the PHB.

In order to establish a traffic-policing scheme, a service provider uses a *traffic classifier* and a *traffic conditioner* at the domain's edge router when packets enter the service provider's network. The traffic classifier routes packets to specific outputs, based on the values found inside multiple fields of a packet header. The traffic conditioner detects and responds if any packet has violated any of the rules specified in the TCA. The DiffServ field value is set at the network boundaries. A DiffServ router uses the traffic classifier to select packets and then uses buffer management and a scheduling mechanism to deliver the specific PHB. The 8-bit DiffServ field is intended to replace the IPv4 ToS field and the IPv6 traffic class field. Six bits are used as a *differentiated services code point* (DSCP) to specify its PHB. The last 2 bits are unused and are ignored by the DS node.

12.3.1 Per-Hop Behavior (PHB)

We define two PHBs: *expedited forwarding* and *assured forwarding*. As for DiffServ domains, the expedited forwarding PHB provides low-loss, low-latency, low-jitter, ensured-bandwidth, and end-to-end services. Low latency and ensured bandwidth can be provided with a few configurations on the DiffServ node. Both the aggregate arrival rate for expedited-forwarding PHB packets and the aggregate arrival rate should be less than the aggregate minimum departure rate.

Several types of queue-scheduling mechanisms may be used to implement expedited-forwarding PHB. Assured-forwarding PHB delivers packets with high assurance and

high throughput, as long as the aggregate traffic does not exceed TCA. However, users are allowed to violate TCA, but the traffic beyond TCA is not given high assurance. Unlike the expedited forwarding PHB, the ensured-forwarding PHB does not provide low-latency and low-jitter application. The ensured-forwarding PHB group can be classified into three service types: good, average, and poor. Three possible drop-precedence values are then assigned to packets within each class, determining the priority of the corresponding packet.

12.4 Resource Allocation

Scheduling algorithms provide QoS guarantees to traffic flows by controlling the flow of transmission, but sufficient buffer space has to be allocated to hold incoming packets, especially in high-speed networks. Some form of protection mechanism has to be implemented to provide flow isolation for preventing ill-behaving flows from occupying the entire queueing buffer. In addition, the mechanism must make packet-discard decisions on the basis of the network congestion level. Thus, buffer management is required to provide rate guarantees for a network.

Issues that involve *resource allocation* for packet-switched networks are fundamentally different from shared-access networks, such as Ethernet, where end hosts are able to directly observe the traffic on the shared medium. End hosts can thus tailor the rate at which they generate and send traffic, based on the traffic on the medium.

Consider the traffic-congestion case shown in Figure 12.15, where two links with a total traffic volume of $\alpha_1 + \alpha_2$ are connected to router R1 with an immediate outgoing link capacity α , where $\alpha < \alpha_1 + \alpha_2$. When it reaches the router, the sum of traffic encounters a low speed at the outgoing gate of the router. Router R1 starts to build up unserved packets and eventually enters a state of congestion. Situations like this, in

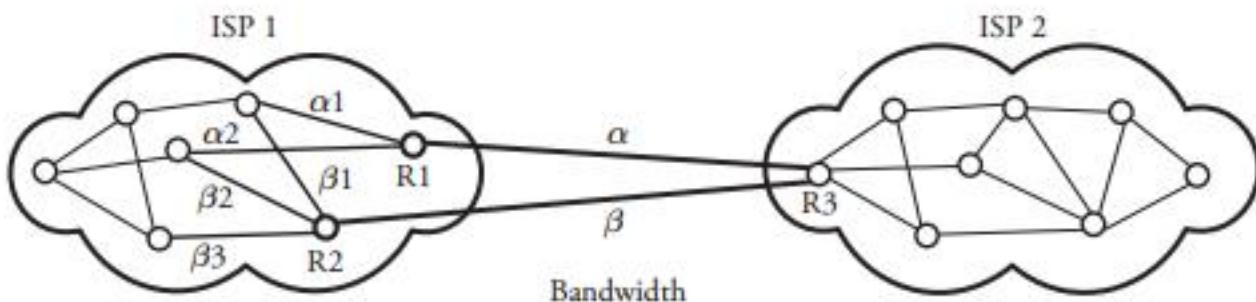


Figure 12.15 The need for network resource allocation when outgoing link capacity (α) is less than the sum of incoming ($\alpha_1 + \alpha_2$) traffic at a router

which traffic from several sources has to be combined onto a common low-speed link, are frequent on the Internet but are unusual in shared-access networks.

If the same situation occurs on router R2, where $\beta < \beta_1 + \beta_2 + \beta_3$, the frequency of severe congestion edge routers of the ISP 1 network is possible. This situation is by all means undesired from the standpoint of network management. Another important situation shown in Figure 12.15 is that router R3, located in a different domain, must be able to handle a great volume of traffic totaled at $\alpha + \beta$. This situation must convince a network manager that the type of router must depend on the location of the router in the network. The type of a router in a specific location of the network must be determined after thorough study, simulation, and taking all the necessary network design factors into consideration.

12.4.1 Management of Resources

Connection-oriented networks have a method of congestion control that leads to an underutilization of network resources. In the virtual-circuit approach, end hosts send a connection set-up message, which traverses the network and reserves buffers at each router for the connection. If adequate resources are not available at each router along the path of the connection, the connection is denied. The resources reserved for one connection cannot be used by another connection, leading to an underutilization of buffers at routers.

Service models of resource allocation are of two basic types:

1. *Best effort*. A network provides no guarantee to end hosts on the type of service to be delivered.
2. *QoS*. An end host may request a QoS connection. The host chooses from a set of QoS classes that the network supports.

In connectionless networks, datagrams are switched independently, and datagrams flowing through a router are called connectionless flows. The router maintains state of each flow to make necessary informed decisions. For example, in Figure 12.15, either router R2 or source host H2 can initiate a flow record. In some cases, the router observes the source/destination address combination for a packet and classifies packets into different flows. In some other cases, before a flow begins, the source sends some information to identify a connectionless flow originating from the host.

12.4.2 Classification of Resource-Allocation Schemes

Resource-allocation schemes can be classified as router based versus host based, fixed versus adaptive, and window based versus rate based.

Router Based versus Host Based

Resource allocation can be classified according to whether a router or a host sets up required resources. In a *router-based scheme*, routers have primary responsibility for congestion control. A router selectively forwards packets or drops them, if required, to manage the allocation of existing resources.

A router also sends information to end hosts on the amount of traffic it can generate and send. In a *host-based scheme*, end hosts have primary responsibility for congestion control. Hosts observe the traffic conditions, such as throughput, delay, and packet losses, and adjust the rate at which they generate and send packets accordingly. In most networks, resource-allocation schemes may place a certain level of responsibility on both routers and end hosts.

Fixed versus Adaptive

In *fixed reservation schemes*, end hosts request resources at the router level before a flow begins. The router then allocates enough resources, such as bandwidth and buffer space, for the flow, based on its available resources. A router also ensures that the new reservation does not affect the quality of service provided to the existing reservations. If resources are unavailable, the router rejects the request from the end host. In an *adaptive reservation scheme*, end hosts send packets without reserving resources at the router and then adjust their sending rates, based on observable traffic conditions or the response from the router.

The observable traffic conditions are the amount of packet loss, delay, or other metrics. A router may also send messages to end hosts to slow down their sending rate. *Fixed reservation schemes* are router based, as the router is responsible for allocating sufficient resources for the flow. Thus, routers have primary responsibility for allocating and managing resources. Adaptive reservation schemes can be either router based or host based. If end hosts adjust rates of transmitted packets based on observable traffic conditions, the scheme is typically host based.

Window Based versus Rate Based

In *window-based resource allocation*, a receiver chooses a window size. This window size is dependent on the buffer space available to the receiver. The receiver then sends this window size to the sender. The sender transmits packets in accordance with the window size advertised by the receiver. In *rate-based resource allocation*, a receiver specifies a maximum rate of bits per second (b/s) it can handle. A sender sends traffic in compliance with the rate advertised by the receiver. The reservation-based allocation scheme might also involve reservations in b/s. In this case, routers along the path of a flow can handle traffic up to the advertised rate.

12.4.3 Fairness in Resource Allocation

The effectiveness of a resource-allocation scheme can be evaluated by considering two primary metrics: throughput and delay. Throughput has to be as large as possible; delay for a flow should normally be minimal. When the number of packets admitted into the network increases, the throughput tends to improve. But when the number of packets increases, the capacity of links is saturated, and thus the delay also increases, because a larger number of packets are buffered at the intermediate routers, resulting in increased delay.

A more effective method of evaluating the effectiveness of a resource-allocation scheme is to consider the ratio of throughput to delay, or *power*. As the number of packets admitted into a network builds up, the ratio of the throughput to delay increases. This is true until the network load threshold for low delay is reached. Beyond this limit, the network is overloaded, and the power drops rapidly.

A resource-allocation scheme must also be fair. This implies that each traffic flow through the network receives an equal share of the bandwidth. However, disregarding the flow throughputs (flow rates) itself is not fair. Raj Jain proposes a *fairness index* for n flows f_1, f_2, \dots, f_n as

$$\sigma = \frac{\left(\sum_{i=1}^n f_i\right)^2}{n \sum_{i=1}^n f_i^2}. \quad (12.24)$$

The fairness index σ is always between 0 and 1 to represent the lowest and the best fairness. With reservation-based resource-allocation schemes, it is always possible for certain traffic, such as voice, to achieve a greater share of the available bandwidth by reservation. This may lead to unfairness in resource allocation. A fair resource-allocation scheme may not always be the most effective method of resource allocation.

12.4.4 ATM Resource Allocation

Traffic management and congestion control differ in ATM networks and regular packet-switched networks. The high-speed multiplexing and the small packet sizes in ATM networks make the task of congestion control difficult. One of the restrictions that ATM networks face in traffic control is an insufficient number of bits available in a cell for network control. Maintaining constant bit rate would help avoid the congestion in ATM networks.

Achieving Constant Bit Rate (CBR)

ATM networks are designed for fast cell switching and efficient routing at switches. One of the important requirements in ATM networks is low cell delay along with

a constant rate of cell delivery. This is important because delay-sensitive applications, such as a voice communication, cannot tolerate long delays among their cells. *Cell-delay variations* can occur both at sources and within a network. In any case, a destination user must be able to adapt to cell-delay variations.

In order to maintain a constant bit rate (CBR), a certain algorithm is used in ATM networks. Let J_i be the cell i delay variation that a given application can tolerate. Let c be the constant rate at which cells are delivered to users. Thus, the interval between two similar points of two consecutive arriving cells is $\frac{1}{c}$. Let t_i be the time at which the i th cell is received. Then, the variable delay for the i th cell is given by

$$J_i = J_{i-1} - \left(t_i - \left(t_{i-1} + \frac{1}{c} \right) \right). \quad (12.25)$$

The analysis bases J_1 as the first value and starts recursively, using i , where $i \geq 2$. For example, for the second cell, we have:

$$J_2 = J_1 - \left(t_2 - \left(t_1 + \frac{1}{c} \right) \right). \quad (12.26)$$

The cell-delay variation can be reduced by increasing both the available network resources and the data rate at the user network interface. The cell delay at the network level can be lower, because cells are small and fixed size and have a fixed header format. For ATM networks, as well as other types of networks, resource allocation has to be controlled to avoid congestion. Variable cell delay can occur because of the processing overhead in the ATM layer and the physical layer. The encapsulation process at the ATM layer and the overhead bits added at the physical layer lead to variable cell delays.

The key factor that makes it extremely difficult to control wide-area ATM networks is the fact that the cell-transmission time in ATM networks is quite small; hence, feedback control may not be able to react quickly, making the use of adaptive reservation systems for congestion control ineffective. The key factors that influence traffic management in ATM networks are latency and cell-delay variation. The network propagation delay can sometimes be comparable to cell-transmission times. Hence, a source cannot use implicit feedback control, as the feedback control signals arrive too late for any appropriate cell-retransmission strategy.

Resource Management Using Virtual Paths

The main resource management in ATM networks involves the proper use of virtual paths. Several *virtual channel connections* (VCC) are grouped into a *virtual path connection* (VPC). ATM networks provide an aggregate capacity for all virtual channels

within a virtual path. The primary parameters for traffic management in ATM networks are *cell-loss ratio*, *cell-transfer delay*, and *cell-delay variation*. If a virtual channel passes through multiple virtual paths, the performance on that virtual channel depends on the performance of all the virtual paths that incorporate the virtual channel.

Different VCCs within a VPC may have different QoS requirements. The aggregate capacity and performance characteristics for a virtual path should be set to meet the requirements of the most demanding VCCs. There are two different approaches to resource allocation for virtual paths. In the first approach, an ATM network can set the capacity of the virtual path to meet the peak data rates of all VCCs within that VPC. However, this may lead to an underutilization of network resources. The second approach follows the idea of *statistical multiplexing*. An ATM network can set the capacity of a VPC to be greater than or equal to the average capacity of all its VCCs. With this approach, the total capacity of the virtual path becomes smaller than the aggregate peak data rate, leading to more efficient utilization of network resources. The second approach works best when a number of VCCs are grouped into a VPC on the basis of similar quality of service.

Connection Admission Control

When requesting a new VCC or VPC in ATM networks, a user must associate the connection with a particular QoS. Generally, a user can choose from a range of QoS classes that the network supports. An ATM network accepts the connection only if it can meet the QoS requirements without compromising on the QoS of existing connections. Once the network accepts a connection, the user and the network enter into a traffic contract. The network continues to provide the QoS as long as the user does not violate the traffic contract. The traffic contract is characterized by four parameters: *peak cell rate* (PCR), *cell-delay variation* (CDV), *sustainable cell rate* (SCR), and *burst tolerance*.

The peak cell rate of a connection is the maximum rate at which a source generates cells within a connection. The sustainable cell rate represents the upper bound on the average cell rate associated with a given connection. Burst tolerance represents the maximum variability of the cell-arrival rate from a SCR. For a constant bit rate (CBR) source, only the first two parameters are relevant. The cell-delay variation may cause an increase in the cell rate from its peak value, because cell delays may cause cells to clump up, thereby accounting for an increase in the cell rate from its peak value.

For variable bit rate (VBR) sources, all four parameters are relevant. A future flow pattern can be predicted by knowing the SCR and burst tolerance. A network characterizes each connection based on these four parameters. The admission-control decision is based primarily on users' QoS requirements. In an ATM connection, a user

is also allowed to assign a cell-loss priority (CLP). The CLP bit in the header indicates either normal priority ($CLP = 0$ or 1) or high priority ($CLP = 0$).

Usage Parameter Control

Once a network's admission-control function accepts a connection, the *usage parameter control* scheme is used to determine whether the connection confirms the traffic contract. The usage parameter control mechanism is normally accompanied by a QoS traffic-shaping scheme. The usage parameter control involves the control of the peak cell rate and the associated cell-delay variation. The traffic flows are monitored to check whether the flow violates the contract by exceeding the peak cell rate, subject to the maximum specified cell-delay variation.

The CLP header bit defined in Section 2.4.2 is used for tagging or discarding noncompliant cells. Those cells that were tagged and cells with $CLP = 1$ are classified as low-priority cells. If the network becomes congested, these cells are dropped first, in order to provide better performance to high-priority cells. Usage parameter control can also provide a mechanism of traffic policing, whereby cells that violate the traffic contract are either tagged or discarded.

12.4.5 Cell Scheduling and QoS

ATM networks provide a *guaranteed frame rate* (GFR) service that requires the network to handle frames in addition to ATM cells. With GFR, all cells of a frame have the same CLP bit setting. Frames with $CLP = 1$ setting are low-priority frames. This service provides a minimum guarantee to a connection by setting up GFR virtual channels. The minimum capacity is ensured only for frames with $CLP = 0$. The implementation involves *tagging* and *policing*.

The tagging process is used to identify frames that violate the GFR traffic contract. The ATM network sets the CLP bit to 1 on all cells in a noncompliant frame. Tagged cells are given a lower priority in the buffer-management and scheduling stages. The buffer management is done when congestion occurs. In such a case, all tagged cells are discarded to give preference to untagged cells. The buffering scheme is set up such that an untagged cell can replace a tagged cell in the buffer when resources are limited.

As a QoS component, a scheduler gives preference to untagged cells over tagged cells. Scheduling among router queues ensures the minimum rate requirements for each virtual channel. The traffic on each connection is monitored all the time. For a frame with confirmed contract terms, all cells in the frame must be confirmed. Frames that violate the traffic contract are either tagged to a lower priority or discarded. Therefore,

this process filters frames that may cause congestion by overloading the network. Frames are then checked to determine whether they qualify for guaranteed delivery.

12.5 Summary

Methods of providing QoS are divided into two broad categories: *integrated services* and *differentiated services*. Integrated services provide QoS to individual applications and flow records. QoS protocols in this category include traffic shaping and packet scheduling. Traffic shaping regulates the spacing between incoming packets. Two traffic-shaping algorithms are *leaky bucket* and *token bucket*. In a leaky-bucket traffic shaper, traffic is regulated at a constant rate, much like the flow of water from a leaky bucket. In the token-bucket algorithm, enough tokens are assigned to each incoming packet. If the bucket becomes empty, packets stay in the buffer until the required number of tokens is generated.

Packet scheduling involves managing packets in queues. Several scheduling techniques *FIFO*, *priority queueing*, *fair queueing*, and *weighted fair queueing* (WFQ). WFQ is an improvement over fair queueing, in which each flow i is assigned a weight w_i . Another version of WFQ is *deficit round-robin*, in which each flow i is allocated b_i bits in each round of service.

The differentiated services (DiffServ) approach is based on providing QoS support to a broad class of applications. The *traffic conditioner*, one of the main features of a DiffServ node, protects the DiffServ domain. The traffic conditioner includes four major components: *meter*, *marker*, *shaper*, and *dropper*.

Resource allocation in networks can be classified by various approaches: fixed versus adaptive, router based versus host based, window based versus rate based. Finally a few methods of ATM resource control were presented.

The next chapter discusses the internal architecture of switch fabrics. This topic is the basis of discussion in several later chapters as well.

12.6 Exercises

1. In a leaky-bucket traffic-shaping scheme, $w = 4$ grants are initially allocated to a session. The count is restored back to 4 every $4/g$ seconds. Assume a Poisson arrival of packets to the system.
 - (a) Develop the first five balance equations for the corresponding Markovian process.
 - (b) Sketch a Markov chain representing the number of grants allocated to packets, and indicate as many transition values as you have computed.

2. A small router is equipped with the leaky-bucket traffic-shaping scheme; the capacity of the grant buffer is set on window size $w = 2$. Packet arrival during $1/g$ is uniformly distributed, with four possibilities of $k = 0, 1, 2$, and 3 packets. Assume $P_0 = 0.008$.
- Find the probability that k packets arrive during $1/g$, denoted by $P_X(k)$.
 - Find the probabilities that $0, 1, 2$, or 3 grants are allocated to arriving packets.
 - Find all the transition probabilities for three states $0, 1$, and 2 .
 - Sketch a Markov chain representing the number of grants allocated to packets for the first three states.
3. A router attached to mail server is responsible only for receiving and smoothing e-mail. The router is equipped with the leaky-bucket traffic-shaping scheme, whereby the capacity of the grant buffer is set on window size $w = 4$. Packets arrive at $\lambda = 20$ packets per second, and grants arrive at $g = 30$ packets per second. Packet arrival during $1/g$ is distributed according to Poisson. Assume that $P_0 = 0.007$.
- Find the probability that k packets arrive during $1/g$, denoted by $P_X(k)$.
 - Find the probabilities that $0, 1, 2$, and 3 grants are allocated to arriving packets.
 - Find all the transition probabilities for four states $0, 1, 2$, and 3 .
 - Sketch a Markov chain representing the number of grants allocated to packets for the first four states.
4. Consider a token-bucket traffic shaper. Let the bucket size be b bits and the token arrival rate be v b/s; let the maximum output data rate be z b/s.
- Derive an equation for T_b , the time consumed for a flow to be transmitted at the maximum rate.
 - Find T_b when the bucket size is $b = 0.5$ Mb, $v = 10$ Mb/s, and $z = 100$ Mb/s.
5. Does nonpreemptive priority queueing change the packet transmission orders if bit-by-bit round-robin service is used?
6. Derive an expression for a priority scheduler to present the mean residual service time, r_i , used in Equation (12.10). (*Hint:* Use μ_i .)
7. Assume that all components of a three-flow ($n = 3$) priority scheduler are identical: $\lambda_i = \lambda = 0.2/\text{ms}$, $1/\mu_i = 1/\mu = 1/\text{ms}$, and $r_i = r = 0.5/\text{ms}$. Find the total system delay for a packet passing each of the three queues $1, 2$, and 3 and the server denoted by $E[T_1]$, $E[T_2]$, and $E[T_3]$, respectively. Do the following.
- Use a nonpreemptive priority scheduler for the task.
 - Use a preemptive priority scheduler for the task.
 - Comment on results obtained in parts (a) and (b).

8. We want to compare the impact of an increased number of inputs on total delay in priority schedulers. Assume that all components of a three-flow ($n = 3$) and a four-flow ($n = 4$) priority scheduler are identical: $\lambda_i = \lambda = 0.2/\text{ms}$, $1/\mu_i = 1/\mu = 1/\text{ms}$, and $r_i = r = 0.5/\text{ms}$. Find the total system delay for a packet passing queue 3 denoted by $E[T_3]$. Do the following
 - (a) Use a nonpreemptive priority scheduler for the task.
 - (b) Use a preemptive priority scheduler for the task.
 - (c) Justify the results obtained in parts (a) and (b).
9. Suppose that four flows are processed in a router that accepts only equal-size packets. Packets in these flows arrive at the following virtual clock times:
Flow 1: 4, 5, 6, 7, 9
Flow 2: 1, 6, 9, 12, 14
Flow 3: 1, 4, 8, 10, 12
Flow 4: 2, 4, 5, 6, 12
 - (a) For each packet, give the virtual clock count at which it is transmitted, using fair queueing. If the arrival times of two packets are the same, the smaller flow number is selected.
 - (b) Now consider weighted fair queueing, whereby flows 1, 2, 3, and 4 are given 10 percent, 20 percent, 30 percent, and 40 percent of the output capacity, respectively. For each packet, give the virtual clock count at which it is transmitted.
10. Consider the router interfaces in Figures 3.11 and 3.16. Explain where and how fair queueing can be implemented.
11. We define the timing parameters s_i , f_i , and a_i for a weighted fair-queueing scheduler similar to what we did for the fair-queueing scheduler. Derive a relationship among these parameters, taking into consideration the weight of each flow (packet), w_i .
12. In a nonpreemptive priority queueing scheme, a low-priority flow is guaranteed to receive 10 percent of the total bit rate s of the transmission link.
 - (a) How much better does this low-priority flow perform?
 - (b) What would be the impact on the performance of the high-priority flows?
13. Each output port processor unit of a router has four inputs designated to four different flows. The unit receives the packets in the following order during a period

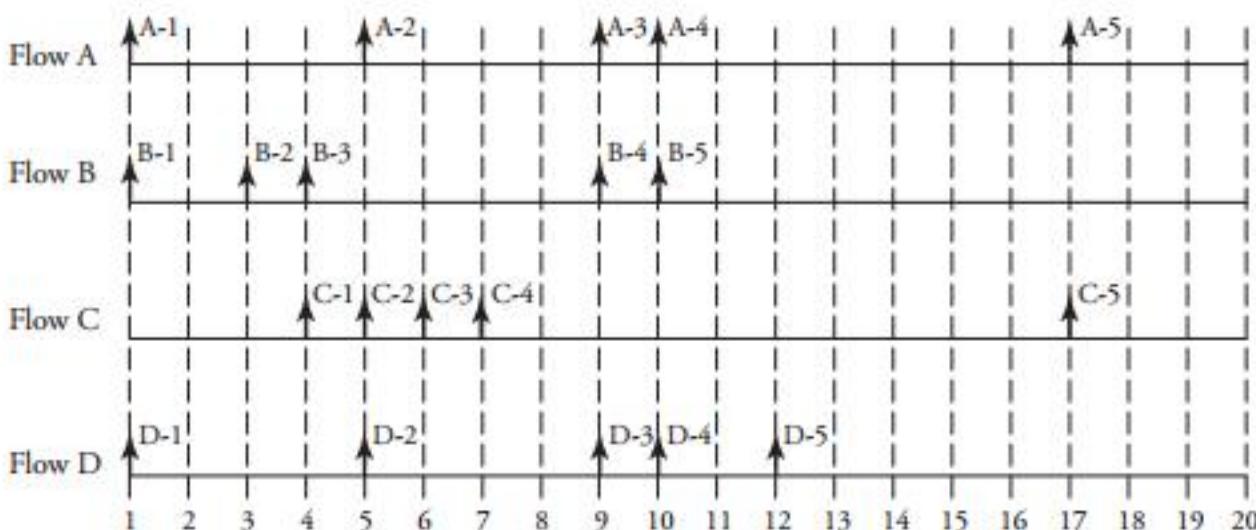


Figure 12.16 Exercise 14 comparison of priority queueing, fair queueing, and weighted fair queueing on processing of four flows

in which the output port is busy but all queues are empty. Give the order in which the packets are transmitted.

Flow: 1, 1, 1, 1, 1, 1, 2, 2, 3, 3, 3, 4

Packet: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

Packet size: 110, 110, 110, 100, 100, 100, 100, 200, 200, 240, 240, 240

(a) Assume a fair queueing scheduler.

(b) Assume a weighted fair-queueing scheduler with flow $i \in \{1, 2, 3, 4\}$ having weights $w_i \in \{10\%, 20\%, 30\%, 40\%\}$ of the output capacity, respectively.

14. In Figure 12.16, flows A, B, C, and D are processed at one of the outputs of a router. For each packet, the time it arrives and its label are indicated in the figure. Specify the order of packets transmitted at this output, using the following three schedulers. All schedulers scan the flows, starting from flow A.
 - (a) *Priority queueing*. Priorities decrease from line A to line D.
 - (b) *Fair queueing*. If the arrival times of two packets are the same, the smaller flow number is selected.
 - (c) *Weighted fair queueing*, where flows A, B, C, and D are given 10 percent, 20 percent, 30 percent, and 40 percent, respectively, of the output capacity.
15. Figure 12.17 shows four flows (A, B, C, and D) to be processed at one of the outputs of a router. For each packet, the time it arrives and its label are indicated

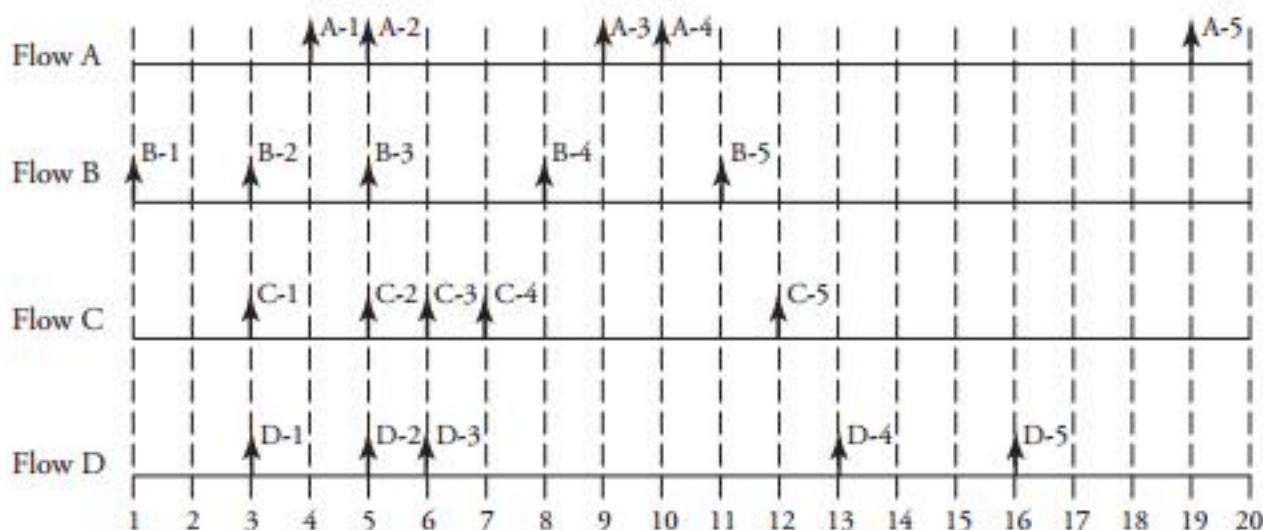


Figure 12.17 Exercise 15 comparison of priority queueing, fair queueing, and weighted fair queueing on processing of four flows

in the figure. Specify the order of packets transmitted at this output, using the following three schedulers. All schedulers scan the flows, starting from flow A.

- Priority queueing.* Priorities decrease from line A to line D.
- Fair queueing.* If the arrival times of two packets are the same, the smaller flow number is selected.
- Weighted fair queueing,* where flows A, B, C, and D are given 20 percent, 10 percent, 40 percent, and 20 percent, respectively, of the output capacity.

CHAPTER 13

Networks in Switch Fabrics

A *switch fabric* is the core switching engine in switching devices as routers. Chapter 3 presented an overview of switching devices. This chapter analyzes this core segment of such devices and introduces several topologies for switching networks.

This chapter explores the following topics:

- *Characteristics and features of switch fabrics*
- *Crossbar switch fabrics*
- *Blocking switch fabrics*
- *Nonblocking switch fabrics*
- *Concentration and expansion switches*
- *Shared-memory switch fabrics*
- *Techniques for improving performance*
- *Case study multipath buffered crossbar*

We begin by classifying characteristics of switching networks and presenting features and definitions of switch fabrics. *Crossbar switches* are the building blocks of switching fabrics, so all aspects of this switch are covered. The case study at the end of this chapter combines several buffered crosspoints to form a buffered crossbar. Blocking switches are constructed with crossbar switches, including *Omega networks*, *Banyan networks*, *Delta networks* and *Beneš networks*. Nonblocking switches constructed with

crossbars include *Clos networks*. *Concentration-based* and *expansion-based* switching networks—two special-purpose switch fabrics—are investigated as well.

A quite different approach for switching in *time domain* uses shared memory, without using any switch elements. Switch fabric techniques that offer better performance include the use of buffering, combined networks, and parallel-plane switching networks.

13.1 Characteristics and Features of Switch Fabrics

The switching function takes place in the switching fabric. The switching structure depends on the need of network operation, available technology, and the required capacity. A switch fabric is an interconnected network of smaller switching units. Several factors can be used to characterize switching systems: buffering, complexity, capability of multipoint connections, speed, performance, cost, reliability, fault tolerance, and scalability. Here, we focus on the topology of the interconnection network as the heart of modern switching systems. The key factors for classifying switching networks are

- Single path versus multipath
- Fixed interstages versus variable interstages
- Deflection routing versus packet discard
- Blocking versus nonblocking

Switching networks are either *single-path* or *multipath* networks. A single-path network has exactly one route between each input port and output port. However, this property can be a source of traffic congestion and traffic blocking. In a multipath network, any connection can be established through more than one path. Each of these two groups of networks can be further classified as either *single-stage* or *multistage* networks. In a single-stage network, a connection is established through one stage of switching; in a multistage network, a packet must pass through several switching stages. In a multistage network, the number of stages often grows with increasing network size.

13.1.1 Blocking and Nonblocking Networks

A switching network is said to be *blocking* if an input port cannot be connected to an unused output port. Otherwise, the switching network is *nonblocking*. A nonblocking switching network can be either of two types. A network is *wide-sense nonblocking* if any input port can be connected to any unused output port without requiring a path to be rerouted. A network is *rearrangeably nonblocking* if, for connecting an input

port to an unused output port, rearranging other paths may be required. A wide-sense nonblocking network is *strictly nonblocking* if there is an idle path from any idle input to idle output for all states of the network. Rearrangeably nonblocking architectures have a more complex control algorithm to make connections, since fewer switches are used. The complexity of the routing algorithm is a trade-off with cost.

13.1.2 Features of Switch Fabrics

Switching networks have several features and options.

- Switching networks may be either *buffered* or *unbuffered*. Buffers may be used to reduce traffic congestion.
- To regulate the flow of packets and prevent buffer overflow, *flow control* can be provided between stages in a multistage network.
- *Discard versus deflection*. At each stage of a switching network, a conflict may arise if several packets request the same output. In networks without flow control, arriving packets that cannot be buffered can be either *discarded* or *deflected*. Or, these two methods can be combined in a switch.
- Modern switching networks are expected to have the capability of *multicasting*—copying to any subset of the outputs—along with *broadcasting*.

13.1.3 Complexity of Switching Networks

A useful measure for approximating the cost of a network is to consider the number of crosspoints and links. In practice, the number of crosspoints has greater impact on the cost of the switching network. Thus, *complexity* refers to the total number of crosspoints used in a switching network. It is important to note that the integrated circuit pin constraints also influence implementation of large-scale networks, especially those with parallel data paths. Practically, a switching network or portions of a network can be placed entirely on a single integrated circuit package. In such cases, the area occupied by the circuits becomes an important complexity measure.

13.1.4 Definitions and Symbols

The topic of switch networking entails a number of definitions and symbols. Consider a network, N , with n inputs and m outputs. This network is referred to as an (n, m) network. An (n, n) network is referred to as an n network. A connection for a given network is identified by (x, y) , where x is an input port and y is an output port. Inputs and outputs to a network are numbered in decimal or binary, starting from 0. Switch

elements are labeled by (i, j) , where i is the stage number of the switch, and j is the location of the switch in that stage. We use $(i, j)_h$ to identify input or output port h on switch (i, j) .

A network is *uniform* if all switch elements have the same dimensions and is *square* if it has the same number of inputs and outputs. The *mirror* of a network N is denoted N^m and is obtained by exchanging inputs and outputs and reversing the directions of all links. Consider network N_1 with n_1 outputs and network N_2 with n_2 inputs. The *concatenation* of two networks N_1 and N_2 is represented by $N_1 + N_2$ and is realized by using output i of N_1 through input i of N_2 . If i is a positive integer and N is an (n, m) network, $i.N$ realizes the network obtained by taking i copies of N , without interconnecting them. The product term $N_1 \times N_2$ is the *series* connection of N_1 and N_2 and is defined by

$$N_1 \times N_2 = (n_2.N_1) + \phi_{n_1, n_2} + (n_1.N_2), \quad (13.1)$$

where ϕ_{n_1, n_2} is the *permutation* among n_1 and n_2 points. Similarly, for network N_1 with n_1 outputs, network N_2 with n_2 inputs and n_3 outputs, and network N_3 with n_1 inputs, the product term $N_1 N_2 N_3$ is defined as

$$N_1 N_2 N_3 = (n_3.N_1) + \phi_{n_1, n_2} + (n_1.N_2) + \phi_{n_3, n_1} + (n_3.N_3) \quad (13.2)$$

and is called the *parallel* connection of N_1 and N_2 .

13.2 Crossbar Switch Fabrics

Crossbar switches are the building blocks of switching fabrics. A crossbar switch with n inputs and n outputs is shown by $X_{n,n}$. Figure 13.1 shows a $X_{4,4}$, or a 4×4 crossbar switching fabric. In a crossbar, every input can be uniquely connected to every output through a *crosspoint*. A crosspoint is the smallest unit of the switching function and can be built using a variety of electronic or digital elements, such as photodiodes, transistors, and AND gates.

Crossbar switches are considered strictly (wide-sense) nonblocking, as a dedicated crosspoint exists for every connection. This is clearly an attractive feature of a crossbar. The blocking, however, may occur when multiple packets are sent to the same output simultaneously. If a particular output port of the crossbar is idle, the desired connection can always be established by selecting the particular crosspoint dedicated to the particular input/output pair.

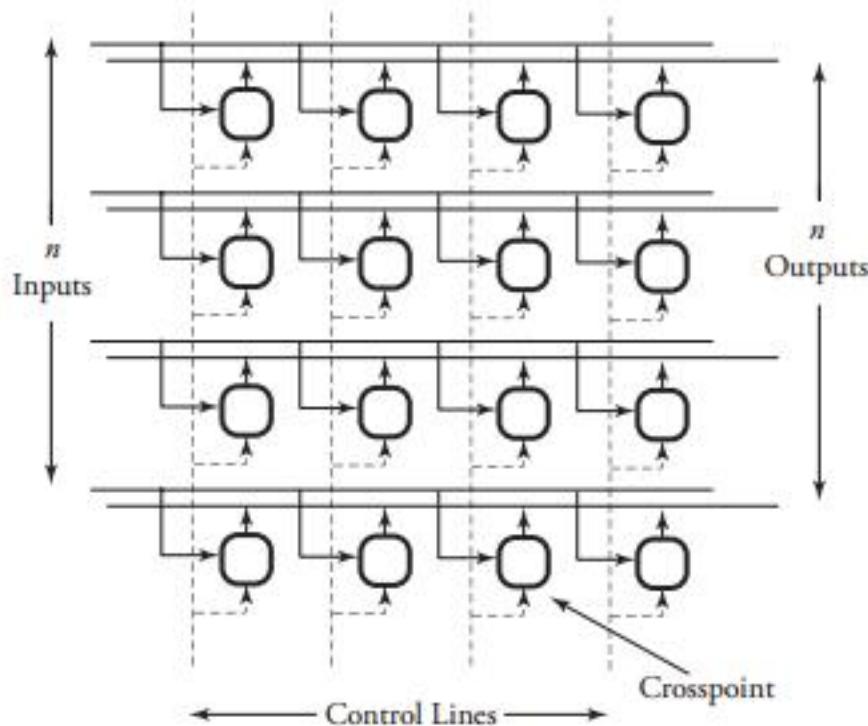


Figure 13.1 A crossbar switching fabric with $n = 4$ inputs/outputs

Crossbar switches are conceptually simple. Their only potential problem is output port contention. The complexity of a switch lies in the complexity of its crosspoints. For an $n \times n$ crossbar, the complexity rises in a quadratic fashion and is equal to n^2 crosspoints. By some clever design work, the complexity can be reduced at the price of bringing certain blocking into the switching.

13.3 Blocking Switch Fabrics

Although, a crossbar can be deployed solely as a switch fabric, it may also be used to construct other multistage switch fabrics, with the trade-off of cost versus higher blocking. This section looks at types of blocking switches: *Omega networks*, *Banyan networks*, *Delta networks*, and *Beneš networks*.

13.3.1 Omega Network

The *Omega network*, $\Omega_{n,d}$, is shown in Figure 13.2 (a). Using $\Omega_{d,d} = X_{d,d}$, we define the Omega network by

$$\Omega_{n,d} = \phi_{n/d,d} + (n/d \cdot X_{d,d}) + \phi_{d,n/d} + (d \cdot \Omega_{n/d,d}). \quad (13.3)$$

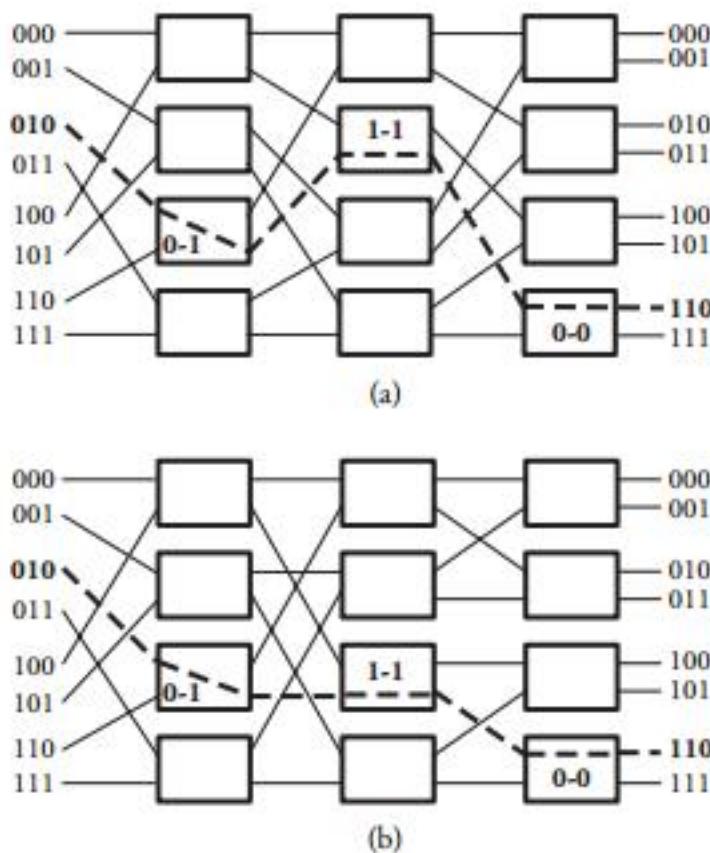


Figure 13.2 (a) An Omega network with an example of its routing; (b) a Banyan network with an example of its routing

In an *Omega network*, only one route exists between each input i_x and output o_x . This fact explains the potential for blocking. The path uniqueness in this network can be shown by the induction on the number of stages, s . The number of stages is exactly $s = \log_d n$.

The Omega network has some useful unique properties. One is that the interconnection segment between each two consecutive stages is identical to that of all other segments. Each of these identical interconnection segments is configured with a *shuffling* property. The shuffling property is easy to understand: An interconnection link starting at the output number of a given crossbar stage ends at the input of the next crossbar stage with the same number but one step rotated to left. For example, consider the second interconnection segment between stages 1 and 2. In this segment, suppose that we are interested in finding the address of the crossbar input this link is connecting to. According to the rotation property, this address is obtained by one left rotation of 010, which results in 100.

The Omega network is also self-routing. The routing in this network is very simple and is achieved by comparing bit by bit between a source address and a destination

address. At each stage, we look at the corresponding bits, starting at the most-significant bit (MSB) of the source and the destination addresses. If bits are the same, a message passes through; otherwise, it is crossed over. For example, consider a case with the source address 010 and the destination address 110, as shown in Figure 13.2. In the first stage, the message is crossed over, as the MSBs of the two addresses are different (0-1). In the second stage, the next corresponding bits are the same (1-1), so the message is sent straight along the crossbar. The case for the third stage is also the same as for the second stage but with corresponding bits of 0-0.

Using the method of blocking probability estimation presented in Section 7.6.4, we can easily obtain an estimation of blocking for the Omega network. Finding all the paths that connect any given input to a given output and assigning p to each and every link of paths as the blocking probability of that link, estimate the blocking probability for an Omega network, $\Omega_{n,d}$, to be $1 - (1 - p)^{s-1}$, where $s = \log_d n$.

13.3.2 Banyan Network

The *Banyan network*, $Y_{n,d}$, as shown in Figure 13.2 (b), is similar to the Delta network. Using $Y_{d,d} = X_{d,d}$, we define the Banyan network by

$$Y_{n,d} = \phi_{n/d,d} + (n/d \cdot X_{d,d}) + \phi_{d,n/d} + (d \cdot Y_{n/d,d}). \quad (13.4)$$

The Banyan network has some useful unique properties. Similar to the Omega network, the Banyan network also has the property of self-routing. Identical to Omega networks, the blocking probability for a Banyan network, $B_{n,d}$, is estimated to be $1 - (1 - p)^{s-1}$, where $s = \log_d n$.

13.3.3 Delta Networks

The *Delta network* consists of a network of crossbar switch elements with shared crosspoints, thus raising the possibility of blocking. The Delta network $D_{n,d}$, using $d \times d$ crossbars, $X_{d,d}$, and with n input/output ports, is defined by

$$D_{n,d} = X_{d,d} \times D_{n/d,d}. \quad (13.5)$$

In a Delta network, only one route between each input i_x and output o_x exists. This fact explains the fundamental reason for blocking. The path uniqueness in this network can be shown by the induction on the number of stages, s . The number of stages is exactly $s = \log_d n$. As with Omega networks, the blocking probability for a Delta network, $D_{n,d}$, is estimated to be $1 - (1 - p)^{s-1}$, where $s = \log_d n$.

If $s = 1$, the network becomes $D_{d,d}$, or simply a crossbar. For $s > 1$, the output o_x belongs to one of d subnetworks denoted by $D_{n/d,d}$. Let s_1 be the first-stage switch,

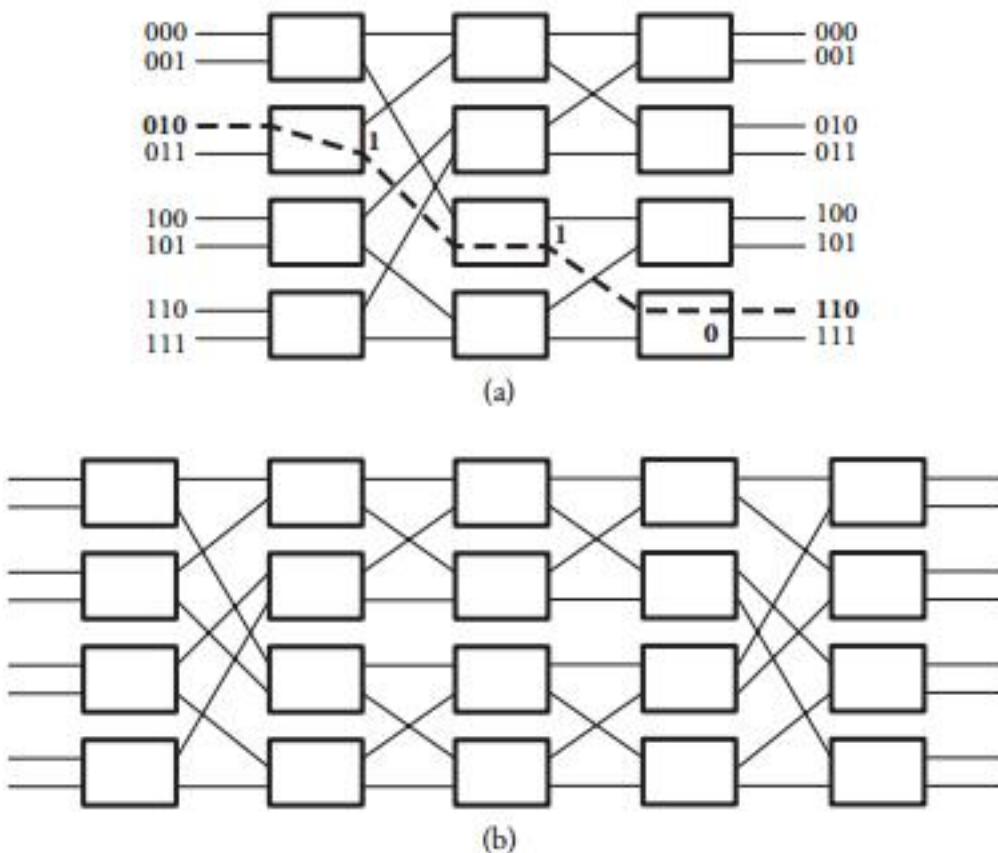


Figure 13.3 (a) A $D_{8,2}$ Delta network with the realization of its routing; (b) extension of the Delta network to a $B_{8,2}$ Beneš

in which i_x is an input. Since there is only one link l connecting s_1 to the subnetwork containing o_x , the only route connecting i_x to o_x is the route consisting of i_x , together with the route connecting l to o_x , which is unique. As seen in Figure 13.3, the Delta network self-routing. A path from any input i_x to output o_y is performed by selecting links determined by successive digits of o'_y 's label. This process is also reversible, so we can route from output o_y back to i_x by following the links determined by successive digits of i'_x 's label.

13.3.4 Beneš Networks

The *Beneš network* consists of a combination of two networks: a Delta network and its mirrored overlapped in one stage, as shown in Figure 13.3. This network is capable of realizing all possible input/output permutations and is defined by $B_{d,d} = X_{d,d} B_{n/d,d} X_{d,d}$ and expanded by

$$B_{n,d} = X_{d,d} B_{n/d,d} X_{d,d}. \quad (13.6)$$

The recursive structure of the network leads us to decomposition of the routing problem into a set of smaller problems corresponding to each of the stages in the recursion. The top-level problem consists of selecting, for each connection, one of the d subnetworks $B_{n/d, d}$ to route through. The routing problems for the d subnetworks can be solved independently. The following algorithm can be used to route a packet to a specific output port:

Define:

- d Crossbar switch element dimension
- $k = \log_d n$
- r Number of stages, $r = 2 \log_d n - 1$
- j Stage number, $j \in \{1, 2, \dots, r\}$
- A Output address of packet
- a_j Address in stage j
- v_j Bit vector specifying outputs that receive copies

Begin Beneš Network Routing Algorithm

For $1 \leq j \leq k - 1 \Rightarrow$

$$a_j = \text{random number } \in [0, d - 1]$$

For $k \leq j \leq r \Rightarrow$

$$a_j = \left\lfloor \frac{A}{d^{r-j}} \right\rfloor \pmod{d}$$

$$v_j = 2^{a_j} \blacksquare$$

With this algorithm, packets in the routing network are guided to the requested outputs. As a result of its structure using Delta networks, we can derive the complexity of the Beneš network by

$$X_c = \frac{n}{d}(2 \log_d n - 1)d^2 = nd(2 \log_d n - 1) \quad (13.7)$$

knowing that $\frac{n}{d}$ is the number of crossbar switches per stage and that d^2 is the complexity of $d \times d$ crossbars.

13.4 Nonblocking Switch Fabrics: Clos Networks

Figure 13.4 shows a *Clos network* with $n = 8$ inputs, $d = 2$, and $k = 5$. If the number of middle-stage crossbar switch elements k is greater than or equal to $2d - 1$, $C_{n,d,k}^3$ is strictly nonblocking. Let d and k be integers with $2 \leq d \leq k$. The three-stage Clos network $C_{n,d,k}^3$ is defined by

$$C_{n,d,k}^3 = X_{d,k} X_{n/d, n/d} X_{k,d}. \quad (13.8)$$

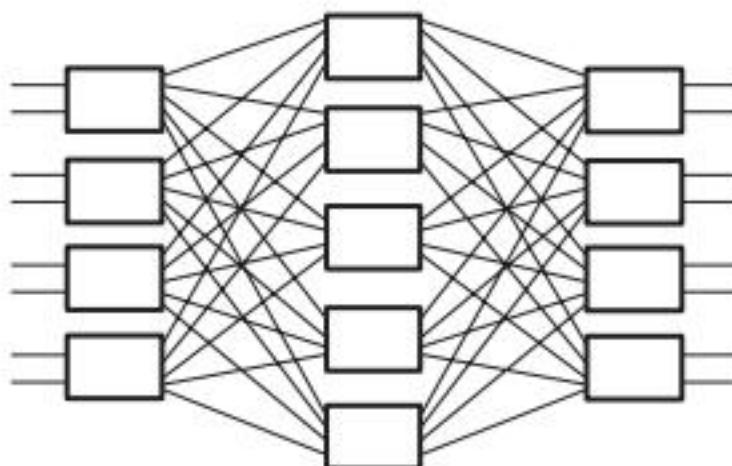


Figure 13.4 A Clos network with $n = 8$ inputs, $d = 2$, and $k = 5$

The proof of this claim can be derived by first observing that a connection through the three-stage switch requires a middle-stage switch element having an idle link from the first stage and an idle link to the third stage, as shown in Figure 13.5. In $C_{n,d,k}^3$, we search for a route realizing a connection request from input x to output y . Thus, the number of outputs of the stage 1 switch elements containing input x that are busy is at most $d - 1$. This means that there are at most $d - 1$ middle-stage switch elements that are not accessible from x . Similarly, there are at most $d - 1$ middle-stage switch elements that are not accessible from y .

Clearly, there are at most $2d - 2$ middle-stage switches that are either not accessible from x or not accessible from y . Hence, the worst-case situation for blocking is that these two sets of middle-stage switch elements are unavailable for connection request (x, y) . However, if one additional free middle-stage switch element exists as shaded in the figure, that element can be used to set up the connection (x, y) . Hence if $k = (d - 1) + (d - 1) + 1 = 2d - 1$, the switch is strictly nonblocking. More generally, at least one middle-stage switch that is accessible from both sides or a state that blocks connection request (x, y) can be found if $k \geq 2d - 1$. The complexity of the three-stage Clos network is

$$\begin{aligned} X_c &= dk \left(\frac{n}{d} \right) + k \left(\frac{n}{d} \right)^2 + dk \left(\frac{n}{d} \right) \\ &= 2kn + k \left(\frac{n}{d} \right)^2. \end{aligned} \tag{13.9}$$

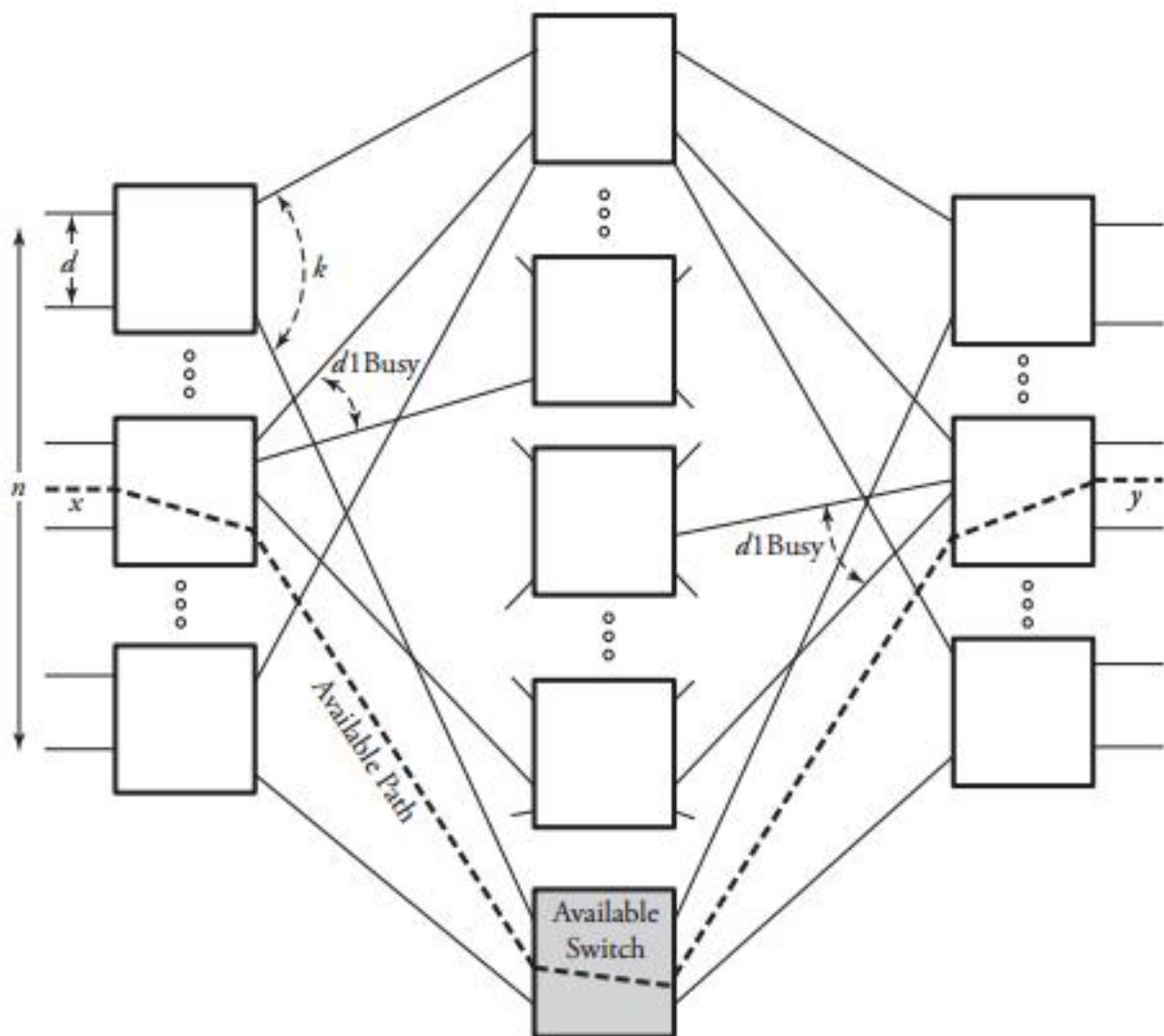


Figure 13.5 Analysis of blocking in the three-stage Clos network

In order to find the complexity of the Clos network under nonblocking conditions, we can substitute $k = 2d - 1$ in Equation 13.9. Thus, the complexity of a nonblocking network, $X_c^{n.b.}$, becomes

$$X_c^{n.b.} = (2d - 1) \left[2n + \left(\frac{n}{d} \right)^2 \right]. \quad (13.10)$$

It is always useful to optimize the complexity of switching networks, especially for the nonblocking Clos networks, in which finding the best d would be beneficial. To achieve this goal, we can optimize the network by

$$\frac{\partial X_c^{n.b.}}{\partial d} = \frac{\partial}{\partial d} \left((2d - 1) \left[2n + \left(\frac{n}{d} \right)^2 \right] \right) = 0. \quad (13.11)$$

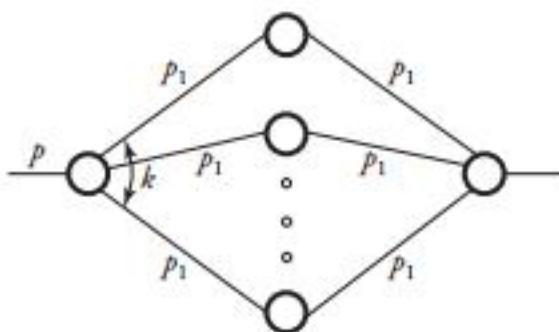


Figure 13.6 A blocking model for Clos switching network

This procedure releases $d = \sqrt{n/2}$, which is the absolute minimized value of d . The optimized crosspoint count under this optimization, therefore, is

$$X_{c,opt}^{n.b.} = 4n(\sqrt{2n} - 1). \quad (13.12)$$

However, the number of crosspoints for large three-stage switches is still quite prohibitive. Large switching systems typically use more than three stages to provide greater reductions in crosspoints. Three-stage Clos networks can be enhanced by substituting another three-stage Clos network for the middle-stage crossbars, resulting in a five-stage Clos network. We can continue in this fashion to construct networks with many stages.

13.4.1 Estimation of Blocking Probabilities

To estimate the internal blocking probability of a Clos network, consider $d \times k$ switch elements in the first stage. Figure 13.6 shows a probability graph of a three-stage network. All possible internal paths for an input/output connection pair are shown. For the middle-stage crossbars, the blocking-probability estimation for n parallel links, each with probability p , is generally $B = p^n$; for a series of n links, the blocking probability is generally $B = 1 - (1 - p)^n$. To apply this rule, let p_1 be the probability that an internal link is busy, as shown in the figure, and thus $1 - p_1$ is the probability that a link is idle. Then, B , the internal blocking probability of the switching network, or the probability that all paths are busy, is

$$B = [1 - (1 - p_1)^2]^k. \quad (13.13)$$

We want the traffic load to balance at both sides of each node. Specifically, at the first node, we want the external and internal traffic to be equal, such that $kp_1 = dp$ or

$p_1 = p(d/k)$. Using this and Equation (13.13) leads to B in terms of p :

$$B = \left[1 - (1 - p(d/k))^2 \right]^k. \quad (13.14)$$

Example. For a Clos network with $k = 8$, $d = 8$, and $p = 0.8$, find the internal blocking probability.

Solution. Obviously, $k/d = 1$, and therefore the internal blocking probability $B = (1 - (1 - 0.8)^2)^8 = 0.72$.

Example. For the Clos network presented in the previous example, if we add up to 100 percent more crossbar switches into the middle stage of the network, how much reduction in internal blocking probability do we obtain?

Solution. Since $k = 16$, $d = 8$, and $p = 0.8$, then $k/d = 2$, and thus $B = ((1 - (1 - 0.8 \times 1/2)^2)^{16} = 0.0008}$. The impact is remarkable, as the internal blocking probability is reduced 900 times.

13.4.2 Five-Stage Clos Networks

Figure 13.7 shows a five-stage Clos network obtained by replacing every middle-stage switch element in a three-stage Clos network. This structure provides less complexity, as the complexity of each middle three-stage network has been reduced as was explained for a three-stage network. Note that this statement can be true only if a five-stage switch is strictly nonblocking, where $k_1 \geq 2d_1 - 1$ and also $k_2 \geq 2d_2 - 1$. This type of design is especially useful for large-scale switching systems, in which a substantial reduction in the overall complexity of a network is targeted. The blocking probability of a five-stage network is modeled in Figure 13.8 and is determined as follows:

$$B = \{1 - (1 - p_1)^2 [1 - (1 - (1 - p_2)^2)^{k_2}]\}^{k_1}. \quad (13.15)$$

13.5 Concentration and Expansion Switches

This section introduces switching networks that either expand or concentrate incoming traffic. In a concentration-based switching network, the number of output ports is less than the number of input ports. In an expansion-based switching network, the number of output ports is more than the number of input ports.

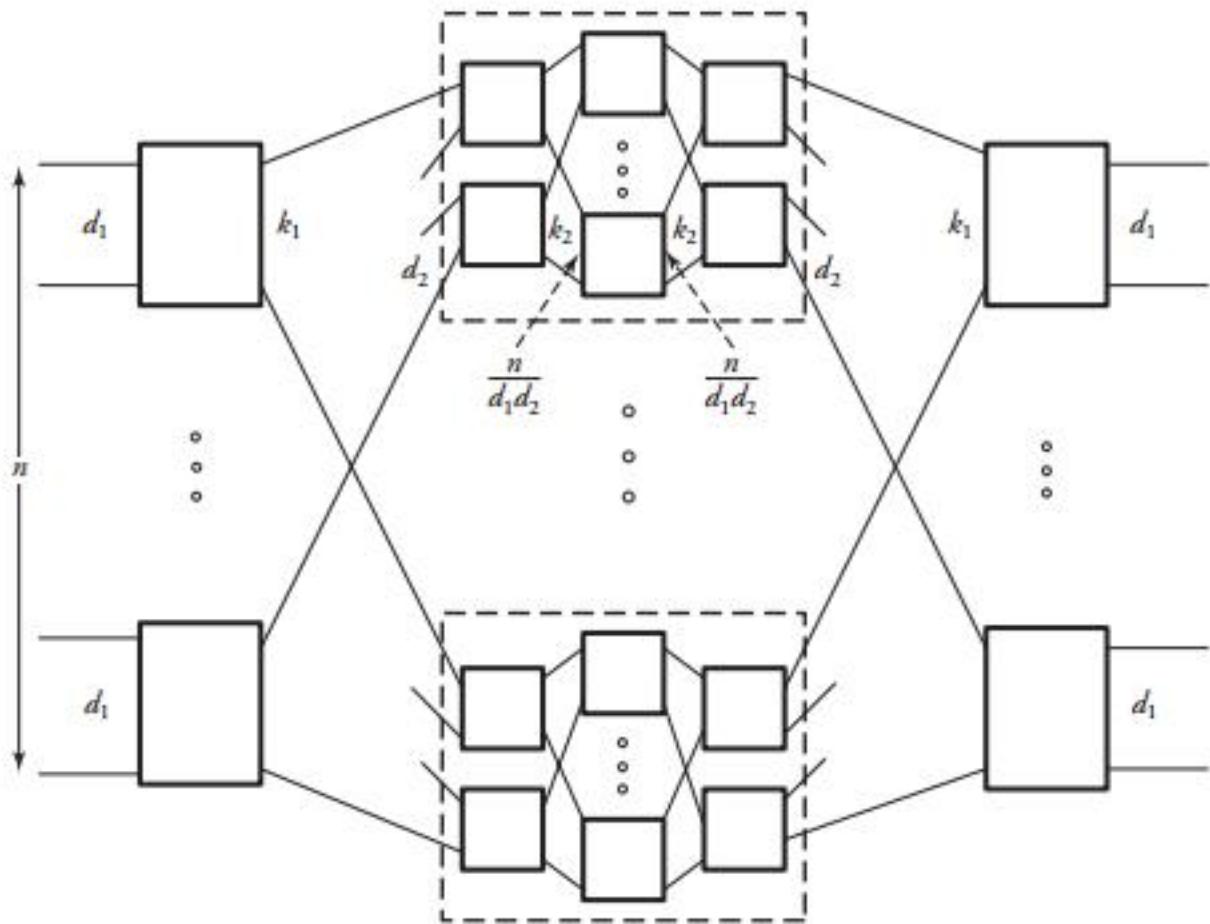


Figure 13.7 Constructing a five-stage Clos network by using three-stage networks to reduce blocking probability.

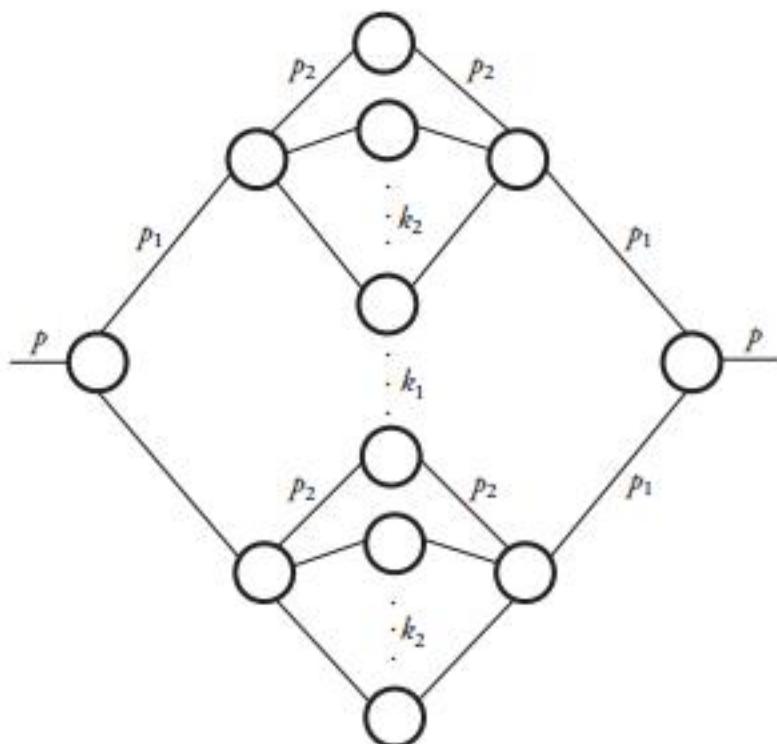


Figure 13.8 A blocking model for the five-stage Clos network

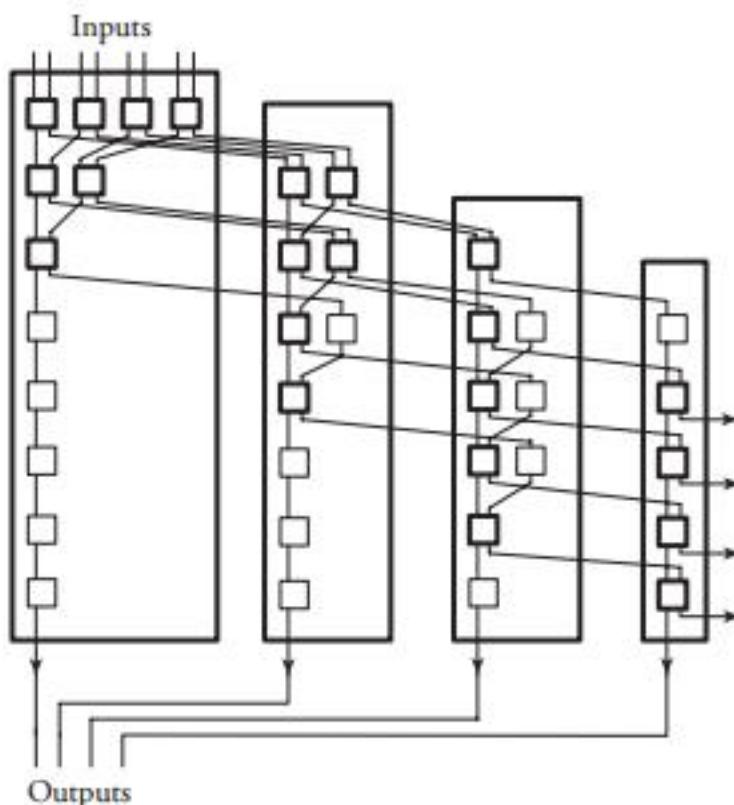


Figure 13.9 The knockout switching network

13.5.1 Knockout Switching Network

One of the simplest *concentration switching networks* is called a *knockout switch*, shown in Figure 13.9. The knockout switch is a blocking network and is constructed with interconnected crossbars with less complexity than a crossbar switch of the same size. The knockout switch is a good substitution for a crossbar switch when the likelihood is small that many inputs will need to send packets to the same output simultaneously.

The idea of this switch is based on the concept of knocking a possible k packets out of n active inputs if the number of switch outputs is m , where $m < n$. To implement this mechanism, consider the example illustrated in Figure 13.9, where $n = 8$ and $m = 4$. Assume that all the eight inputs have packets to transmit to outputs; obviously, the system would need to discard a minimum of four packets. The switching network consists of a number of either 2×2 switch elements and delay elements.

The switch elements act as concentrators, ensuring fairness to all entering packets. The eight incoming packets from eight switch inputs compete in the first four switch elements. A packet that wins the competition, based on a random selection in a switch element, stays in the same column of switches and continues to reach its desired output. Otherwise, a losing packet is knocked out to the next column of switch elements for

the next round of competition. Thus, the first-round losers in this process go straight to the second column and play off against one another and at later stages, face the second- and the third-round losing packets from the first column.

Note that the system needs to ensure that no input port is singled out for bad treatment every time the output is overloaded. This fairness task is implemented by playing packets against one another in a form of a knockout column-by-column scenario to select m winners. In the second column, two packets, including those losing packets, are competing in a similar fashion to reach their desired outputs, and so on. Finally, all the losing packets are dropped, and eventually, we select up to $m = 4$ packets, dropping all the rest. Note that this process requires timing synchronization if all the incoming packets are to experience the same switching delay. To ensure packet time synchronization, the delay elements present units of delay to packets, as shown in the figure by additional delay units.

13.5.2 Expansion Network

An *expansion network* with n_1 inputs and n_2 outputs, where $n_1 < n_2$, can scale up the number of receivers from n_1 to n_2 . A three-stage *expansion network* is shown in Figure 13.10. This network is constructed with three types of crossbars of sizes $d_1 \times m$, $\frac{n_1}{d_1} \times \frac{n_2}{d_2}$, and $m \times d_2$, respectively, and is defined by

$$M(n_1, d_1, n_2, d_2, m) = X_{d_1, m} X_{n_1/d_1, n_2/d_2} X_{m, d_2}. \quad (13.16)$$

An expansion network is a generalization of the three-stage Clos network and is useful in applications in which an incoming signal is to be sent to multiple outputs. Such networks are also called *distribution networks*. Considering the previous network dimensions, the complexity of a three-stage expansion network, X_e , is derived by

$$X_e = d_1 m \frac{n_1}{d_1} + \frac{n_1 n_2}{d_1 d_2} m + d_2 m \frac{n_2}{d_2}. \quad (13.17)$$

An expansion network is nonblocking if $m \geq (d_1 - 1)n_2/d_2 + d_2$. We can prove this claim by assuming that if a given connection goes to two or more outputs of the same third-stage switch, the connection branches in the third-stage switch supply all the outputs. Suppose that we want to add a new end point to an existing connection from an input x to an idle output y . If an output is in the same third-stage switch as y , we can add the necessary branch at that point. Note that $d_2 - 1$ middle-stage switches are inaccessible from y , and at most $(d_1 - 1)n_2/d_2$ are inaccessible from x . Since m is greater than the sum of these two values, at least one middle-stage switch

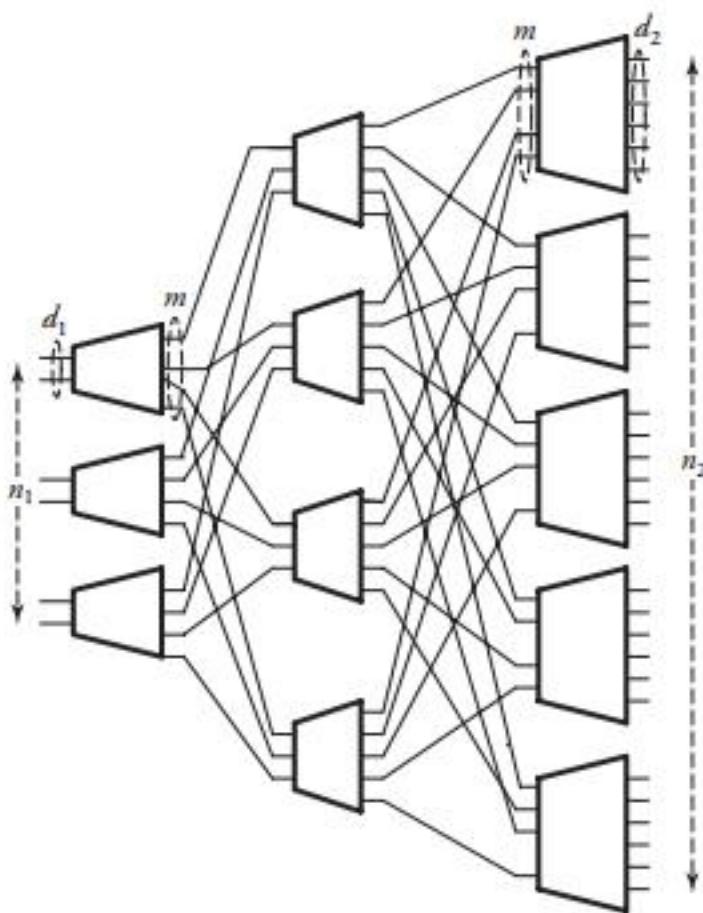


Figure 13.10 A three-stage expansion switching network

must be accessible to both x and y . A similar argument applies if we wish to create a new connection from an idle input x to an idle input y .

Example. Suppose that $n_1 = n_2 = 256$, $d_1 = 16$, and $d_2 = 64$. From the preceding theorem, if $m \geq 1024 + 64 = 1,088$, the network becomes nonblocking.

13.6 Shared-Memory Switch Fabrics

A totally different approach for switching can be achieved in time domain and without using any switch elements: the *shared-memory switch fabric*. As shown in Figure 13.11, the distinctive feature of this switch fabric is the use of a high-speed $n : 1$, time-division multiplexer (TDM) with a bit rate n times as large as the rate on each individual input/output line. Assume packet a at input i needs to be switched on output j and that packet b at input j is supposed to be switched on output i . First, the multiplexer creates an n -channel-long frame from the n inputs. Each frame created at the output

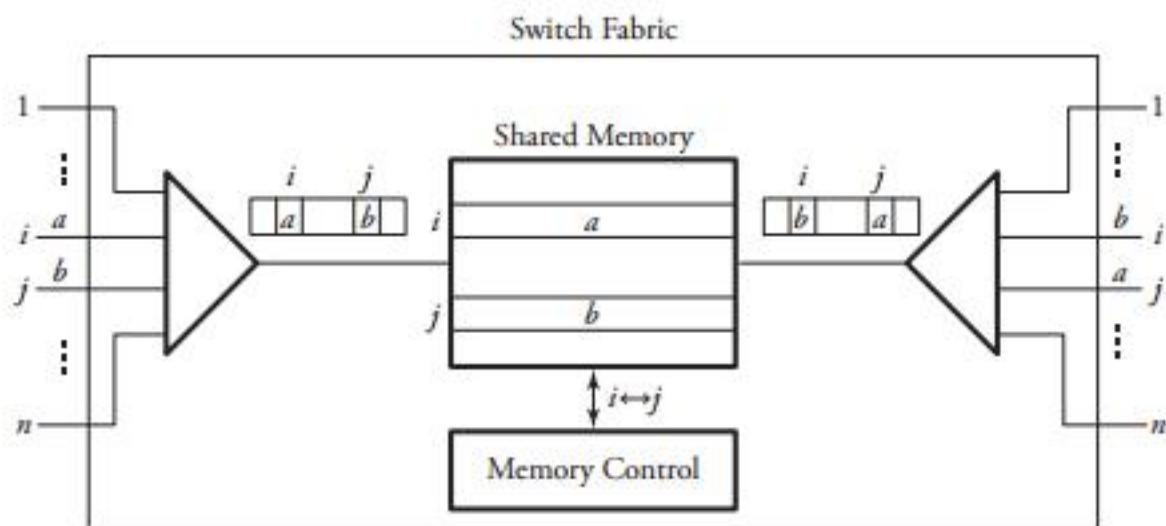


Figure 13.11 Shared-memory switch fabric

of the multiplexer is stored in *shared memory* in the same order as the frame's channels. Thus, *a* is written at location *i*, and *b* is written at location *j* of memory.

To implement the switching action between the two channels *i* and *j*, the *memory control* exchanges the reading order of the two memory locations *i* and *j*. This way, the new frame made out of memory contains packet *a* placed in channel *j* and packet *b* placed in channel *i*. This frame is scanned by a demultiplexer for output ports. If the memory partition for an output is empty, the corresponding minislot remains unfilled. It is a matter of implementation whether frame size in shared memory should be rigid or flexible. Clearly, variable-size frames require more sophisticated hardware to manage, but the packet-loss rate improves, since memory does not suffer from overflow until all memory slots are occupied. See Chapter 3 for more on multiplexers handling variable-size frames.

13.7 Techniques for Improving Performance

Several practical methods can be used to improve the performance of switching systems. The enhancement of a switch performance is typically related to speed, throughput, and therefore the possibility of blocking. Some possible methods for enhancing performance in switching systems are as follows:

- *Buffering.* The use of buffers in switching networks reduces traffic congestion and thus increases system throughput. With the use of buffers in a switch, packets are

not discarded when they request the same switch output but instead are kept in buffers for a later contention resolution.

- *Combined networks.* Combined networks consists of several cascaded switch fabrics, providing extra stages and producing extra paths, although a combined network can also be designed for other purposes, such as multicasting.
- *Randomizing traffic.* This technique is used to distribute traffic evenly across the switching network to prevent local congestion.
- *Recirculation of traffic.* A packet that is not successfully delivered to a given port can be recirculated. Such packets contend for delivery—possibly with higher priorities—in the next cycle.
- *Increase speed-up factor.* This *speed advantage* in a switching system refers to the ratio of the internal link speed to the external link speed.
- *Parallel-plane switching networks.* This technique forms parallel planes of switch fabrics to process slices of a packet in parallel.

The use of buffering, combined networks, and parallel-plane switching networks results in increased system complexity and cost but also better performance. Let's take a look at one such technique.

13.7.1 Parallel-Plane Switching Networks

One possible way to improve the performance and speed of switching networks, especially to reduce the blocking probability, is to arrange parallel planes, or slices, of the switching network, as shown in Figure 13.12. The *Cantor network*, $K_{n,d,k}$, is a widely used example of parallel-plane networks and is constructed by m planes of Beneš switching networks, $B_{n,d}$. The Cantor network is defined by

$$K_{n,d,m} = X_{1,m} B_{n,d} X_{m,1}. \quad (13.18)$$

The Cantor network is strictly nonblocking whenever

$$m \geq \frac{2}{d}(1 + (d - 1) \log_d(n/d)). \quad (13.19)$$

For $d = 2$, this reduces to $m \geq \log_2 n$, which gives us a strictly nonblocking network with approximately $4n(\log_2 n)^2$ crosspoints.

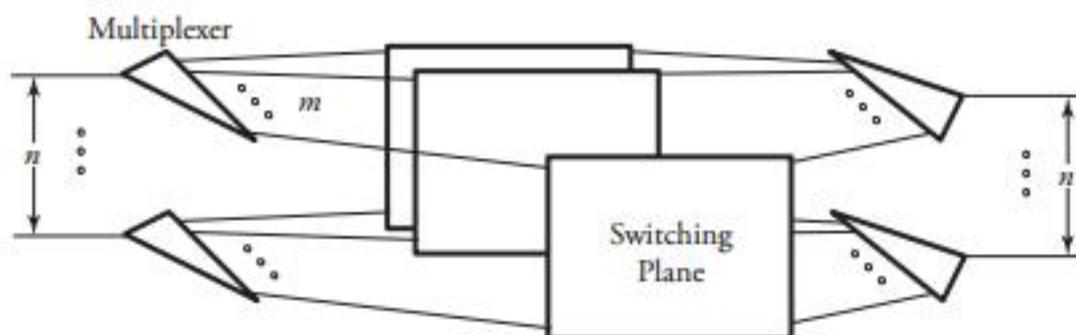


Figure 13.12 A Cantor network with m parallel switching planes for better reliability

Example. For $n = 2^{12}$, compare crossbar, Clos network, and Cantor network complexities.

Solution. If $n = 2^{12}$, the complexity for a crossbar is $n^2 = 2^{24} \approx 17 \times 10^6$. For a Clos network, the complexity is $4\sqrt{2}n^{3/2} = 1.5 \times 10^6$. For a Cantor network, the complexity is $4n(\log n)^2 = 2^{24} = 2.4 \times 10^6$.

In parallel-plane switching networks, the chip pin constraint limits the number of signals that can enter or leave a physical component. The integrated circuit pin constraints also influence implementation so that such systems are best designed with each data path passing through physically separate components. Such *bit-sliced organization* causes complexity to grow in proportion to the data path width.

13.8 Case Study: Multipath Buffered Crossbar

For our case study on a switching fabric, we choose an expandable switch fabric constructed with identical building-block modules called the *multipath buffered crossbar* (MBC). Conventional $n \times n$ crossbars are vulnerable to faults. Unlike a conventional crossbar, MBC is a crossbar with n rows and k columns. Each row contains an input bus and an output bus. A packet being transferred from input i to output j is sent by input port processor i on input bus i to one of the k columns. The packet then passes along this column to the output bus in row j . The crosspoints that make up the MBC switch fabric include mechanisms that allow the inputs to contend for access to the various columns. The crosspoint buffers in one row act as a distributed output buffer for the corresponding output port. As shown in Figure 13.13, every crosspoint has two different *switches* and a buffer.

Consider a case of point-to-point connection from input i_1 to output a_1 . The packet is first randomly sent to a crosspoint at one of the k *shared data buses*, say, column

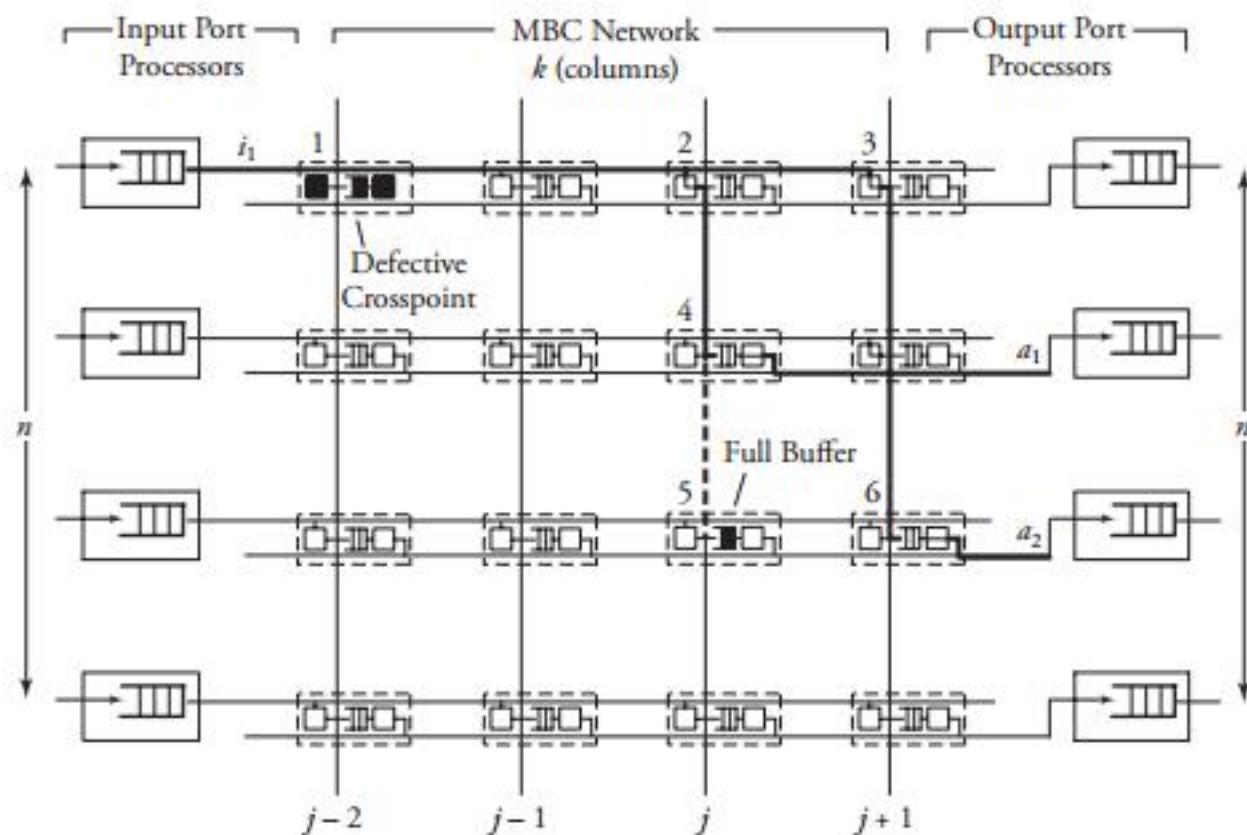


Figure 13.13 Routing and multicasting in the multipath buffered crossbar ($n = k = 4$)

($j - 2$). In case the crosspoint on this bus is not available for any reason, another crosspoint selected at random is examined, and so on. The reason for randomizing the initial location is to distribute the traffic as evenly as possible. When a functioning crosspoint (crosspoint 3) at column j is selected, the packet can be sent to a second crosspoint (crosspoint 4) at this column unless the selected crosspoint is found to be faulty or its buffer is full. The packet is buffered in crosspoint 4 and can be sent out after a contention-resolution process in its row.

The availability of each crosspoint at a given node (i, j) is dynamically reported to the input ports by a flag $g_{i,j}$. If more than one input port requests a given column, based on a contention-resolution process, only one of them can use the column. Packets in the other input ports receive higher priorities to examine other columns. Obviously, the more the packet receives priority increments, the sooner it is granted one free column, which it selects randomly. Once it gets accepted by one of the β buffers of a second crosspoint, the packet contends to get access to its corresponding output. The losers of the contention resolution get higher priorities for the next contention cycle until they are successfully transmitted. In Figure 13.13, the connections $i_1 \rightarrow a_1$ and $i_1 \rightarrow a_2$ are made possible through nodes $\{2, 4\}$ and nodes $\{3, 6\}$, respectively.

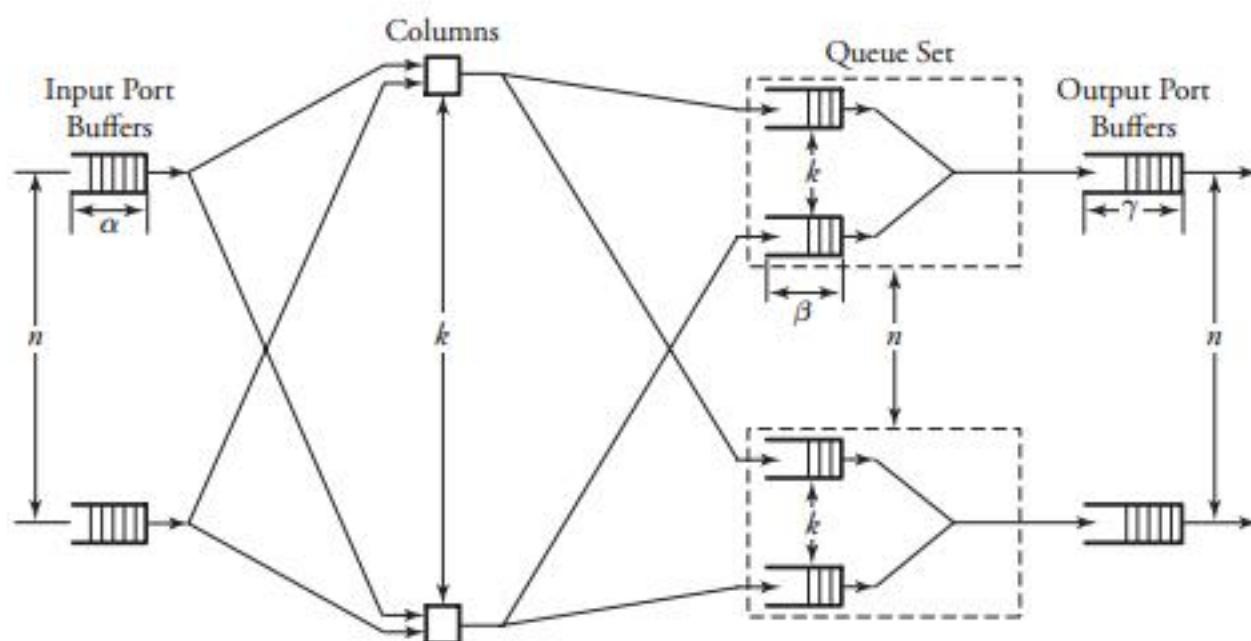


Figure 13.14 Queueing model for multipath buffered crossbar

13.8.1 Queueing Model

The queueing model for this system is shown in Figure 13.14. Packets enter the network from n *input port buffers* and pass through any of the k crosspoints belonging to the same row as the input. These crosspoints do not contribute in any queueing operation and act only as ordinary connectors. Each packet exiting these nodes can confront n crosspoint buffers of length β that are situated on one of the k -specified columns. A packet selects a buffer on the basis of the desired address. In this system architecture, each crossbar row contains k buffers, constructing a related group of buffers called a *queue set*, as seen in the figure. The winning packet of each set is then permitted to enter the corresponding *output port buffer*.

Note that the state of a buffer within each queue set is dependent on the state of all other buffers in that queue set. There are no *grant flows* between the output port buffers and the crosspoint buffers, and the admission of a packet by a queue set is made possible by granting the *flag acknowledgments* to the input port buffers. It should also be noted that the analysis that follows assumes that all crosspoints are functional. Let $z_k^\beta(s)$ be the number of ways to distribute s packets among k distinct buffers within a queue set, under the restriction that each buffer may contain at most β packets. Note that here, we are not concerned about the number of combinations of different packets in the queue. Therefore, $z_k^\beta(s)$ can be recursively computed by

$$z_k^\beta(s) = \begin{cases} 1 & \text{if } (k = 1 \wedge s \leq \beta) \vee (s = 0) \\ 0 & \text{if } s > k\beta \\ \sum_{0 \leq i \leq \min\{\beta, s\}} z_{k-1}^\beta(s-i) & \text{if } 0 < s \leq k\beta. \end{cases} \quad (13.20)$$

Let $X_k^\beta(r, s)$ be the probability that a given crosspoint buffer has r packets when the entire queue set contains s packets. Then

$$X_k^\beta(r, s) = \frac{z_{k-1}^\beta(s-r)}{z_k^\beta(s)}. \quad (13.21)$$

To further extend this analysis, an expression for the flow control between a crosspoint queue set and the input buffers is required. A second crosspoint generates a final flag acknowledgment, Ψ , after having completed a three-step process. The probability that a packet is available to enter the network, a , is the same probability that the input port buffer is not empty and is given by

$$a = 1 - \pi_i(0). \quad (13.22)$$

Let ψ be the probability that a packet contending for access to a column wins the contention. Note that the probability that any given input attempts to contend for any given column is a/k . Hence,

$$\psi = \sum_{0 \leq c \leq n-1} \frac{1}{c+1} \binom{n-1}{c} (a/k)^c (1-a/k)^{(n-1)-c}, \quad (13.23)$$

where $\binom{n-1}{c}$ is a *binomial* coefficient indicating the number of different combinations of $n-1$ possible existing contenders on one of the k columns, taken c contenders at a time. Now, the probability that a packet forwarded by input port buffer i is admitted, Ψ , is given by:

$$\Psi = \psi \sum_{0 \leq s \leq b} \pi_i(s) [1 - X_k^\beta(\beta, s)]. \quad (13.24)$$

We use Ψ later to derive the probability that an input port buffer contains exactly a certain number of packets.

13.8.2 Markov Chain Model

The expression for Ψ facilitates the determination of the state of the input port buffers. The input port queue can be modeled as a $(2\alpha + 1)$ -state *Markov chain* (see Figure 13.15). The transition rates are determined by the offered load ρ and

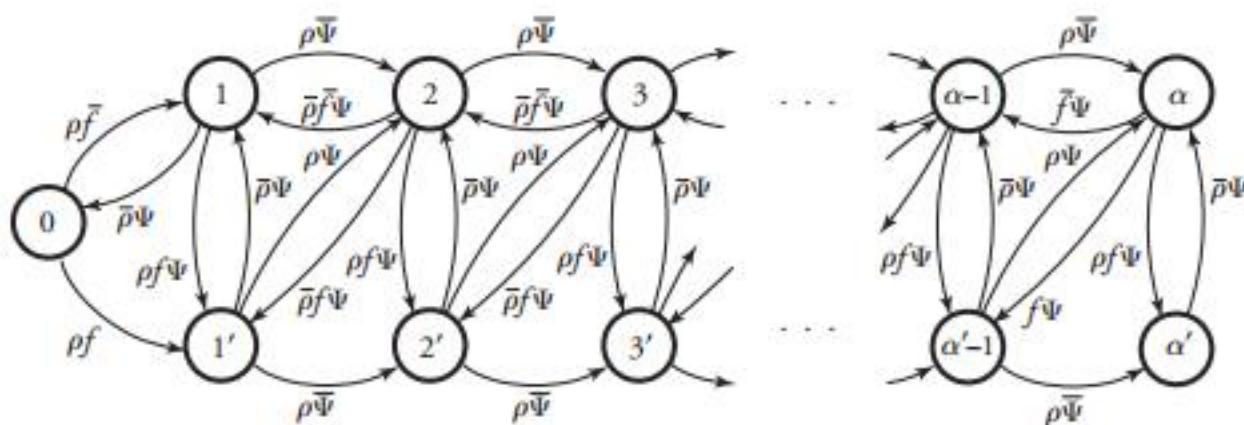


Figure 13.15 Markov chain model of the input port queue

the crosspoint queue-set grant probability Ψ . Let f be the probability that a packet has fanout 2 (a *bipoint connection*). In this model, a bipoint connection, which is a two-copying packet-recirculation task, is treated as two independent *point-to-point* connections. The upper row of the Markov chain depicts the state of the queue when a packet needs no additional copy. However, if the packet has fanout 2, it proceeds to the second copy of the packet on the upper row once the operation for its first copy in the lower row of the chain is completed.

The state of the queue can be changed from 0 to 1 with transition probability $\rho\bar{f}$ if the packet has fanout 1, or it can enter state $1'$ with transition probability ρf if the packet has fanout 2. On the arrival of a fanout-2 packet at the head of queue, a state in the upper chain is changed to a lower one with probability of $\rho f\Psi$ if a new packet arrives at the queue, or $\bar{\rho}f\Psi$ if no new packet arrives. Similarly, any of the lower chain states can be changed with probabilities $\rho\Psi$ or $\bar{\rho}\Psi$ to an upper one as soon as the first copy of a fanout-2 packet is completed. These transition probabilities are free of f , since only the first copy has been completed at this point. The second copy is processed in the upper chain independently. The probability that an input port buffer contains exactly s packets is computed recursively by

$$\begin{aligned}\pi_i(s) = & \pi_i(s-1)\rho\bar{\Psi} + \pi_i(s'-1)\rho\Psi + \pi_i(s)(\bar{\rho}\bar{\Psi} + \rho\bar{f}\Psi) \\ & + \pi_i(s')\bar{\rho}\Psi + \pi_i(s+1)\bar{\rho}\bar{f}\Psi\end{aligned}\quad (13.25)$$

and

$$\pi_i(s') = \pi_i(s'-1)\rho\bar{\Psi} + \pi_i(s)\rho f\Psi + \pi_i(s')\bar{\rho}\bar{\Psi} + \pi_i(s+1)\bar{\rho}f\Psi. \quad (13.26)$$

Next, let $w_k^\beta(r, s)$ be the number of distributions that leave exactly r buffers of capacity β "not full" while a queue set contains s packets, and let $W_k^\beta(r, s)$ be the probability that exactly r buffers are not full when a queue set contains s packets. Therefore:

$$w_k^\beta(r, s) = \binom{k}{r} z_r^{\beta-1} (s - (k-r)\beta), \quad (13.27)$$

and then,

$$W_k^\beta(r, s) = \frac{w_k^\beta(r, s)}{z_k^\beta(s)}. \quad (13.28)$$

Now, we need to compute the probability that exactly j packets enter a queue set when the queue is in state s at the beginning of the current packet cycle. The probability that a packet is available to enter a particular buffer of a queue set, a_x , is calculated based on a that was introduced earlier and is given by

$$a_x = \frac{1}{n} \left[1 - \left(1 - \frac{a}{k} \right)^n \right]. \quad (13.29)$$

Then, $p(j, s)$ is

$$p(j, s) \approx \sum_{j \leq r \leq k} W_k^\beta(r, s) \binom{r}{j} (a_x)^j (1 - a_x)^{r-j}. \quad (13.30)$$

We also need to compute the probability that exactly j packets leave a queue set when the queue is in state s at the beginning of the current packet cycle. It should be remembered that for $q(j, s)$, there is only one output for every queue set, owing to the queueing model, so the probability is introduced by a simple form of

$$q(j, s) = \begin{cases} 0 & \text{if } (s = 0 \wedge j = 1) \vee (s = 1 \wedge j = 0) \vee (j > 1) \\ 1 & \text{if } (s \geq 1 \wedge j = 1) \vee (s = 0 \wedge j = 0). \end{cases} \quad (13.31)$$

Each queue set is modeled as a $(b + 1)$ -state *Markov chain* ($b = k\beta$). The transition probability, $\lambda(\hat{s}, s)$, and the next-state probability, $\pi_x(s)$, defined earlier, can be determined as follows:

$$\lambda(\hat{s}, s) = \sum_{h=\max\{0, s-\hat{s}\}}^{\min\{k, b-s\}} p(h, \hat{s}) q(h - (s - \hat{s}), \hat{s}), \quad (13.32)$$

where $0 \leq (s \text{ and } \hat{s}) \leq b$, and

$$\pi_x(s) = \sum_{\hat{s}=\max\{0, s-k\}}^{\min\{b, s+k\}} \pi_x(\hat{s}) \lambda(\hat{s}, s). \quad (13.33)$$

By having $\pi_x(s)$, we can estimate the system delay and throughput.

13.8.3 Throughput and Delay

By putting together all the preceding parameters, we can compute the system *throughput* (T) and *delay* (D). The throughput refers to the number of packets leaving a queue set per output link per packet cycle. Since the entire network is modeled as one middle-stage queue, the throughput, on the other hand, is the probability of having at least one packet in a queue set and is simply

$$T = 1 - \pi_x(0). \quad (13.34)$$

Clearly, from the definition given by Equation (13.31), $q(j, s) \in \{0, 1\}$. Thus, so long as at least one packet is in the queue set, $q(j, s) = 1$, and T can be expressed free of $q(j, s)$. The *delay* is defined as the number of packet cycles that a given packet spends in the crossbar from network entry until network exit. This is another interpretation of the ratio of *queue length* and the *arrival rate*. Let the queue length of the network and the input port processors be denoted by Q_x and Q_i , respectively. They are expressed by

$$Q_x = \sum_{0 \leq s \leq b} s \pi_x(s) \quad (13.35)$$

and

$$Q_i = \sum_{0 \leq j \leq \alpha} j \pi_i(j). \quad (13.36)$$

The arrival rate at the input port buffers is the offered load, ρ , and the arrival rate for each queue set can be defined by A :

$$A = \sum_{0 \leq j \leq k} \sum_{0 \leq s \leq b} \pi_x(s) j p(j, s). \quad (13.37)$$

Thus, the total delay is the summation of the delays:

$$D = \frac{Q_x}{A} + \frac{Q_i}{\rho}. \quad (13.38)$$

The delay (D) measurement is based on three main variables: ρ , n , and k . Figure 13.16 shows a set of results. This figure shows variations of incoming traffic load

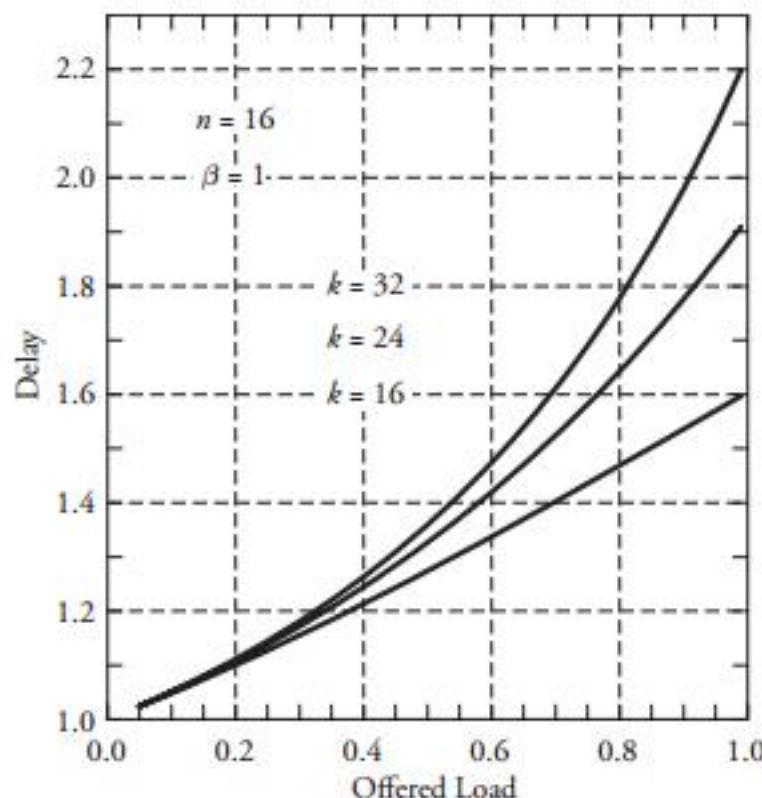


Figure 13.16 Delay (D) of a 16-port network when offered load (ρ) and number of columns (k) vary

on a 16-port MBC network. In these examples, $\beta = 1$, and the fraction of multipoint packets is assumed to be $f = 0.75$. As expected, the throughput is improved when k increases as 16, 24, 32, mainly because of the availability of a larger number of columns for packets waiting in the input buffers. Delay curves show that the average time that packets spend in the network cannot exceed 2.2 packet times. This amount of time was predictable, since packets pass through only two stages of switching from an input port to their destinations. This behavior explains the better performance capability of an MBC switch fabric compared to a typical multistage switch fabric.

13.9 Summary

Switch fabrics are the core segments of switching devices as routers. A *crossbar* is a nonblocking switch and can be used as the building block of other switch fabrics. A crossbar requires n^2 crosspoints. Blocking switches constructed with crossbar switches are *Omega*, *Banyan*, and *Delta* networks, each requiring $\log_d n$ stages; the fourth, the *Beneš* network, requires $2 \log_d n - 1$ stages. The *Beneš* network consists of a combined *Delta* network and its mirrored *Delta* network. *Clos networks* are nonblocking switches constructed with crossbars. A $C_{n,d,k}^3$ Clos network is strictly nonblocking if $k \geq 2d - 1$.

Concentration-based and *expansion-based* switching networks are two special-purpose switch fabrics.

In a different approach for switching, *time domain*, a memory controlled by another memory, without using any crossbar, performs the task of switching. Finally, various techniques such as deploying buffers in the structure of switches, offer better performance. The case study presented a switch fabric constructed with a crossbar and buffers in its structure. We estimated the delay and throughput for this switch.

Having finished networking issues in switch fabrics, we are ready to study the basics of optical switching networks and other optical networking topics. An optical network is the backbone of high-speed networking and is presented in the next chapter.

13.10 Exercises

1. Compare the complexity of crossbars and Delta networks in one plotting chart, in which the network size varies over the range $n = 2^2, 2^4$, and 2^5 . For the Delta networks, assume that $d = 2$.
2. For the following switching networks $D_{16,2}$ and $D_{16,4}$:
 - (a) Sketch the networks.
 - (b) Compare $D_{16,2}$ and $D_{16,4}$ in terms of complexity and communication delay.
3. For the following switching networks $\Omega_{16,2}$ and $\Omega_{16,4}$:
 - (a) Sketch the networks with the minimum number of stages that provide all possible input/output connections.
 - (b) Compare the two networks in terms of complexity and communication delay.
4. Using the blocking-probability estimation method, derive an expression for the internal blocking of the two networks $\Omega_{16,2}$ and $\Omega_{16,4}$.
5. For the following switching networks, $B_{16,2}$ and $B_{16,4}$:
 - (a) Sketch the networks.
 - (b) Compare $B_{16,2}$ and $B_{16,4}$ in terms of complexity and communication delay.
6. Using Lee's method (discussed in Chapter 7), derive an expression for the internal blocking of $B_{16,2}$ and $B_{16,4}$ networks.
7. For the following switching networks $Y_{16,2}$ and $Y_{16,4}$:
 - (a) Sketch the networks.
 - (b) Is there any self-routing rule existing for Banyan networks?
8. For the following switching networks $B_{9,3}$ and $\Omega_{9,3}$:
 - (a) Sketch the networks.

- (b) Compare $B_{9,3}$ and $\Omega_{9,3}$ in terms of complexity, internal blocking, and communication delay.
9. Delta networks can be improved by extending stages of switching. For d , s , and h with $h < s$ and $n = d^s$, we define the extended Delta network $D_{n,d,h}^E$ as
$$D_{n,d,h}^E = D_{d^h,d} D_{d^{k-h},d} D_{d^h,d}.$$
 - Sketch a picture of an extended Delta network with eight inputs/outputs, using switch elements of size 2.
 - Derive an equation for the complexity of this network.
 - Derive an equation for the blocking probability.

10. Design an $n = 8$ port three-stage Clos switching network.
 - Find d and k that meet the minimum nonblocking condition, and sketch the network.
 - To design a nonblocking Clos network, the network is nonblocking as long as $k = 2d - 1$. Why would anyone want to design a network with $k > 2d - 1$?

11. Compare the complexity of large-scale three-stage and five-stage Clos switching systems, where the number of ports varies from 1,000 ports to 2,000 ports.

12. For any nonblocking switching network, Lee's method can still estimate some blocking. Justify this contradiction.

13. There are two ways to design a three-stage Clos network with six inputs and six outputs yielding the "minimum" nonblocking operation.
 - Select dimensions for both types of crossbars used in either of the two types of networks, and sketch the networks.
 - Show a Lee's model for both networks, and find the total blocking probability for any of the networks.
 - Which choice is better? (Compare them from all possible aspects.)

14. To design a five-stage Clos network with $n = 8$ and $d = 2$ yielding "minimum" nonblocking operation:
 - Select dimensions for all three types of crossbars—no complexity optimizations needed—and sketch the network.
 - Show a Lee's model.
 - Assuming the probability that any link in the networks is busy is $p = 0.2$, find the total blocking probability for the network.

15. To increase the speed and reliability of the five-stage Clos network $C_{8,2,3}^5$ using $C_{4,2,3}^3$, we form three multiplexed parallel planes, each of which contains this network.

- (a) Sketch the overview of the network.
 - (b) Show a Lee's model.
 - (c) Assuming the probability that any link in the networks including multiplexing links is busy is $p = 0.2$, find the total blocking probability for the network.
16. To design a five-stage Clos switching network with n inputs and n outputs:
- (a) Give dimensions for all five types of crossbars to yield nonblocking operations.
 - (b) Select dimensions for all five types of crossbars to yield nonblocking operation and minimum crosspoint count. (No optimizations is needed.)
17. Design a shared-memory switching system that supports up to 16 links and uses a multiplexer, RAM, and a demultiplexer. The multiplexer output carries 16 channels, each as wide as a segment (packet fragment). Each segment is as large as 512 bytes, including the segment header for local routing. A total of $0.4 \mu s$ form a frame at each multiplexer. RAM is organized in 32-bit words with 2 ns write-in time, 2 ns read-out time, and 1 ns access time (controller process) per word.
- (a) Find the minimum required size of RAM.
 - (b) Find the size of a RAM address.
 - (c) Find the maximum bit rate that can be supported at the output of RAM.
 - (d) Find the speed of this switch, which is the segment-processing rate per second per switch port.
18. *Computer simulation project.* Write a computer program to simulate a 2×2 crossbar switch.
- (a) Assign each packet a switch-output destination at random.
 - (b) Construct and simulate a single crosspoint. Clearly demonstrate how it switches on and off, using a central crossbar controller.
 - (c) Extend your program to construct a four-crosspoint crossbar.
19. *Computer simulation project.* Carry the program you developed for a single buffer in Chapter 11 (computer simulation project) and extend the preceding 2×2 crossbar switch to a switch with a simple buffered input port. Each of the two inputs of the switch has a buffer with $K = 64$ buffer slots, and each slot can fit only in a packet of size 1,000 bytes. Dynamically assign packets of different size every 1 ms, and send out a packet to the switch every t seconds from the buffer.
- (a) Assign each packet a switch-output destination at random.
 - (b) Construct, simulate, and test each of the two buffers individually.
 - (c) Integrate these two buffers with the switch, and measure the performance metrics of both buffers together, given different values of t .
 - (d) Measure average delay for a packet.

CHAPTER 14

Optical Networks and WDM Systems

Optical communication systems have been developed to meet two main objectives: reaching higher data transmission bandwidths and reducing the overall cost of communication networks. Optical communication technology uses principles of light emission in a glass medium, which can carry more information over longer distances than electrical signals can carry in a copper or coaxial medium. This chapter focuses on principles of optical communications, optical multiplexing, and switching used in optical computer networks. Following are the major topics covered:

- *Overview of optical networks*
- *Basic optical networking devices*
- *Large-scale optical switches*
- *Optical routers*
- *Wavelength allocation in networks*
- *Case study of SSN, an all-optical switch*

Some basic optical devices are *optical filters*, *wavelength-division multiplexers* (WDM), *optical switches*, and *optical buffers* and *delay lines*. Optical switches are given special attention, as they are core engine of all-optical networks. Contention-resolution methods for optical switches are explained and types of switch elements identified.

In optical networks, a number of optical routing devices are interconnected by optical links. The routing devices have optical multiplexing, switching, and routing

components, and links are optical fibers. An important issue in optical networks is *wavelength reuse and allocation*. To keep lightpaths separated on a link, they should be allocated different wavelengths. The case study at the end of the chapter describes an optical switching network with a *spherical switching network* (SSN) topology.

14.1 Overview of Optical Networks

Optical fibers can transmit digitized light signals over long distances because of the purity of glass fiber combined with improved electronics technology. With some acceptable transmission loss, low interference, and high-bandwidth potential, an optical fiber is almost an ideal transmission medium. Optical communication systems may sometimes be combined with electrical components. In such systems, electrical data bits are converted into light, using a certain *wavelength*; when the transmission is completed, the optical signal is converted back to an electrical signal, which is then passed to higher layers that manage switching, restoration, and grooming.

Optical networks can provide more bandwidth than regular electronic networks can. Major advantages of partial optical networks are gained by incorporating some electronic switching functions. The optical network shown in Figure 14.1 consists of *optical nodes* interconnected with physical *optical links*. A point-to-point physical optical link may consist of several logical paths called *lightpaths*, each carrying a different wavelength. A lightpath is set up by assigning a dedicated wavelength to it. The data can be sent over the lightpath once it is set up. A lightpath can create virtual neighbors out of nodes that may be geographically far apart in the network. A lightpath uses the same wavelength on all fiber links through which it passes.

14.1.1 Protocol Models and Standards

Optical networks provide routing, grooming, and restoration of data at the wavelength level and developed from the emergence of the optical layer in the transport and network layers. The challenges of optical networks are network management in the presence of different wavelengths and keeping the cost down to be able to add new services. An optical network's agility gives it the speed and intelligence required for efficient routing. Thereby improving network management and faster connections and network restorations. For an optical system with many channels on a single fiber, a small fiber cut can potentially initiate multiple failures, causing many independent systems to fail.

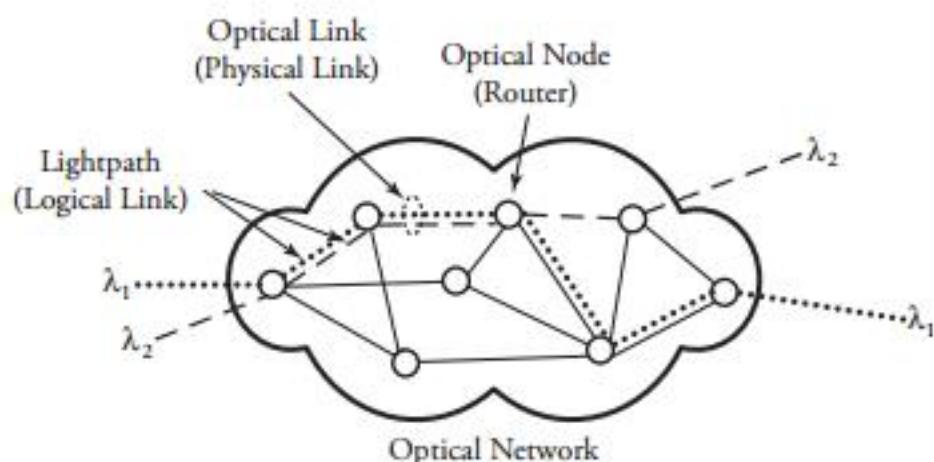


Figure 14.1 Overview of an optical network

Several standards apply in the optical communication environment. The widely used *synchronous optical network* (SONET) standard provides the transport infrastructure for worldwide telecommunications and defines interface standards at the physical layer of the protocol stack. SONET also defines a hierarchy of interface rates that allow data streams at different rates to be wavelength multiplexed. SONET establishes *optical carrier* (OC) levels ranging from level 1 at 51.8 Mb/s (OC-1) to level-192 at 9.95 Gb/s (OC-192). Communication carriers throughout the world use this technology to interconnect their existing digital carriers and fiber optic systems.

In optical networks, client layers and optical layers are managed separately, with no direct interaction between the clients and optical-layer equipment. A centralized network-management system provides optical-layer connections. However, distributed control protocols are used to handle network protection and restoration. Because of the centralized management scheme for connection provisioning, a network may not be reliable. From a control and management perspective, there is a great deal of interest in obtaining a closer interaction between the network layer and the optical layer.

Two popular models are the *overlay model* and the *peer model*. In the overlay model, the optical layer and the network (IP) layer have their own independent control planes. Here, the client layer interacts with the optical layer through a *user-to-network interface* (UNI), and optical-layer elements talk to each other through a *network-to-network interface* (NNI). In an overlay model, the optical network topology is hidden from the client layer through UNI. In the peer model, the network layer and the optical layer run the same control-plane software. Routers have full topology awareness of the optical layer. But the peer model is complicated because the optical layer imposes

significantly different constraints from those on the IP layer. This elegant model has a closer coupling between the IP layer and the optical layer.

14.2 Basic Optical Networking Devices

An optical network comprises optical devices interconnected by optical transmission links. Other basic elements on an optical networks are: tunable lasers, optical buffers or delay elements, optical amplifiers, optical filters, wavelength-division multiplexers, and optical switches.

14.2.1 Tunable Lasers

Tunable lasers can continuously change their emission wavelengths, or colors, in a given spectral range. These changes work in collaboration with optical switches to select a particular wavelength for connection establishment. A *tunable dispersion compensator* is used to compensate for fiber-dispersion losses over the length of the fiber.

14.2.2 Optical Buffers or Delay Elements

Buffering in optical nodes also faces serious limits forced by optical-technology shortcomings. *Optical buffers*, or *optical delay elements*, can be implemented by using a certain length of fiber to delay signals. No practical optical memory is available with the current technology.

14.2.3 Optical Amplifiers

Often in non-all-optical networks, a signal may not be able to remain in optical form and may have to be regenerated, requiring the *amplification* of the signal or the conversion of the signal from optical to electronic. The use of *optical amplifiers* allows achieving large-distance communication without the need of a regenerator. A major milestone in the evolution of optical fiber transmission systems was *Erbium-doped fiber amplifiers* (EDFAS), or simply optical amplifiers. An optical amplifier can amplify signals at many wavelengths simultaneously. The amplification of a signal involves the extraction of the clock from the signal and reclocking the signal, resulting in the creation of “speed bumps” in a network.

14.2.4 Optical Filters

Signal filtering is often needed in optical networks. An *optical filter* equalizes the gain of transmission systems and filters the noise or any unwanted wavelength. In

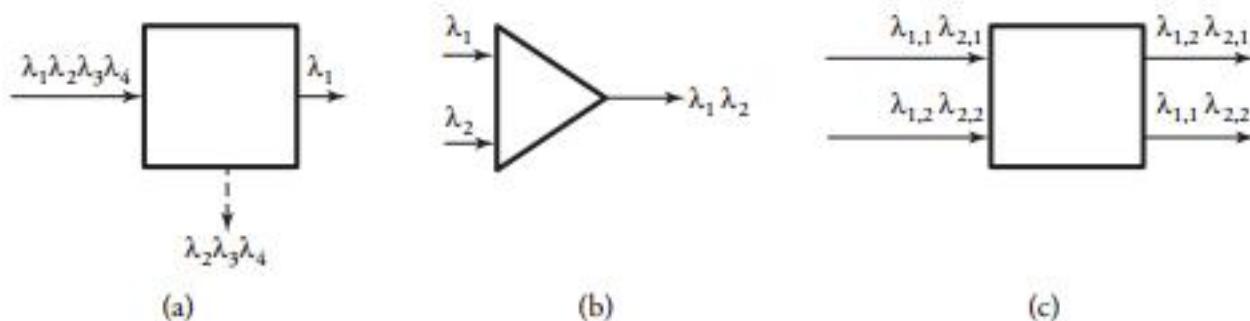


Figure 14.2 Basic communication devices for optical networks: (a) optical filter; (b) wavelength-division multiplexer; (c) optical switch

Figure 14.2 (a), an optical cable carries four wavelengths, λ_1 , λ_2 , λ_3 , and λ_4 , and is connected to a filter. This particular filter is designed to allow only λ_1 to pass and to filter λ_2 , λ_3 , and λ_4 .

The design of an optical filter has a number of challenging factors. *Insertion loss* is one and is the loss of power at a filter. A low insertion loss is one of the specifications for a good optical filter. The state of polarization of the input signals should not affect the loss.

Keeping the temperature at the desired degree in optical systems, especially on filters, is another factor. Temperature variation should not affect the passband of a filter. As a result, wavelength spacing between adjacent channels on transmission systems should be large enough so that the wavelength shift does not affect the entire operation. The passband of filters in an optical system can be narrower if a number of filters are cascaded. The objective is that at the end of cascading the broad passband, each filter causes a small change in operating wavelengths.

14.2.5 Wavelength-Division Multiplexer (WDM)

Wavelength-division multiplexing (WDM) transmission systems divide the optical-fiber bandwidth into many nonoverlapping optical wavelengths: so-called WDM channels. As shown in Figure 14.2 (b), a WDM mixes all incoming signals with different wavelengths and sends them to a common output port. A demultiplexer does the opposite operation, separating the wavelengths and dispatching them onto output ports. On the common link, each channel carries information at light speed with minimal loss. This multiplexing mechanism provides much higher available transmission capacity in communication networks.

14.2.6 Optical Switches

An *optical switch* is the heart of an optical network. The objective in using optical switches rather than semiconductor switches is to increase the speed and volume of traffic switching in a core node of a computer communication network. The optical switch acts as an *optical cross-connect* (OXC) and can accept various wavelengths on network input ports and route them to appropriate output ports. The optical switch performs three main functions:

1. Routes all wavelengths of an incoming fiber to a different outgoing fiber
2. Switches specific wavelengths from an incoming fiber to multiple outgoing fibers
3. Takes incoming wavelengths and converts them to another wavelength on the outgoing port

Figure 14.2 (c) shows a simple 2×2 optical switch. A signal with wavelength i arriving at input j of the switch denoted by $\lambda_{i,j}$ can be switched on any of the four output ports. In this figure, $\lambda_{1,1}$ and $\lambda_{2,1}$ arrive on the first input port of the switch; $\lambda_{1,1}$ can be switched on output port 2, and $\lambda_{2,1}$ can be forwarded on output port 1. This basic optical switch is a switch element in larger-scale switch architectures. The basic switch can be made using various technologies. Such switch elements are broadly classified as either *non-electro-optical* or *electro-optical*.

Classification of Switch Elements

Non-electro-optical switches have simple structures. For example, a *mechanical optical switch* uses mirrors at a switch's input and output ports. A switch can be controlled by moving the mirrors and directing a light beam to a desired direction, and eventually to the desired output port. The advantages of mechanical switches are low insertion, low cross talk, and low cost. However, they have low speed. Another example is the *thermo-optic switch*, which is built on a waveguide. In this type of switch, a change of temperature in the thermo-optic structure of the waveguide can change the refractive index of the material, thereby allowing a switching function to be formed. Although similar to mechanical switches, thermo-optical switches operate at low speeds, but their cross talk is lower.

Figure 14.3 shows a typical 2×2 *electro-optic switch*, which uses a *directional coupler*. A 2×2 directional coupler is an integrated waveguide that can combine or split signals at its output ports. A coupler is the building block of filters, multiplexers, and switches and is known as a *star coupler*. A star coupler operates by changing the refractive index of the material placed in the coupling region. The switching function can be achieved by applying an appropriate voltage across the two electrodes.

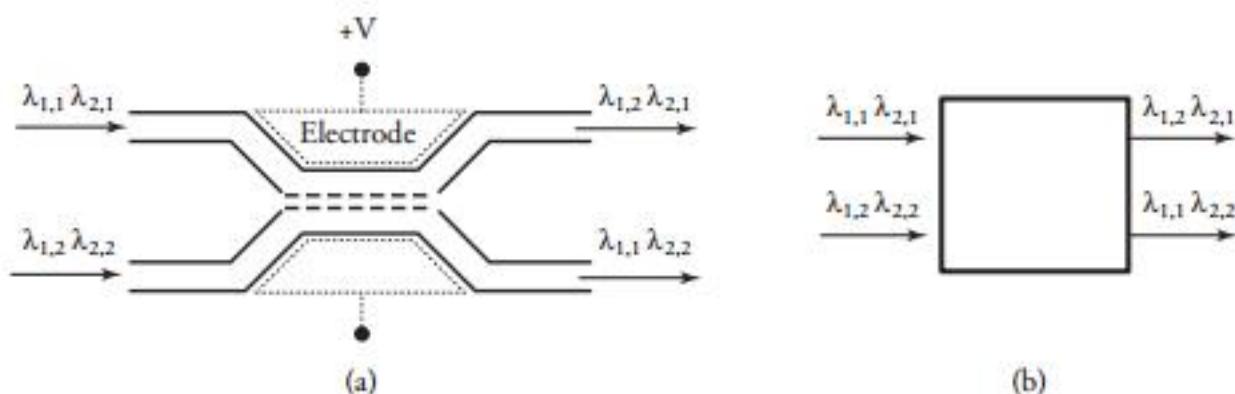


Figure 14.3 Smallest optical switch constructed with directional coupler: (a) architecture; (b) symbol

The switching function is achieved by taking a fraction, Φ , of the power from input port 1 and placing it on output port 1, with the remaining fraction $1 - \Phi$ of the power on output port 2. Similarly, a fraction $1 - \Phi$ of the power from input port 2 is placed on output port 1, with the remaining power fraction Φ on output port 2. The advantages of this switch are its speed and a modest level of integration to larger modules. However, the high cost of its production is a disadvantage of this switch over the other switch types.

Contention Resolution

Designing large-scale switching devices is one of the main challenges in optical networks. Ideally all the functions inside an optical node should be performed in the optical domain. However, packet contention in switches can be processed only in electronic devices. Several technological factors bring restrictions to the design of an optical switch.

The expansion of a fixed-size switch to higher scales is also a noticeable challenge. Electronic switches have much better flexibility of integration than do optical devices. Some performance factors other than switching speed should be considered in order to prove the suitability of a switch for optical networks. As do regular switches, optical switches face the challenge of *contention resolution* within their structures. Contention resolution is of the following three types:

1. *Optical buffering*. Although buffering is a challenge in an optical environment, optical buffers are implemented using fixed-length delay fibers.
2. *Deflection routing*. If two or more packets need to use the same output link, only one is routed along the desired output link; the others are deflected onto undesired paths directed to the destination through a longer route but with higher priorities.
3. *Wavelength conversion*. By changing wavelengths, signals can be shuffled and forwarded onto other channels.

Another important factor in optical switches is *insertion loss*, which is the fraction of power lost in the forward direction. This factor should be kept as small as possible in optical switches. Sometimes, the dynamic range of the signals must be increased in switches just because the level of loss may not be acceptable for the corresponding particular connection.

The factor of *cross talk* in switches must be minimized. Cross talk is the ratio of output power from a desired input to the output power from all other inputs. At issue is the amount of energy passing through from adjacent channels. This energy can corrupt system performance. The cost of optical devices, especially WDMs, is high compared to that of electronic modules. The production of all-fiber devices can also be a way to reduce the cost.

14.3 Large-Scale Optical Switches

Large-scale optical switches can be achieved either by implementing large-scale star couplers or by cascading 2×2 or even 1×2 multiplexers. However, the implementation of star couplers larger than 2×2 is expensive, owing to difficulties in the manufacturing process. Cascading small switches is a practical method to expand a switch. This method of switch expansion can make a desired-size switch quickly and at lower expense. However, some factors affect the overall performance and cost of an integrated switch, as follows:

- *Path loss.* Large-scale switches have different combinations of switch elements; therefore, a signal may experience different amounts of losses on different paths. Hence, the number of switch elements cascaded on a certain path might affect the overall performance of the switch.
- *Number of crossovers.* Large optical switches can be manufactured on a single substrate by integrating multiple switch elements. In an integrated optical system, a connection between two switch elements is achieved by one layer of a waveguide. When paths of two waveguides cross each other, the level of cross talk increases on both paths. As a result, the number of crossovers in an optical switch must be minimized.
- *Blocking.* As discussed in Section 13.1, a switch is *wide-sense nonblocking* if any input port can be connected to any unused output port without requiring a path to be rerouted; is *rearrangably nonblocking* if, for connecting any input port to an unused output port, a rearrangement of other paths is required.

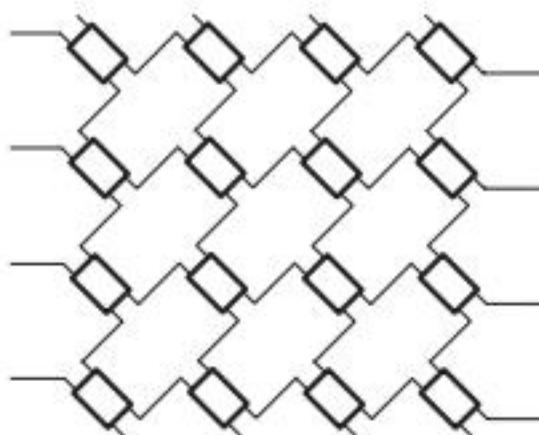


Figure 14.4 A 4×4 wide-sense nonblocking optical crossbar without waveguide crossovers, using basic 2×2 switch elements

There are several proposed topologies for large-scale switching networks. Two of the more practical ones are *crossbar* and the *Spanke-Beneš network* architectures.

14.3.1 Crossbar Switching Network

The architecture of an optical *crossbar* is not quite the same as the one in Chapter 13, as indicated in Figure 14.4. An optical crossbar is wide-sense nonblocking. The fundamental difference in the architecture of the optical crossbar and traditional crossbars is the existence of different and variable-length paths from any input to any output. This feature was created in the structure of the optical crossbar to minimize the crossover of interconnections. An optical crossbar is made up by 2×2 switches. Denoting the number of switches a signal faces to reach an output port from an input port by ℓ , the bounds for ℓ can be investigated from the figure as

$$n \leq \ell \leq 2n - 1, \quad (14.1)$$

where n is the number of inputs or outputs of the crossbar. In an $n \times n$ crossbar, the number of required 2×2 switches is n^2 . The 16 2×2 switches in the Figure 14.4 crossbar are structured in an appropriate way. The main issue with the crossbar structure is its cost, which is a function of n^2 . But a great advantage of such an architecture is that the switch can be manufactured without any crossovers.

14.3.2 Spanke-Beneš Switching Network

Figure 14.5 shows a rearrangably nonblocking switch called a *Spanke-Beneš switching network*. This network is designed with identical 2×2 switch elements, and there are

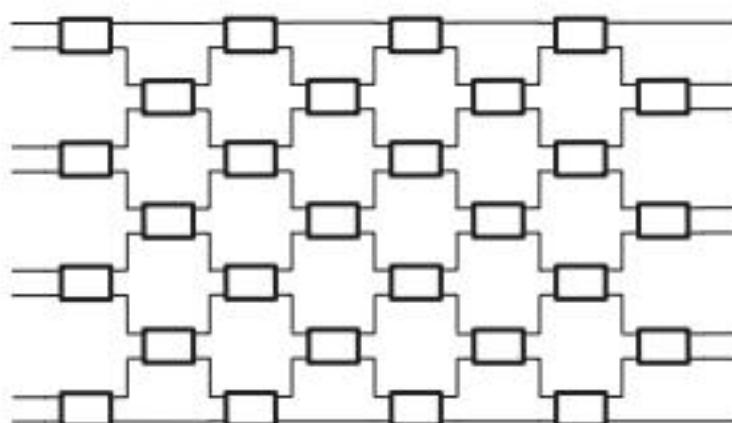


Figure 14.5 An 8×8 Spanke-Beneš switching network with rearrangeably nonblocking structure without requiring any interconnection crossover

no crossover paths. This switching network is also known as n -stage planar architecture. Denoting the number of switches a signal faces to reach an output port from an input port by ℓ , we can derive the bounds for ℓ by

$$\frac{n}{2} \leq \ell \leq n. \quad (14.2)$$

This inequity can be verified from Figure 14.5, which shows multiple paths between each input/output pair. Thus, the longest possible path in the network is n , and the shortest possible path is $n/2$. This arrangement is made to achieve a moderately low blocking. Thus, there are n stages (columns) and $\frac{n(n-1)}{2}$ switch elements in an $n \times n$ switching network.

14.4 Optical Routers

An *optical router*, known as the *wavelength router*, or simply a *network node*, is a device that directs an input wavelength to a specified output port. Each router in a network is thus considered a node. Optical nodes are classified as either *broadcast nodes* or *wavelength routing nodes*.

In a broadcast node, a wavelength is broadcast by a passive device to all nodes. The passive device can be an *optical star coupler*. A coupler combines all incoming signals and delivers a fraction of the power from each signal on to each output port. A *tunable optical filter* can then select the desired wavelength for reception. Broadcast nodes are simple and suitable for use in access networks, mainly LANs. The number of optical links connected to a broadcast node is limited, since the wavelengths cannot be reused in the network.

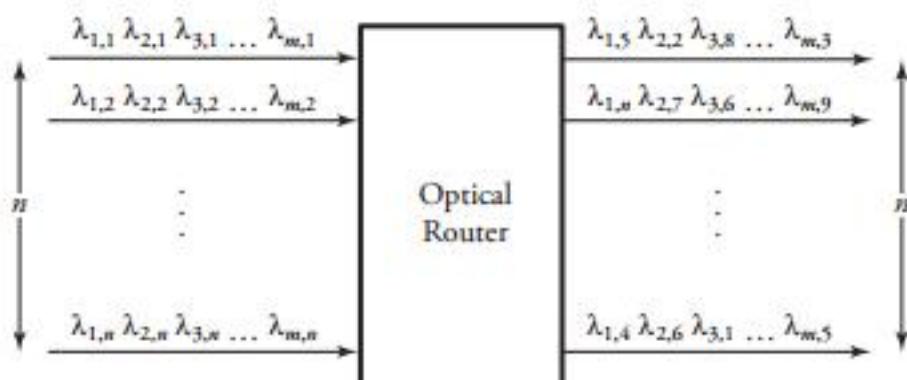


Figure 14.6 Overview of a wavelength routing node

Wavelength routing nodes are more practical. Figure 14.6 shows an overview of an $n \times n$ wavelength routing node in which full connectivity is ensured between n inputs and n outputs, using m wavelengths. In general, a signal with wavelength i arriving at input j of the router denoted by $\lambda_{i,j}$ can be switched on any of the n output ports. For example, a light with wavelength $\lambda_{3,2}$ among all wavelengths $\{\lambda_{1,2}, \lambda_{2,2}, \lambda_{3,2}, \dots, \lambda_{m,2}\}$ arrives on the second input port of the router. This wavelength can be switched on any of the outputs and, possibly, with a different wavelength, as $\lambda_{4,7}$.

14.4.1 Structure of Wavelength Routing Nodes

Wavelength routers are of two types. The first type can switch an input only to an output using the same wavelength. Figure 14.7 gives an overview of this router. Consider this router with n inputs and n outputs, where each input i can bring up to m wavelengths as $\{\lambda_{1,i}, \lambda_{2,i}, \lambda_{3,i}, \dots, \lambda_{m,i}\}$. In this case, any wavelength $\lambda_{k,i}$ can be switched on output j with wavelength $\lambda_{k,j}$. This way, in each round of switching, all wavelengths at the input ports with the same wavelengths can be switched to desired output ports taking place in the same corresponding channels. As shown in Figure 14.7, the router consists of n input WDMs, m optical switches of size $n \times n$, and n output WDMs.

The second type of wavelength router requires *wavelength-conversion* capabilities, as illustrated in Figure 14.8. Consider this router with n inputs and n outputs with up to m wavelengths per input. With this router, a wavelength $\lambda_{k,i}$ at input i can be switched on output port j ; in addition, the k th wavelength can be converted to the g th, to be shown by $\lambda_{g,j}$. This way, in each round of switching, any wavelength at the input port with the same wavelengths can be switched on any output port, using any

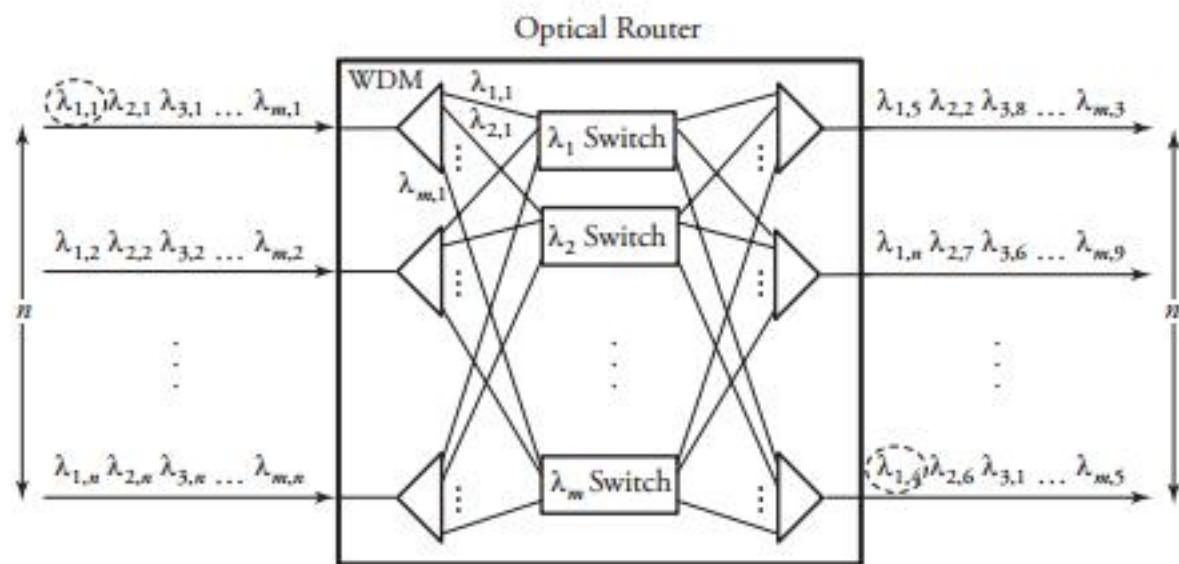


Figure 14.7 An all-optical router that replaces wavelength channels

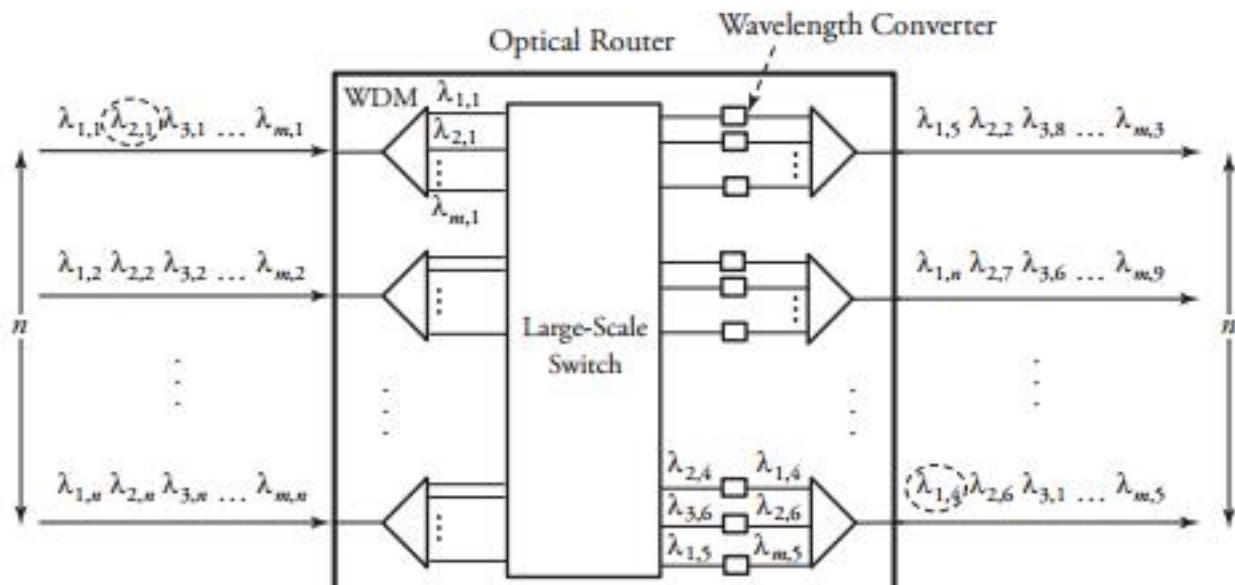


Figure 14.8 An all-optical router, which can replace wavelength channels and convert wavelengths

available wavelength. In Figure 14.8, $\lambda_{2,1}$ from input 1 is switched to $\lambda_{1,4}$ on the n th output.

In general, wavelengths may be required to be converted in a network for the following reasons:

- The wavelength of a signal entering the network may not be appropriate at a node.
- A node may realize the availability of different and better wavelengths on the network links.

Depending on whether a wavelength converter converts a fixed- or variable-input wavelength to a fixed- or variable-output wavelength, there are different categories for wavelength converters, such as fixed-input fixed-output, fixed-input variable-output, variable-input fixed-output, and variable-input variable-output.

Semi-Optical Routers

Depending on the type of a switch fabric used in the structure of an optical router, we can further classify optical routers into two main groups:

1. Routers with *all-optical switches*, in which signals traverse the network entirely in the optical domain.
2. Routers with *optically opaque optical switches*, in which some forms of optoelectronic processing take place within multiplexing elements.

All-optical, or *photonic*, switches, are the ones we have studied so far. In such switches, the optical signal need not be converted to electrical, and switching is handled in the optical domain. All-optical domains have an advantage of large-capacity switching capabilities. In the optically opaque, or so-called semioptical switches, an optical signal is converted to electrical, and then the appropriate switching takes place. It is obviously a case with an electrical switch core followed up with optical interfaces.

In an all-optical switch, switching and wavelength conversion take place entirely in the optical domain. An optically opaque switch uses an optoelectronic conversion mechanism. This architecture requires other interfaces, such as optical-to-electrical converters or electrical-to-optical converters. The electronic segments are used around the optical switch matrix.

14.5 Wavelength Allocation in Networks

Similar to nonoptical networks, optical networks consist of a number of routing nodes connected by communication links. In optical networks, nodes may have optical multiplexing, switching, and routing components, and links are optical fibers. In the early generations of optical fibers, transmission of signals was degraded on short distances mainly because of lack of amplification. In later versions of optical-fiber systems, the loss of signals was significantly reduced through carefully designed amplifiers. In the latest generations of optical communication systems, the overall cost has been lowered, the effect of dispersion has been eliminated, the data bit rate has been enhanced up to beyond tens of gigabits per second, optical amplifiers have replaced regenerators, and WDM technology has been used for multiplexing purposes.

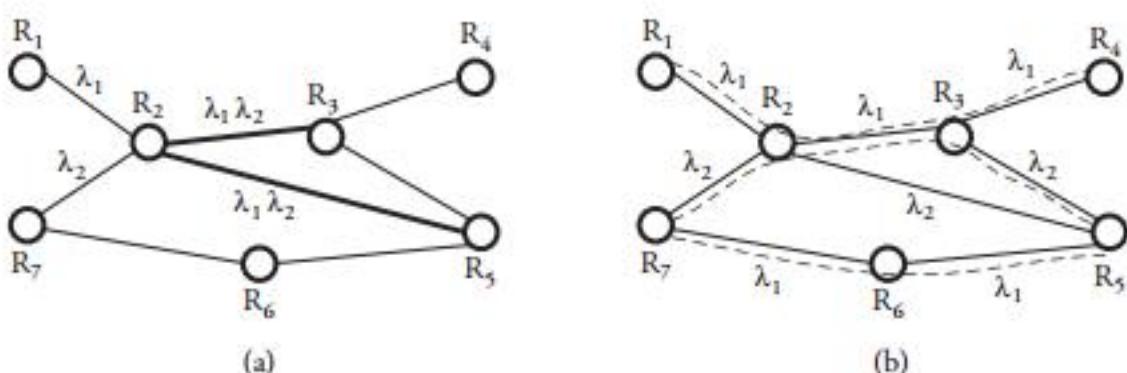


Figure 14.9 An optical network with (a) broadcast nodes and (b) wavelength routing nodes

All-optical networks can be designed to carry data using any wavelength regardless of the protocol and framing structure. All-optical networks were encouraged by the development of ultralong-haul fibers mounted on a single physical fiber. By transmitting each signal at a different frequency, network providers could send many signals on one fiber, just as though each signal were traveling on its own fiber. All-optical networks handle the intermediate data at the optical level. This efficient functioning of the network provides a number of advantages in handling data: reduced overall cost and increased system bandwidth.

A lightpath carries not only the direct traffic between the nodes it interconnects but also traffic from nodes upstream of the source to downstream of the destination. Thus, a lightpath can reduce the number of allocated wavelengths and improve the network throughput. In practice, a large number of lightpaths may be set up on the network in order to embed a virtual topology.

14.5.1 Classification of Optical Networks

Optical networks can be classified according to the type of nodes being used in the network. Figure 14.9 (a) shows an optical network of broadcast nodes. A broadcast node combines all incoming signals and delivers a fraction of the power from each signal on to each output port. A *tunable optical filter* can then select the desired wavelength for reception.

Figure 14.9 (b) illustrates a network that uses wavelength routing nodes. These types of nodes are capable of reusing wavelengths and handling many simultaneous lightpaths with the same wavelength in the network. Two lightpaths—R₁-R₂-R₃-R₄ and R₇-R₆-R₅—do not use any shared link and can therefore be assigned the same

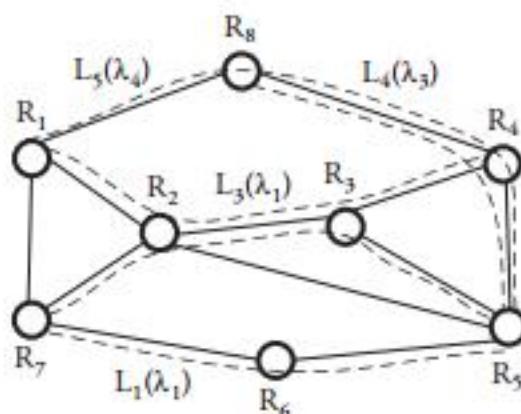


Figure 14.10 Allocation of wavelengths in an optical network

wavelength λ_1 . Because lightpaths $R_1-R_2-R_3-R_4$ and $R_7-R_2-R_3-R_5$ share part of their common path (R_2-R_3), they must therefore use a different wavelength.

14.5.2 Wavelength Allocation

Consider an all-optical network in which each link in the network can carry a maximum of λ_n wavelengths. Because of the maximum wavelength capacity, a network may not be able to handle all lightpath requests, so some requests may be blocked. To keep lightpaths separated on the same link, they should be allocated different wavelengths. For example, consider the all-optical network shown in Figure 14.10, with five lightpaths: L_1 through L_5 . On link R_2-R_3 , the same wavelength cannot be allocated.

A lightpath with wavelength λ_i at any node's input can be converted to any available wavelength $\lambda_j \in \{\lambda_1, \dots, \lambda_n\}$ on the node's output link. If no wavelength is available, a lightpath uses the same wavelength on all links of its path. Based on our earlier discussion, wavelength allocations for this case can be arranged as follows: L_1 is assigned λ_1 , L_2 is assigned λ_2 , L_3 is assigned λ_1 , L_4 is assigned λ_3 , and L_5 is assigned λ_4 .

The concept of wavelength allocation can be analyzed in two ways. One way is to assume that the probability of a wavelength being used on a link is independent of the use of the same wavelength on all other links of the lightpath. Although this method is not practical, the analysis provides a quick approximation on how effective the assignment of wavelengths is. The second method removes the assumption of independence.

Wavelength Allocation Without Dependency

The wavelength-allocation algorithm assigns an arbitrary but identical wavelength on every link of a lightpath when one such wavelength is free on every piece of its path.

In this section, we assume that the probability of a wavelength being used on a link is independent of the use of the same wavelength on all other links of the lightpath. Consider a single link in which λ_n wavelengths are available. For each lightpath request, the first available wavelength is assigned. The wavelength-request arrival is assumed to be Poisson with the rate that leads to a utilization ρ . Then, the blocking probability on this link follows the Erlang-B formula expressed by Equation (11.46):

$$P(\lambda_n) = \frac{\rho^{\lambda_n}}{\lambda_n! \left(\sum_{i=0}^{\lambda_n} \frac{\rho^i}{i!} \right)}. \quad (14.3)$$

This formula calculates the probability that an arriving request finds no available wavelength while there is no waiting line for request. For gaining the highest efficiency on assigning wavelengths to requests, wavelengths must be reused effectively. If lightpaths overlap, the *wavelength-conversion gain* would be low.

Not only the overlap between lightpaths impacts the wavelength-conversion gain; so too does the number of nodes. Assume that for each lightpath, the route through the network is specified. Let the probability that a wavelength is used on a link be p . If the network has λ_n wavelengths on every link and a lightpath request chooses a route with r links, the probability that a given wavelength is not free on at least one of existing r links of the route can be derived by

$$P_b = 1 - (1 - p^r)^{\lambda_n}. \quad (14.4)$$

Note that for P_b , we use the rules developed in Section 7.6.4. The probability that a given wavelength is free on any given link is $(1 - p)$. Consequently, the probability that a wavelength is not free on all the r links of the path is $1 - (1 - p)^{\lambda_n}$. Using the parallel rule explained in Section 7.6.4, P_b expresses the probability that a given wavelength is not available on at least one of existing r links of the route. Similarly, the probability that a lightpath request is blocked is

$$P_r = 1 - (1 - p^{\lambda_n})^r. \quad (14.5)$$

The wavelength allocation with dependency is much more complicated. Next, we try to present a simplified analysis for this case.

Wavelength Allocation with Dependency

In practice, the allocation of a free wavelength to a lightpath on its every link is dependent on the use of other wavelengths on the same link. Let $P(\ell_i | \hat{\ell}_{i-1})$ be the probability that a wavelength is used on link i , given that the wavelength is not used

on link $i - 1$. Also, let $P(\ell_i | \ell_{i-1})$ be the probability that a wavelength is used on link i , given that the wavelength is used on link $i - 1$. Then:

$$P(\ell_i | \ell_{i-1}) = P(\ell_i | \hat{\ell}_{i-1}). \quad (14.6)$$

If we substitute $P(\ell_i | \hat{\ell}_{i-1})$ for p in Equation (14.4), P_b can be reexpressed for the dependency condition as

$$P_b = 1 - \left(1 - P(\ell_i | \hat{\ell}_{i-1})^r\right)^{\lambda_n}. \quad (14.7)$$

We can now estimate the packet delay over a lightpath on link i, j from node i to node j . Assume that packet-transmission times are exponentially distributed with mean time $1/\mu$, and Poisson packet arrival distribution with mean arrival rate Λ . Let $s_{i,j}$ be the loading value on link i, j , or the number of source/destination pairs whose traffic is routed across link i, j . Then, we can obtain the average delay on link i, j by using Equation (11.21):

$$E[T] = \frac{1}{\mu - s_{i,j}\Lambda}. \quad (14.8)$$

If a network has a total of n nodes, the average queueing delay incurred for a packet through all nodes over all source/destination pairs is expressed by

$$E[T_n] = \frac{1}{n(n-1)} \sum_{i,j} \frac{s_{ij}}{\mu - s_{i,j}\Lambda}. \quad (14.9)$$

14.6 Case Study: An All-Optical Switch

As a case study on optical switching networks, we consider the *spherical switching network* (SSN), a switch fabric first introduced by the author of this book in the *Journal of Computer Networks* (issue 40, 2002). The spherical network is a regular mesh network that can be realized as a collection of horizontal, vertical, and diagonal rings, as shown in Figure 14.11. Rings appear in bidirectional pairs that can form cyclical entities and create full regularity in the structure of the network. The spherical network uses a simple self-routing scheme. Furthermore, the availability of a large number of interconnections among switch elements and the special arrangement of interconnections potentially reduce the number of deflections compared to the existing deflection-routing networks.

The network is constructed with *fixed-size* switch elements, regardless of the size of the network. Each switch element consists of a 9×9 crossbar and a local controller. Although not indicated in the figure, an incoming link and an outgoing link connect the

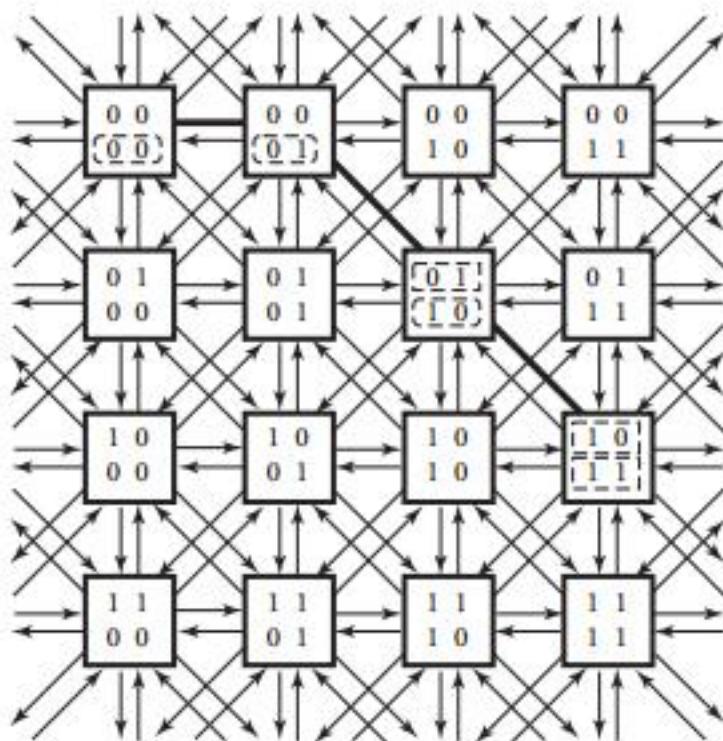


Figure 14.11 A 16-port proposed spherical switching network. Each square represents a 9×9 switch element with eight pairs of internal links (shown) and one pair of external links (not shown). An example for self-routing is from switch element 0000 to 1011.

network at each switch element to external links and then to local processors, which deliver and receive data sent through the network. One of the nine pairs of links is external and carries the external traffic; the other eight pairs are internal.

The spherical network doesn't use any buffering within the network, and thus this network gains from the lower complexity when compared later with other switching networks. Contention resolution in each switch element is based on the deflection of losing packets on undesired internal links and with increments in their priority fields. With the *torus*-shaped topology embedded in the network, when more than one packet requests the same outgoing link at each switch element, only one of them is forwarded on the preferred link; the others are deflected onto other links. By maintaining this rule in the system, once a packet is admitted to a network, it is not discarded when congestion occurs. Instead, the packet receives an increment in its priority field and is misrouted temporarily but will reach its destination. One main advantage of this network over many others is the existence of possible multiple equally desired outputs at each switch element. In other words, it might be possible at each point in the network to find *more than one shortest path* to a destination.

14.6.1 Self-Routing in SSN

The routing in the spherical network is fairly simple and is self-routing. Any switch element is given an index number. For example, the indices in a 16-port network are [00 – 00] through [11 – 11], as in Figure 14.11. Depending on whether the flowing traffic is directed toward decreasing index or increasing index, the self-routing is performed by decremental or incremental addressing, respectively. Clearly, the network has four routing cases:

1. Straight horizontal
2. Straight vertical
3. Straight diagonal
4. A combination of diagonal/horizontal routing or diagonal/vertical

In each case, there are a certain number of *shortest paths* through which a packet is channeled to its destination by taking a corresponding direction. In the first case, depending on whether the index number of the source is greater or smaller than the index number of the destination, the value of the second two index bits of the source address is decreased or increased hop by hop, respectively, until the address of a source and a destination becomes identical where the routing is completed.

In the second case, the process is identical to the horizontal case but in a vertical direction, and index decrement or increment is done on the value of the first two index bits of the source address. In case 3, the routing process is accomplished when the values of both the first and second two index bits of the source address simultaneously receive increment or decrement, as described in cases 1 and 2. Case 4 can be a combination of either the first and third cases or the second and third cases. In case 4, there might be more than one preferred path.

The preceding routing rules use the preferred directions to route each packet to its destination along a shortest path. Contention resolution in each switch element is based on deflection of a losing packet onto an undesired output if all preferred outputs are not available and giving an increment to the priority field of the packet.

14.6.2 Transmission in SSN

SSN can be used to construct an all-optical switch core. Suppose that any optical signal is assigned a unique wavelength. To use the fiber bandwidth efficiently, it is divided into 72 nonoverlapping optical wavelengths corresponding to 8 + 1 pairs of links at

each switch element. Each node is connected into eight other nodes through eight pairs of internal optical links plus a pair of external links. There are nine groups of wavelengths: $\Lambda(1)$, $\Lambda(2)$, $\Lambda(3)$, $\Lambda(4)$, $\Lambda(5)$, $\Lambda(6)$, $\Lambda(7)$, $\Lambda(8)$, and $\Lambda(9)$. Each group of wavelengths includes eight nonoverlapping optical wavelengths:

$$\begin{aligned}\Lambda(1) &= (\lambda_{1,1}, \lambda_{1,2}, \lambda_{1,3}, \lambda_{1,4}, \lambda_{1,5}, \lambda_{1,6}, \lambda_{1,7}, \lambda_{1,8}) \\ \Lambda(2) &= (\lambda_{2,1}, \lambda_{2,2}, \lambda_{2,3}, \lambda_{2,4}, \lambda_{2,5}, \lambda_{2,6}, \lambda_{2,7}, \lambda_{2,8}) \\ \Lambda(3) &= (\lambda_{3,1}, \lambda_{3,2}, \lambda_{3,3}, \lambda_{3,4}, \lambda_{3,5}, \lambda_{3,6}, \lambda_{3,7}, \lambda_{3,8}) \\ \Lambda(4) &= (\lambda_{4,1}, \lambda_{4,2}, \lambda_{4,3}, \lambda_{4,4}, \lambda_{4,5}, \lambda_{4,6}, \lambda_{4,7}, \lambda_{4,8}) \\ \Lambda(5) &= (\lambda_{5,1}, \lambda_{5,2}, \lambda_{5,3}, \lambda_{5,4}, \lambda_{5,5}, \lambda_{5,6}, \lambda_{5,7}, \lambda_{5,8}) \\ \Lambda(6) &= (\lambda_{6,1}, \lambda_{6,2}, \lambda_{6,3}, \lambda_{6,4}, \lambda_{6,5}, \lambda_{6,6}, \lambda_{6,7}, \lambda_{6,8}) \\ \Lambda(7) &= (\lambda_{7,1}, \lambda_{7,2}, \lambda_{7,3}, \lambda_{7,4}, \lambda_{7,5}, \lambda_{7,6}, \lambda_{7,7}, \lambda_{7,8}) \\ \Lambda(8) &= (\lambda_{8,1}, \lambda_{8,2}, \lambda_{8,3}, \lambda_{8,4}, \lambda_{8,5}, \lambda_{8,6}, \lambda_{8,7}, \lambda_{8,8}) \\ \Lambda(9) &= (\lambda_{9,1}, \lambda_{9,2}, \lambda_{9,3}, \lambda_{9,4}, \lambda_{9,5}, \lambda_{9,6}, \lambda_{9,7}, \lambda_{9,8})\end{aligned}$$

Wavelength groups $\Lambda(1)$ through $\Lambda(8)$ are assigned to the eight pairs of internal links, and $\Lambda(9)$ is assigned to the external links. At each incoming fiber link, there are at most eight packets multiplexed into one fiber. All packets are destined to different addresses. When packets arrive at a node, they are demultiplexed, and the node makes the routing decision for each packet. After a next hop for each incoming packet is decided, the wavelength of that packet is converted to one of the available wavelengths by wavelength converter and is then transmitted to the next switch element.

14.7 Summary

This chapter focused on optical communication networks, beginning with basic definitions and a review of such optical devices as *optical filters*, *wavelength-division multiplexers*, *optical switches*, and *optical buffers* and *optical delay lines*.

Optical switches can be classified as *non-electro-optical* switches using non-electro-optical devices such as *mechanical optical switches* or *thermo-optic switches*, and *electro-optic* switches using *directional couplers*. The cost of large-scale optical switches constructed by a one-segment optical switch is high. Therefore, several topologies of large-scale switching networks use simple switch elements, such as *crossbar* and *Spanke-Beneš network* architectures.

Optical networks constructed with optical devices provide routing, grooming, and restoration of data at the wavelength level. Two popular models for managing the net-

work (IP) layer and the optical layer are the *overlay model* and the *peer model*. In the overlay model, the optical layer and IP layer each have their own independent control planes.

Wavelength reuse and allocation in optical networks is a key topic. Because of the maximum capacity on the number of wavelengths that can be carried on a link, a network may not be able to handle all lightpath requests; consequently, some requests may be blocked. To keep lightpaths separated on a same link, they should be allocated different wavelengths. Wavelength allocation can be analyzed in two ways. One way is to assume that the probability of a wavelength being used on a link is independent of the use of the same wavelength on all other links of the lightpath. The other method does not make any assumption of independence.

An all-optical switching network called *spherical switching network* (SSN) was presented as a case study. Routing in that network is fairly simple and self-routing. Any switch element is given an index number. Depending on whether the flowing traffic is directed toward decreasing index or toward increasing index, the self-routing is performed by decremental or incremental addressing respectively.

The next chapter discusses multicast protocols and algorithms. In addition, switching techniques at the node level are used to show how copies of packets are made in switch fabrics.

14.8 Exercises

1. Consider a crossbar switching network similar to the one shown in Figure 14.4 but of size 8×8 . Suppose that the crosstalk suppression of each 2×2 switch element is 40 dB.
 - (a) Calculate the maximum and minimum overall cross-talk suppressions for the crossbar switch.
 - (b) Calculate the average of overall cross-talk suppressions, using all possible existing paths.
2. Consider the Spanke-Beneš network shown in Figure 14.5. Suppose that the cross-talk suppression of each 2×2 switch element is 40 dB. Calculate the overall cross-talk suppressions, using all possible existing paths.
3. To design an 8×8 optical router, we are comparing three different structures, each using, respectively, a crossbar switching network, a Spanke-Beneš network, and an 8×8 directional coupler.
 - (a) Which structure offers the best overall cross-talk suppression?
 - (b) Which structure has the lowest overall average switching delay?

4. Consider a ring optical backbone network consisting of eight 2×2 switching nodes labeled 1 through 8. Nodes are interconnected with two fiber-optic rings carrying traffic in opposite directions. Three wavelengths, λ_1 , λ_2 , and λ_3 , are available on each link.
 - (a) A SONET network consisting of duplex lightpaths 2-4, 4-7, and 3-6 is constructed over this backbone. For convenience, assume that both halves of the duplex SONET network are identical in terms of wavelengths. Find the minimum number of wavelengths to construct the SONET network.
 - (b) Now, assume that another SONET network, consisting of duplex lightpaths 2-5, 5-6, and 2-8, is constructed over this backbone. Using the same assumptions as in part (a), find the minimum number of wavelengths to construct this SONET network.
 - (c) Both SONET networks are supposed to be constructed over this backbone network simultaneously. Find the minimum number of wavelengths to construct the two SONET networks.
5. Suppose that four wavelengths exist on each single link of a three-link path. For each lightpath request, the first available wavelength is assigned. The wavelength request arrival is assumed to be Poisson, with the rate that leads to 80 percent utilization. For each lightpath, the probability that a wavelength is used on a link is 20 percent. Assume that a lightpath request chooses a route with two links.
 - (a) Find the blocking probability on this link.
 - (b) Find the probability that a given wavelength is not free on at least one of existing two links of a route.
 - (c) Find the probability that a lightpath request is blocked.
6. Consider an optical network with n nodes. Let $L_{i,j}$ be an arriving Poisson traffic rate on link i, j in packets per second, and let $1/\mu_{i,j}$ be the mean time of exponentially distributed packet transmission on link i, j .
 - (a) Find the average queueing delay on link i, j .
 - (b) Let $s_{i,j}$ be the number of source/destination pairs whose traffic is routed over link i, j . In this case, find the average queueing delay on link i, j .
 - (c) Find the average queueing delay for a packet, given all source/destination pairs.

CHAPTER 15

Multicasting Techniques and Protocols

The capability of *multicasting* traffic has become a fundamental factor in the design of computer communications. Multicasting is the transmission of data from one source to a group of destinations. Data networks must be able to support such multimedia applications by multicasting data, voice, and video. Multicasting has led to the development of such applications as teleconferencing, multipoint data dissemination, educational distance learning, and Internet TV. In such applications, audio and video streaming technologies are needed to implement multicasting, owing to the real-time component of multimedia applications. In this chapter, we focus on several advanced multicast protocols used in the Internet. The following topics are covered:

- *Basic definitions and techniques*
- *Intradomain multicast protocols*
- *Interdomain multicast protocols*
- *Node-level multicast algorithms*

The chapter begins with some basic definitions and techniques: multicast group, multicast addresses, and multicast tree algorithms. Next, tree algorithms must be explored, as they are the next set of foundations for understanding Internet packet multicasting. Two main classes of protocols are considered: *intradomain* multicast routing protocols, by which packets are multicast within a domain, and *interdomain* routing protocols,

by which packets multicast among domains. Techniques and algorithms used within router hardware are introduced at the end of the chapter.

15.1 Basic Definitions and Techniques

A simple operation of data multicasting can be viewed through a transmitting host and several receiving hosts. Instead of forcing the source host to send a separate packet to each destination host or user, the source needs to be able to forward a single packet to multiple addresses, and the network must be able to deliver a copy of that packet to each *group* of receiving hosts or users. Hosts can then choose to join or leave this group without synchronizing with other members.

A host may also belong to more than one group at a time. Figure 15.1 shows a typical packet multicasting in different layers. As shown, multicasting a packet could involve almost all layers of a communication network. A multicast task can be performed at the application layer, where two hosts can directly connect through a copy of the source. Meanwhile, a message can be multicast systematically through the physical, link, and network layers.

With the use of a robust multicasting protocol, a network reduces traffic load by simultaneously delivering a single stream of information to potentially thousands of recipients. An example is the distribution of real-time stock quotes and news. In such cases, the application source traffic is delivered to multiple receivers without burdening the source or the receivers and using a minimum amount of network bandwidth. Among the challenges of implementing data multicasting are the following two, which are especially relevant in large networks:

1. *Manageability*. As a data network becomes larger, constructing a central management system for distributed multicast tasks can become increasingly challenging.
2. *Scalability*. In practice, large networks that lack mechanisms for extending or adding equipment can experience substantial scalability problems, especially in constructing multicast trees.

Tight real-time components require a constant flow of data and have a very low jitter tolerance. To this end, corrupted data frames are often dropped rather than retransmitted. In this context, one would like to see a dynamic multicast system that adapts to deletions and additions of ongoing data over time. In the multicast service, we want to make copies of data at or near a source with only one access to the transport media. But having copies of packets floating around raises issues of

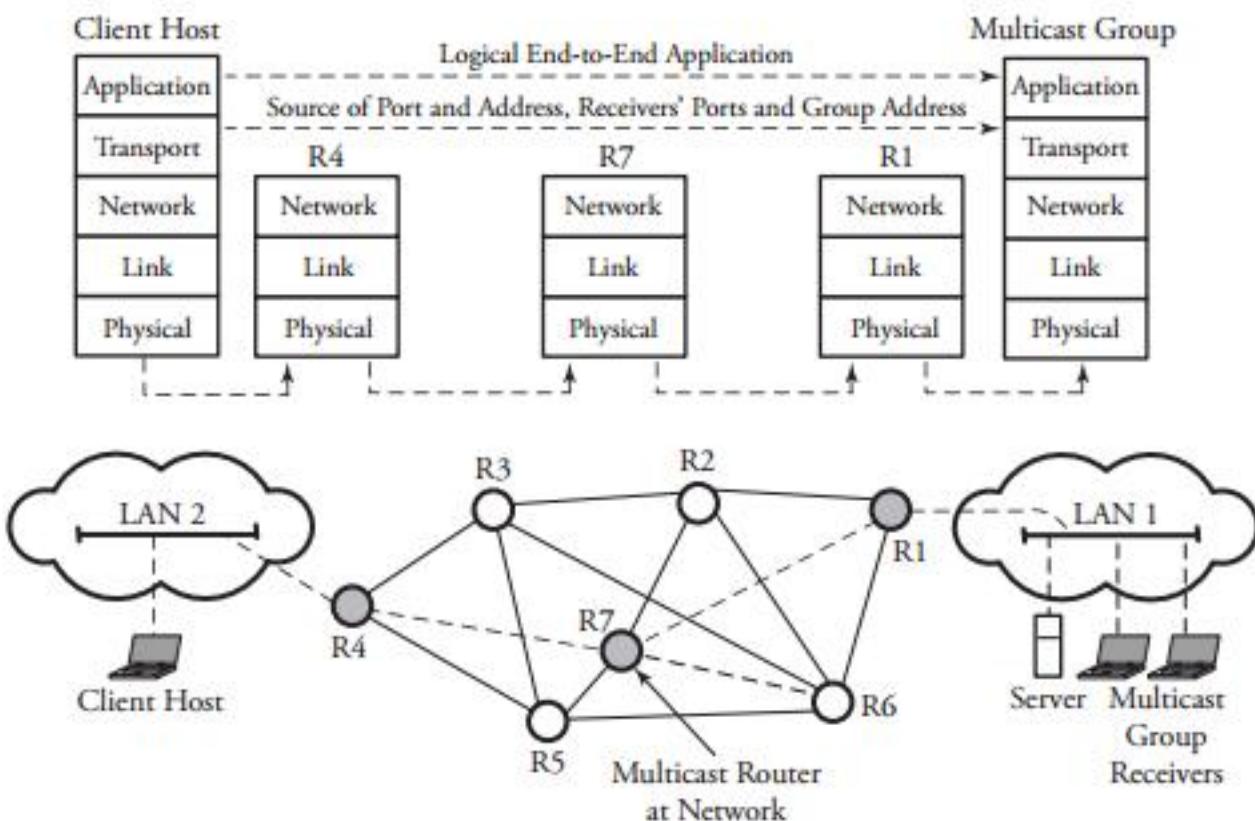


Figure 15.1 An IP multicast model

congestion. However, copying packets near destinations risks losing all copies of packets if the original packet is lost at that location.

15.1.1 IP Multicast Address

Chapter 2 discussed IP addressing, noting that multicast packets are of class D. The first four bits of a class D IP address identify the class of the address, where 1110 represents a multicast address. The remaining 28 bits identify a particular multicast group. Thus, multicast addresses can range from 224.0.0.1 through 239.255.255.255, and 2^{28} is the maximum number of multicast groups that can be formed simultaneously.

Multicast addressing is an open issue at both the network and the link layers. As shown in Figure 15.2, a multicast address from the network layer needs to be mapped to multicast addresses of the data-link layer when mapping between IP multicast and an Ethernet multicast address is performed. Note that the space for the Ethernet multicast address is smaller than that for an IP multicast address. Therefore, the first 23 bits of the IP multicast address are placed into the first 23 bits of the Ethernet multicast address field. The 25 bits left in the Ethernet multicast address are assigned a fixed value. Note

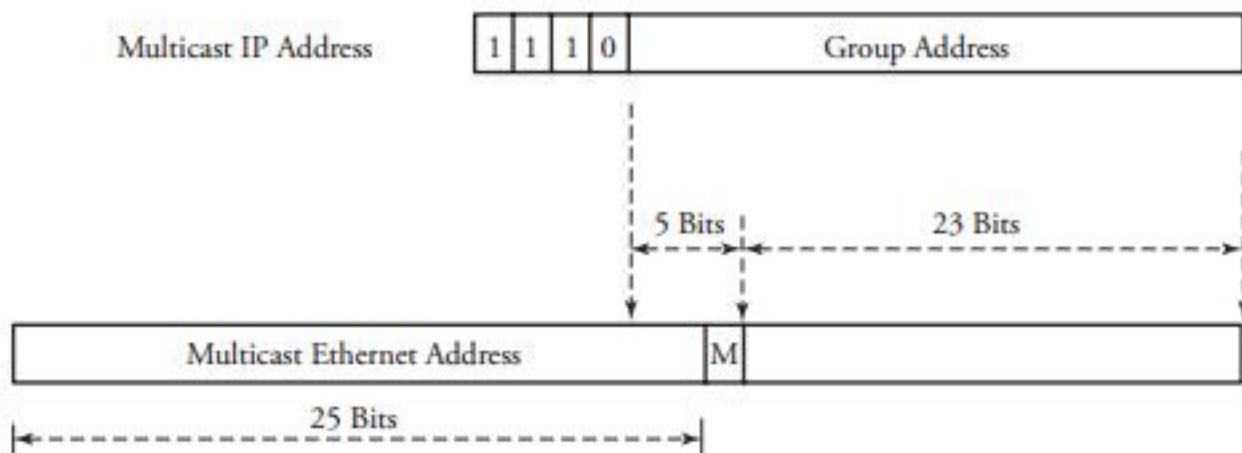


Figure 15.2 Mapping between IP multicast and Ethernet multicast addresses

also that a 1-bit field, M , represents the two cases: the Internet multicast with 0 and any other applications with 1.

15.1.2 Basic Multicast Tree Algorithms

Multicast group membership is a dynamic function, so any host can join or leave an IP multicast group at any time. Multicast packets are also subject to loss, delay, duplication, and out-of-order delivery. In WANs, data multicast transmission requires routers capable of managing multicast trees. A main difference between a regular unicast communication and a multicast communication is the nature of their routing. Unicast routing uses the shortest path, whereby two nodes communicate by means of a minimum-weight path.

The challenge in multicast routing is to identify the multicast group of nodes and to then find in the multicast group the minimum-weight tree that spans all members of the group. The multicast tree must also have low end-to-end delay, be scalable and survivable, and support dynamic membership. In a multicast transmission, a single copy of the packet is sent to a router with multicast capability, and the router makes copies and forwards them to all receivers. This is an efficient use of the bandwidth, as there is only one copy of a message over a link. The standard multicast model for IP networks expresses the way end systems send and receive multicast packets. This model is summarized as follows.

- A source typically uses the User Datagram Protocol (UDP), not TCP. As a result, multicast packets are delivered using a best-effort algorithm anytime, with no need to register or to schedule transmission.

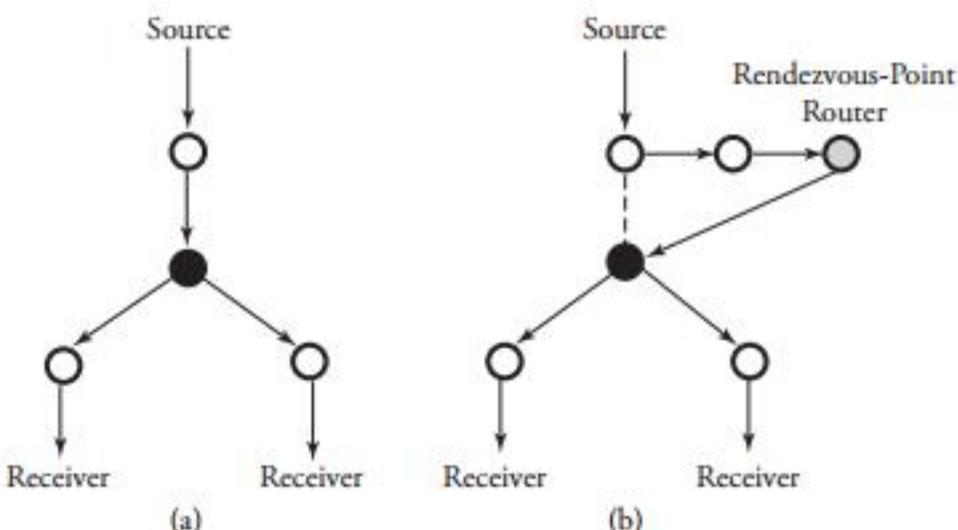


Figure 15.3 Two methods of constructing multicast algorithms: (a) dense mode, using a source-based tree; (b) sparse-mode, using a shared tree.

- Sources need to consider only the multicast address; they need not be a member of the multicast group to which they are sending.
- Multicast group members do not need to negotiate within the central group manager. Thus, they can join or leave a group at any time.

A *multicast tree algorithm* is the heart of multicast protocols and implements the fundamental task of packet copying and distribution in networks. Note that unicast routing uses the destination address to make its forwarding decision, whereas multicast routing normally uses the source address to make its forwarding decision. Tree algorithms form a triangle, or a tree, whereby the top vertex is represented by the source, and all recipients of the packet reside on the bottom line. During construction of a distribution tree, a *group* membership has to be formed, with copying nodes as members of the group. Updated information has to be maintained in a router if it is a member of the group. Two types of methods are used to construct a multicast tree: *dense-mode* trees and *sparse-mode* trees (see Figure 15.3).

Dense-Mode Algorithm

With *dense-mode*, or *broadcast-and-prune*, algorithm, a multicast tree technique called *source-based tree* is used. This technique requires the determination of a shortest-path tree to all destinations and uses a reverse shortest-path tree rooted at a source. The tree starts at the source; for every source, there is always a corresponding shortest-path tree. As a result, the spanning tree guarantees the lowest cost from a source to all leaves of

the tree. Packets are forwarded on the shortest-path tree according to: (1) the source address they originated from, and (2) the group address they are addressed to. The source-based tree in a network provides the shortest distance and the least delay from a source to all receivers.

Sparse-Mode Algorithm

The *sparse-mode algorithm* uses a *shared-tree* technique. Relatively short distances are selected, and no effort is made to find the best paths to destinations. First, some shared trees are first formed. These trees consider multiple senders and receivers and use some nodes acting as *rendezvous points* to connect sources and receivers. A rendezvous point—known as the *core*, or *root*—is considered a point in the network where roots of distribution subtrees are shared and the multicast data flows down to reach the receivers in the network. The sparse-mode algorithm uses a rendezvous router to coordinate forwarding packets from source to receivers and to prevent the initial flooding of datagrams. However, this may introduce extra delay from the source to all receivers but requires less hardware complexity for finding the shortest path. All routers in the network must discover the information about network routers that become rendezvous routers and the mappings of multicast groups to rendezvous routers.

The sparse mode has a couple of drawbacks. One is that a rendezvous router can be a hot-spot for multicast traffic and a point of failure in routing. Also, forwarding traffic to the rendezvous router and then to receivers causes delay and longer paths.

15.1.3 Classification of Multicast Protocols

The fundamental algorithms described in Section 15.1.2 are the basic procedures for constructing networking multicast protocols. These algorithms are used as foundations of protocols for multicasting packets in the Internet. Two main classes of multicast protocols are described in Sections 15.2 and 15.3, respectively: *intradomain* multicast routing protocols, by which packets are multicast within a domain, and *interdomain* routing protocol, by which packets are multicast among domains.

15.2 Intradomain Multicast Protocols

Intradomain routing protocols carry out the multicast function within domains. The implementation of multicast routing faces the following particular challenges:

- Dynamic change in the group membership
- Minimizing network load and avoiding routing loops
- Finding concentration points of traffic

In practice, several protocols play major roles in establishing multicast connections. The *Distance Vector Multicast Routing Protocol* (DVMRP) and the *Internet Group Management Protocol* (IGMP) are the two original protocols forming the early version of the *multicast backbone* (MBone). Other protocols, such as *Multicast Open Shortest Path First* (MOSPF), *core-based trees* (CBT), and *protocol-independent multicast* (PIM) enhance MBone performance.

15.2.1 Distance Vector Multicast Routing Protocol (DVMRP)

The Distance Vector Multicast Routing Protocol (DVMRP) is one of the oldest multicast protocols. It is based on a concept of exchanging routing table information among directly connected neighboring routers. The MBone topology can enable multiple tunnels to run over a common physical link. Each participating router maintains information about all the destinations within the system. DVMRP creates multicast trees, using the dense-mode algorithm. A multicast router typically implements several other independent routing protocols besides DVMRP for multicast routing, such as RIP or OSPF for unicast routing.

This protocol is not designed for WANs with a large number of nodes, since it can support only a limited number of hops for a packet. This is clearly considered a drawback of this protocol, as it causes packet discarding if a packet does not reach its destination within the maximum number of hops set. Another constraint in this protocol is the periodic expansion of a multicast tree, leading to periodic broadcasting of multicast data. This constraint in turn causes the issue of periodic broadcast of routing tables, which consumes a large available bandwidth. DVMRP supports only the source-based multicast tree. Thus, this protocol is appropriate for multicasting data among a limited number of distributed receivers located close to the source.

15.2.2 Internet Group Management Protocol (IGMP)

The *Internet Group Management Protocol* (IGMP) is used for TCP/IP between a receiver and its immediate multicast-enabled routers reporting multicast group information. This protocol has several versions and is required on all machines that receive IP multicast. As the name suggests, IGMP is a group-oriented management protocol that provides a dynamic service to registered individual hosts in a multicast group on a particular network.

When it sends an IGMP message to its local multicast router, a network host in fact identifies the group membership. Routers are usually sensitive to these types of messages and periodically send out queries to discover which subnet groups are active. When a host wants to join a group, the group's multicast address receives an IGMP

Byte:

	1	1	2	4	4	
Type	Maximum Response Time	Checksum	Group Address	Resv S QRV QQIC N	Source Addresses [1] ... [N]	

Figure 15.4 IGMP packet format

message stating the group membership. The local multicast router receives this message and constructs all routes by propagating the group membership information to other multicast routers throughout the network.

The IGMP packet format has several versions; Figure 15.4 shows version 3. The first 8 bits indicate the message *type*: which may be one of *membership query*, *membership report v₁*, *membership report v₂*, *leave group*, and *membership report v₃*. Hosts send IGMP membership reports corresponding to a particular multicast group, expressing an interest in joining that group.

IGMP is compatible with TCP/IP, so the TCP/IP stack running on a host forwards the IGMP membership report when an application opens a multicast socket. A router periodically transmits an IGMP membership query to verify that at least one host on the subnet is still interested in receiving traffic directed to that group. In this case, if no response to three consecutive IGMP membership queries is received, the router times out the group and stops forwarding traffic toward that group. (Note that *v_i* refers to the version of the protocol the membership report belongs to.)

IGMP version 3 supports *include* and *exclude* modes. In *include* mode, a receiver announces the membership to a host group and provides a list of source addresses from which it wants to receive traffic. With *exclude mode*, a receiver expresses the membership to a multicast group and provides a list of source addresses from which it does not want to receive traffic. With the *leave group* message, hosts can report to the local multicast router that they intend to leave the group. If any remaining hosts are interested in receiving the traffic, the router transmits a group-specific query. In this case, if the router receives no response, the router times out the group and stops forwarding the traffic.

The next 8 bits, *max response time*, are used to indicate the time before sending a response report (default, 10 seconds). The *Resv* field is set to 0 and is reserved for future development. The next field is *S* flag, to suppress router-side processing. *QRV*

indicates the querier's robustness variable. $QQIC$ is the querier's query interval code. N shows the number of sources, and *Source Address* [i] provides a vector of N individual IP addresses.

15.2.3 Multicast OSPF (MOSPF) Protocol

The *Multicast Open Shortest Path First* (MOSPF) protocol, an extension to the unicast model of OSPF discussed in Chapter 7, constructs a link-state database with an advertisement mechanism. Let's explore what new features a link-state router requires to become capable of multicast functions.

Link-State Multicast

As explained in Chapter 7, *link-state routing* occurs when a node in a network has to obtain the state of its connected links and then send an update to all the other routers once the state changes. On receipt of the routing information, each router reconfigures the entire topology of the network. The link-state routing algorithm uses Dijkstra's algorithm to compute the least-cost path.

The multicast feature can be added to a router using the link-state routing algorithm by placing the spanning tree root at the router. The router uses the tree to identify the best next node. To include the capability of multicast support, the link-state router needs the set of groups that have members on a particular link added to the *state* for that link. Therefore, each LAN attached to the network must have its host periodically announce all groups it belongs to. This way, a router simply detects such announcements from LANs and updates its routing table.

Details of MOSPF

With OSPF, a router uses the flooding technique to send a packet to all routers within the same hierarchical area and to all other network routers. This simply allows all MOSPF routers in an area to have the same view of group membership. Once a link-state table is created, the router computes the shortest path to each multicast member by using Dijkstra's algorithm. This protocol for a domain with n LANs is summarized as follows.

Begin MOSPF Protocol

1. **Define:** $N_i = \text{LAN } i \text{ in the domain, } i \in \{1, 2, \dots, n\}$
 $G_j(i) = \text{Multicast group } j \text{ constructed with connections to } N_i$
 $R_i = \text{Router attached to } N_i$
2. **Multicast groups:** R_i maintains all N_j group memberships.

3. **Update:** Use the link-state multicast whereby each router R_i floods all its multicast group numbers, $G_j(i)$, to all its directly attached routers.
4. Using the shortest-path tree algorithm, each router constructs a least-cost tree for all destination groups.
5. When it arrives at a router, a multicast packet finds the right tree, makes the necessary number of copies of the packet, and routes the copies. ■

MOSPF adds a link-state field, mainly membership information about the group of LANs needing to receive the multicast packet. MOSPF also uses Dijkstra's algorithm and calculates a shortest-path multicast tree. MOSPF does not flood multicast traffic everywhere to create state. Dijkstra's algorithm must be rerun when group membership changes. MOSPF does not support sparse-mode tree (shared-tree) algorithm. Each OSPF router builds the unicast routing topology, and each MOSPF router can build the shortest-path tree for each source and group. Group-membership reports are broadcast throughout the OSPF area. MOSPF is a dense-mode protocol, and the membership information is broadcast to all MOSPF routers. Note that frequent broadcasting of membership information degrades network performance.

Example. In Figure 15.5, each of the five LANs in a certain domain has an associated router. Show an example of MOSPF for multicasting from router R_1 to seven servers located in LANs 1, 2, and 3.

Solution. Multicast groups 1 and 2 are formed. For group 1, the best tree is implemented using copying root R_4 . For group 2, the best tree is implemented using copying root R_7 .

15.2.4 Protocol-Independent Multicast (PIM)

Protocol-independent multicast (PIM) is an excellent multicast protocol for networks, regardless of size and membership density. PIM is “independent” because it implements multicasting independently of any routing protocol entering into the multicast routing information database. PIM can operate in both *dense mode* and *sparse mode*. Dense-mode is a flood-and-prune protocol and is best suited for networks densely populated by receivers and with enough bandwidth. This version of PIM is comparable to DVMRP.

Sparse-mode PIM typically offers better stability because of the way it establishes trees. This version assumes that systems can be located far away from one another and that group members are “sparsely” distributed across the system. This protocol for a domain with n LANs is summarized as follows.

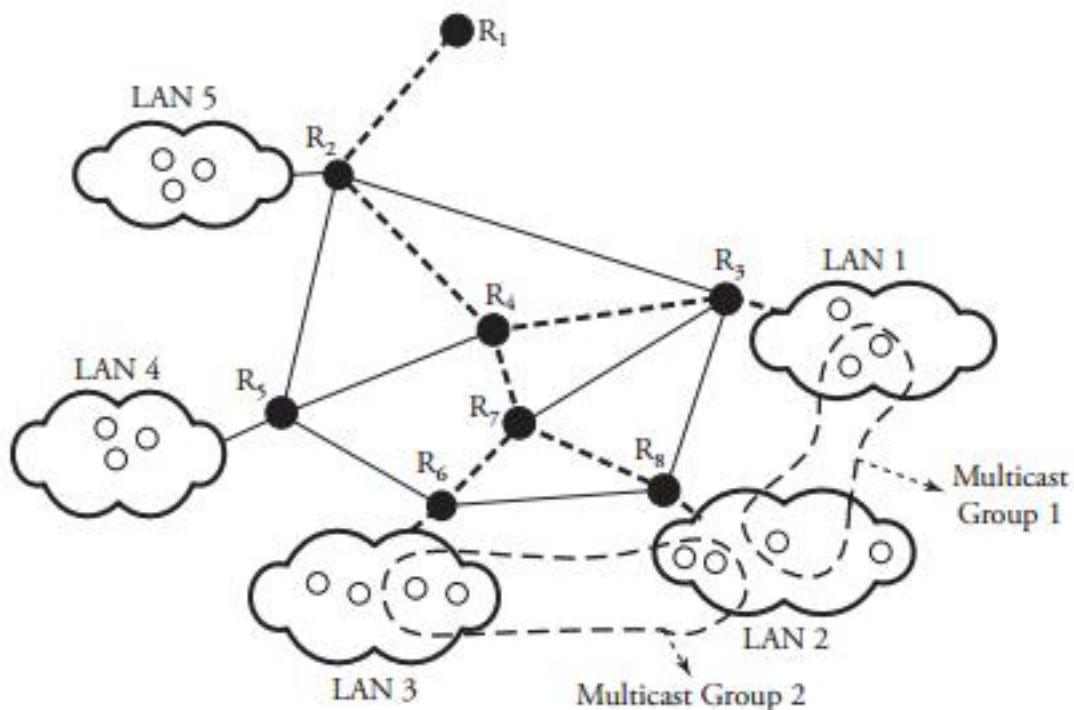


Figure 15.5 Use of MOSPF to multicast from router R_1 to seven servers located in two different multicast groups spread over three LANs

Begin PIM Algorithm

1. **Define:** $N_i = \text{LAN } i \text{ in the domain } i \in \{1, 2, \dots, n\}$
 $G_j(i) = \text{Multicast group } j \text{ constructed with connections to } N_i$
 $R_i = \text{The router attached to } N_i$
2. **Multicast groups:** R_i maintains all N_j group memberships.
3. **Rendezvous router:** Using the sparse-mode tree algorithm, each group $G_j(i)$ chooses a rendezvous router.
4. **Update:** Each Router R_i updates its rendezvous router whenever there is a change in the group.
5. When it arrives at a rendezvous router, a multicast packet finds the right tree and routes the packet. ■

Sparse-mode PIM first creates a shared tree, followed by one or more source-specific trees if there is enough traffic demand. Routers join and leave the multicast group *join* and *prune* by protocol messages. These messages are sent to a rendezvous router allocated to each multicast group. The rendezvous point is chosen by all the routers in a group and is treated as a meeting place for sources and receivers. Receivers join the

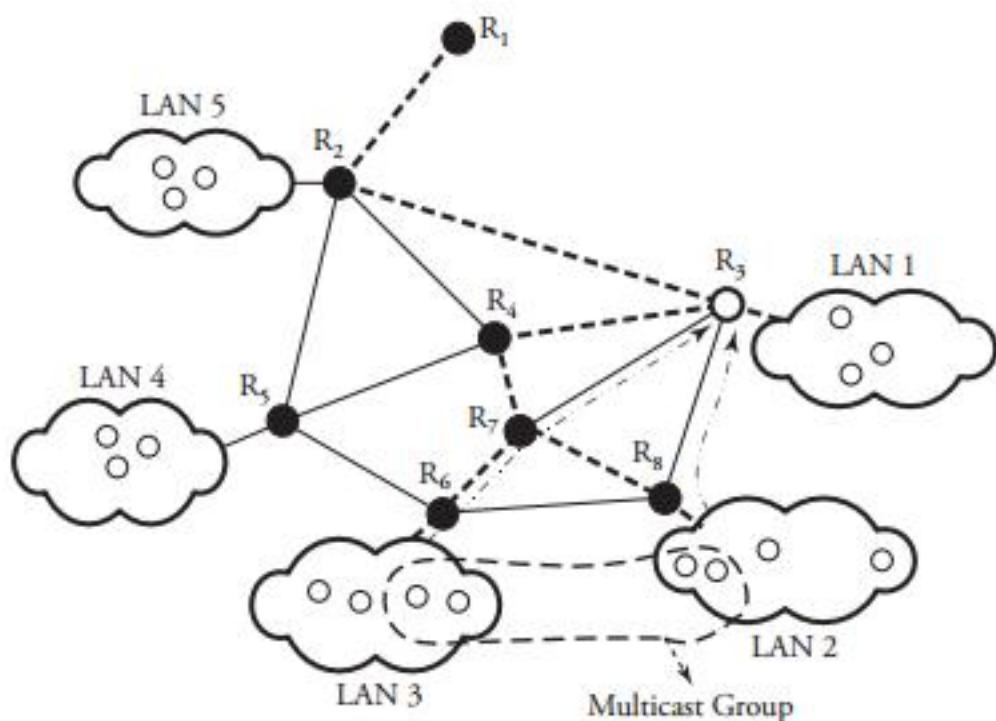


Figure 15.6 Sparse-mode PIM for multicasting from router R_1 to four servers in a multicast group spread over LANs 2 and 3

shared tree, and sources register with that rendezvous router. Note that the shortest-path algorithm made by the unicast routing is used in this protocol for the construction of trees.

Each source registers with its rendezvous router and sends a single copy of multicast data through it to all registered receivers. Group members register to receive data through the rendezvous router. This registration process triggers the shortest-path tree between the source and the rendezvous router. Each source transmits multicast data packets encapsulated in unicast packets to the rendezvous router. If receivers have joined the group in the rendezvous router, the encapsulation is stripped off the packet, and it is sent on the shared tree. This kind of multicast forwarding state between the source and the rendezvous router enables the rendezvous router to receive the multicast traffic from the source and to avoid the overhead of encapsulation.

Example. In Figure 15.6, the five LANs in a certain domain each have an associated router. Show an example of sparse-mode PIM for multicasting from router R_1 to four servers located in a multicast group spread over LANs 2 and 3.

Solution. Multicasting is formed with R_3 the associated rendezvous router for this group. Thus, the group uses a reverse unicast path as shown to update the rendezvous

router of any joins and leaves. For this group, the multicast tree is implemented using copying root R_7 .

15.2.5 Core-Based Trees (CBT) Protocol

In sparse mode, forwarding traffic to the rendezvous router and then to receivers causes delay and longer paths. This issue can be partially solved in the *core-based tree* (CBT) protocol, which uses bidirectional trees. Sparse-mode PIM is comparable to CBT but with two differences. First, CBT uses bidirectional shared trees, whereas sparse-mode PIM uses unidirectional shared trees. Clearly, bidirectional shared trees are more efficient when packets move from a source to the root of the multicast tree; as a result, packets can be sent up and down in the tree. Second, CBT uses only a shared tree and does not use shortest-path trees.

CBT is comparable to DVMRP in WANs and uses the basic sparse-mode paradigm to create a single shared tree used by all sources. As with the shared-trees algorithm, the tree must be rooted at a *core* point. All sources send their data to the core point, and all receivers send explicit join messages to the core. In contrast, DVMRP is costly, since it broadcasts packets, causing each participating router to become overwhelmed and thereby requiring keeping track of every source-group pair. CBT's bidirectional routing takes into account the current group membership when it is being established.

When broadcasting occurs, it results in traffic concentration on a single link, since CBT has a single delivery tree for each group. Although it is designed for intradomain multicast routing, this protocol is appropriate for interdomain applications as well. However, it can suffer from latency issues, as in most cases, the traffic must flow through the core router. This type of router is a bottleneck if its location is not carefully selected, especially when transmitters and receivers are far apart.

15.2.6 Multicast Backbone (MBone)

The first milestone in the creation of a practical multicast platform was the development of the *multicast backbone* (MBone), which carried its first worldwide event when several sites received audio simultaneously. The multicast routing function was implemented using unicast-encapsulated multicast packets. The connectivity among certain receivers was provided using point-to-point IP-encapsulated *tunnels*. Figure 15.7 shows an example of tunneling among routers in the early version of MBone. Each tunnel connects two end points via one logical link and crosses several routers. In this scenario, once a packet is received, it can be sent to other tunnel end points or broadcast to local members. The routing in earlier version of MBone was based on DVMRP and IGMP.

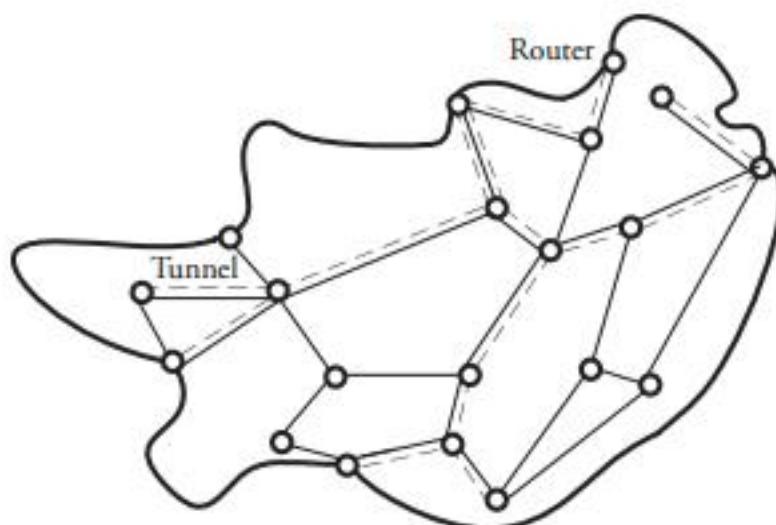


Figure 15.7 Tunneling in the multicast backbone

15.3 Interdomain Multicast Protocols

Interdomain multicast protocols are designed for hierarchical and Internet-wide multicast purposes. Within a domain, a network manager can implement any routing protocol desired. The challenge in interdomain multicast administration is choosing the best external link to route to hosts in an external domain. Among the protocols in this category are *multiprotocol Border Gateway Protocol* (MBGP), *Multicast Source Discovery Protocol* (MSDP), and *Border Gateway Multicast Protocol* (BGMP).

15.3.1 Multiprotocol BGP (MBGP)

Multiprotocol BGP (MBGP) is an extension to its unicast version, *Border Gateway Protocol* (BGP). In Figure 15.8, three domains are connected through two types of paths: a unicast path handled by BGP and a multicast path handled by MBGP. With BGP, the multicast routing hierarchy operates the same way as multiple unicast routing does. Between each two domains, two corresponding border routers compute the set of domain parameters that should be traversed to reach any network. Typically, the parameters in one domain are not known or trusted by the others.

Each domain-management system advertises the set of routes that can reach a particular destination. Packets are routed on a hop-by-hop basis. Thus, MBGP can determine the next hop to a host but cannot provide multicast tree-construction functions. Sources within a domain register with a rendezvous router, and receivers send joins to this router. This join message is required to find the best reverse path to the source.

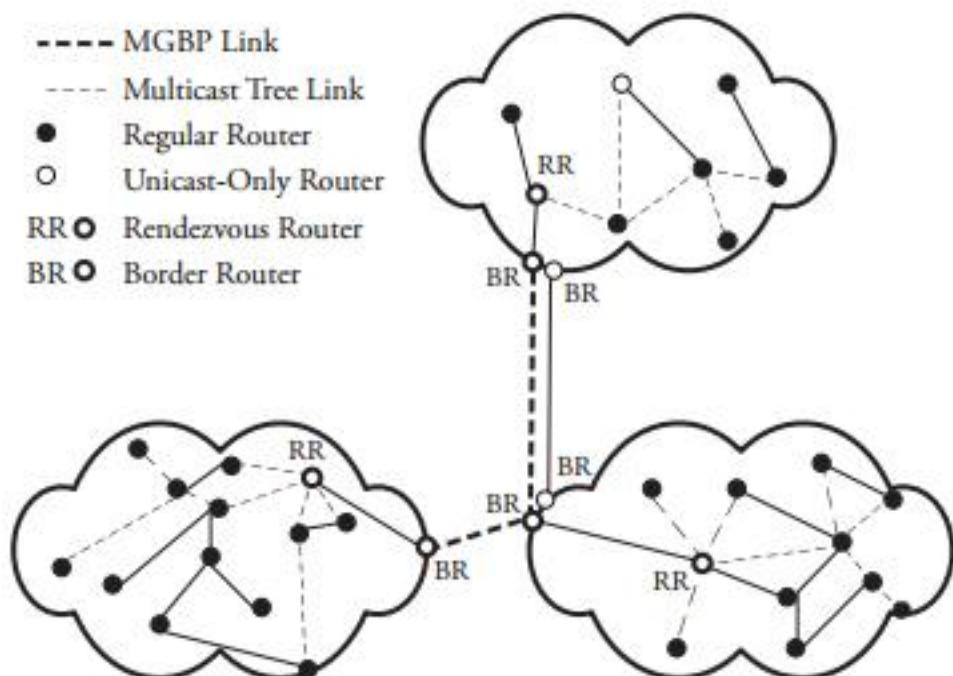


Figure 15.8 MBGP: interdomain multicast routing

This protocol can handle multiprotocol paths, and each participating router needs to know only the topology of its own domain and the paths to reach each of the other domains. Since MBGP is an interdomain protocol, a class D address is not carried in an MBGP message, and it does not carry information about multicast groups. Because the time and frequency of join messages transmission are not determined in this protocol, and multicast trees in various domains are not connected, other multicast protocol choices are more attractive.

15.3.2 Multicast Source Discovery Protocol (MSDP)

One of the main issues in interdomain multicasting is how to inform a rendezvous router in one domain while there are sources in other domains. In other words, rendezvous routers of two adjacent domains are not able to communicate with each other when one receives a source register message. In practice, only one rendezvous point is in each domain. This issue becomes more critical when group members should be spread over multiple domains. In this case, multicast trees among domains are not connected. As a result, the traffic can reach all receivers of a domain, but any sources outside the domain stay disjoint.

The *Multicast Source Discovery Protocol* (MSDP) has potential solutions to these issues. Figure 15.9 shows how this protocol operates. A unique feature of this protocol is

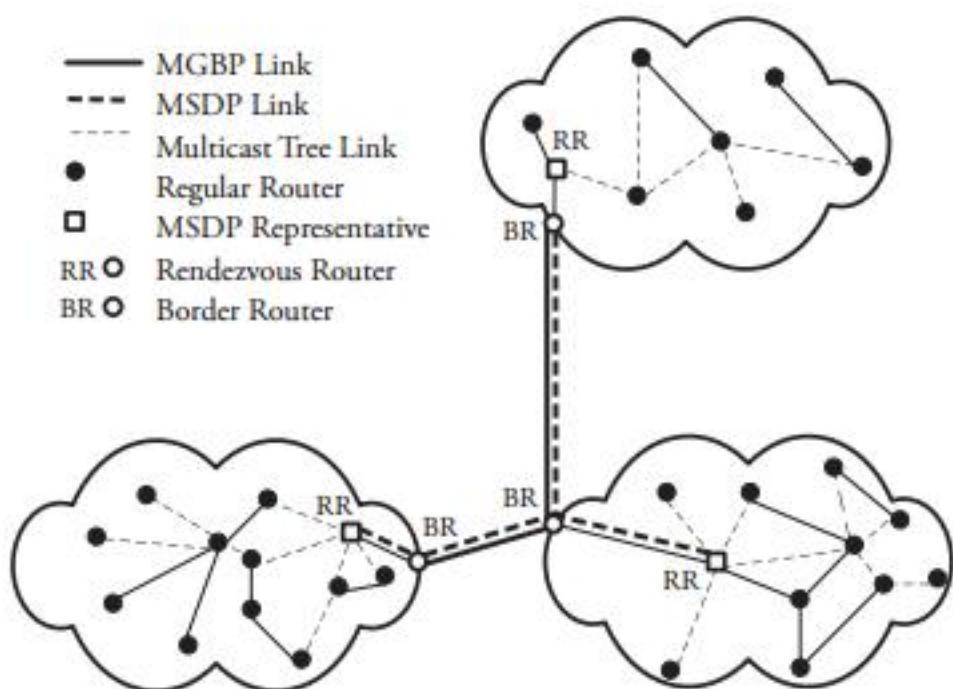


Figure 15.9 MSDP: interdomain multicast routing

that it assigns representatives in each domain. A representative reports to other domains the existence of active sources. A new source for a group must first register with the domain's rendezvous router.

Each MSDP representative in the domain detects the existence of the new source and sends a message to all MSDP representatives, as shown. The representatives check whether the broadcasting representative has sent the message through the correct path to prevent possible message looping. Once this process is complete and the message is on the correct router interface, the message is forwarded to all remaining associated representatives. An MSDP representative and the rendezvous router within a domain can be the same point; this is the checkpoint where the state of group membership is checked. If it has the correct state, the router sends a join message to the source address mentioned in the message. Here, we assume that the intradomain multicast is performed using PIM, whereby a join message is generated and processed. The message is then forwarded on the multicast tree by the rendezvous router; once all group members receive data attached to the message, they may use a shortest-path tree, using sparse-mode PIM. This process ends when all the representatives finish the process.

This multicast procedure uses a combination of three protocols: sparse-mode PIM, MBGP, and MSDP. Although this multicast procedure has been well accepted as a

practical multicast method, its complexity results in a timing issue. One aspect is scalability. Also, when sources are bursty or group member join and leave events frequently, the overhead of managing the group can be noticeable. This fact in turn creates the timeout issue, whereby the period of silence between packet bursts becomes greater than the forwarding-state timeout. MSDP solves this issue by selecting and processing every n packets in burst. However, this trick is not quite a practical solution when a fairly large network is under multicasting.

15.3.3 Border Gateway Multicast Protocol (BGMP)

The *Border Gateway Multicast Protocol* (BGMP) is based on the construction of bidirectional shared trees among domains using a single root. Finding the best domain to place the root of such shared trees is a challenge, but several solutions are available. One of the methods of address resolution is the *Multicast Address-Set Claim* (MASC) protocol, which guarantees the immediate resolution of address collisions.

Another method of resolving this issue is to use *root-addressed multicast architecture* (RAMA) by which a source is selected as the root of the tree. This way, the complexity of root placement in other multicast routing protocols can be eliminated. RAMA is of two types. The first type is *express multicast*: The root of the tree is placed at the source, and group members send join messages on the reverse path back to the source. This protocol is aimed at systems that use logical channels, such as single-source multimedia applications, TV broadcast, and file distribution. The second type is *simple multicast*: Multiple sources per group are allowed. In this case, one source must be selected, and the root of the tree is put at this node as a primary and first-hop router. Then, all receivers send join messages to the source. The next step is to construct a bidirectional tree through which additional sources send packets to the root. Since a bidirectional tree is constructed, packets arrive at a router in the tree and are forwarded both downstream to receivers and upstream to the root.

15.4 Node-Level Multicast Algorithms

Multicasting techniques can also be used at the router level. To implement a multicast connection, a binary tree is normally constructed with the source switch port at the root and the destination switch ports at the leaves. Internal nodes act as relay points that receive packets and make copies. A number of such multicast methods are used. One is a *tree-based multicast algorithm* using a separate copy network. The *Boolean*

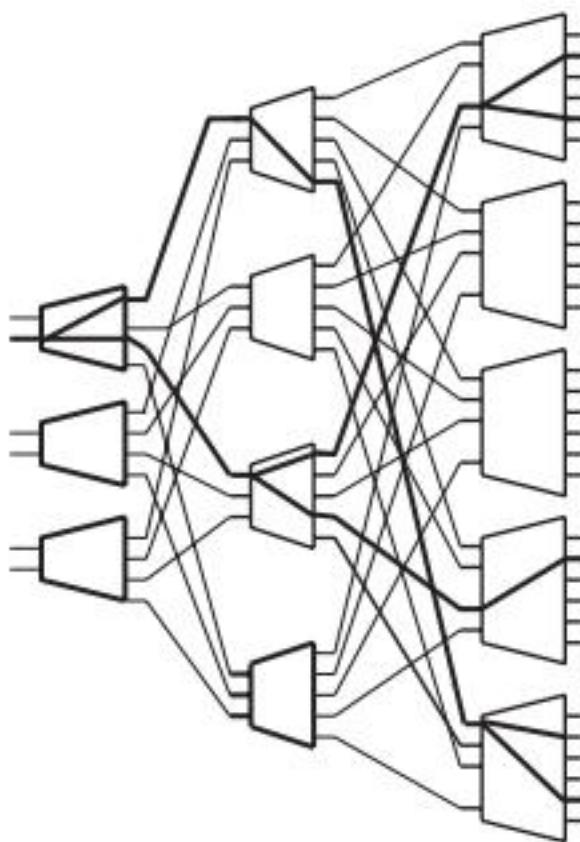


Figure 15.10 Tree-based multicasting in an expansion switch

splitting multicast algorithm is used for multistage switches. The third technique is the *packet recirculation multicast algorithm*. The fourth is multicasting in *three-dimensional switches*.

15.4.1 Tree-Based Multicast Algorithm

Imagine that a multicast tree algorithm must be applied on a multistage switch fabric, as shown in the expansion switch in Figure 15.10. Multicasting is implemented as a tree structure. A source generates a packet and sends it out to the first crossbar switch. The packet may have a field that specifies how many copies of this packet are to be made. All copies may be destined for other crossbars, in which more copies of the packet are made, and some packets may move to their destinations.

For implementing a *tree-based multicast algorithm*, the copying tasks in multistage switch fabrics typically involve a copy switch and a routing switch. An example of such systems consists of two Beneš networks for constructing the copying and routing networks. The following algorithm presents the multicast steps within a copy network with n inputs or outputs constructed with $d \times d$ switch element:

Define: $k = \log_d n$ $r = \text{Number of stages}$ $j = \text{Stage number, } j \in \{1, 2, \dots, r\}$ $F = \text{Global number of copies, } F \in \{1, 2, \dots, n\}$ $F_j = \text{Number of copies remaining when packet arrives at stage } j$ $f_j = \text{Local number of copies made at stage } j$ **Begin Tree-Based Multicast Algorithm****For Stages** $1 \leq j \leq r - k \Rightarrow$

$$F_j = F$$

$$f_j = 1$$

For Stages $(r - k) + 1 \leq j \leq r \Rightarrow$

$$F_j = \left\lceil \frac{F_{j-1}}{f_{j-1}} \right\rceil$$

$$f_j = \left\lceil \frac{F_j}{d^{r-j}} \right\rceil$$

■

In the copy network shown in Figure 15.11, packets are replicated as designated by the initial *global number of copies*, F , given in the routing-control field of the packet header. The global number of copies refers to the total number of packet copies requested. Let F_j be the remaining number of copies of a packet when it arrives at stage j , and let f_j be the number of copies made locally at stage j . First, the algorithm initializes F_j to F and f_j to 1. The copying method is such that the replication of the packets takes place stage by stage within the network in order to distribute the traffic as evenly as possible. The routing is either point to point or multipoint connection.

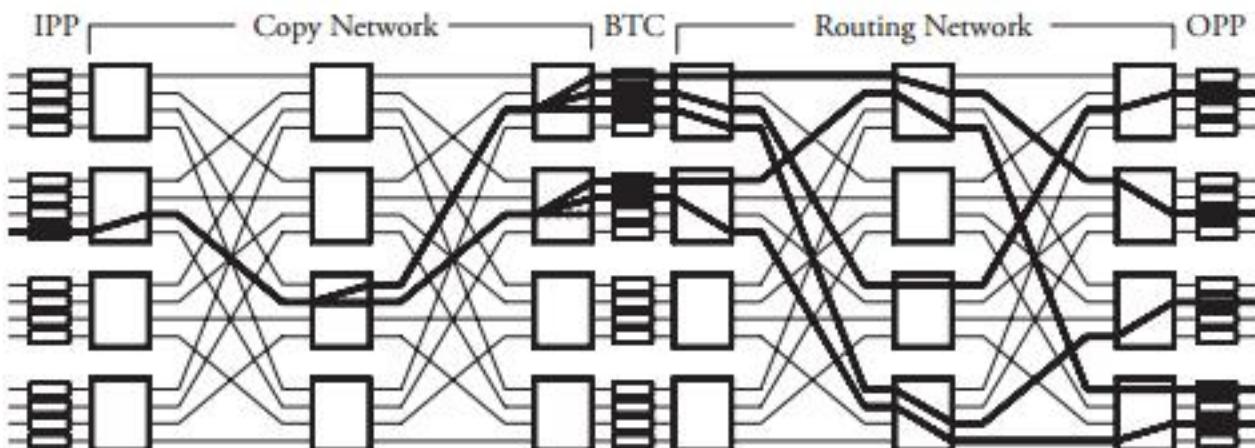


Figure 15.11 Tree-based multicasting through a three-stage copy network followed by a routing network

Consider a copy network with k stages. When multipoint connections are requested, a packet is routed randomly in the first $k - 1$ stages. In the last k stages, the packet is copied. The operation $F_j = \lceil \frac{F_{j-1}}{d^{k-j}} \rceil$ computes a new number of copies for a packet at stage j , with the local number of copies $f_{j-1} = \lceil \frac{F_j}{d^{k-j}} \rceil$. The algorithm is set up to generate the final number of copies that appear on the outputs to be the smallest multiple of 2 greater than or equal to F .

This technique reduces the hardware complexity in the controller. If the final number of copies that appear on the outputs is more than requested, the unnecessary packet copies can be easily thrown away. This task is completed by the broadcast translated circuit (BTC), shown in Figure 15.11. BTCs receive all the copies of a packet. A central monitoring mechanism for all BTCs compares the number of requested copies read from the packet's header and the number of copies made in the copy network. This mechanism then decides to remove unused copies, based on this comparison. For the routing network, routing is similar to that in a point-to-point copy network.

Example. Find all numbers of copies in every stage of a $B_{16,4}$ switching network in which the global number of copies of an incoming packet is $F = 5$ and copies of the packet are to be routed to output port numbers 1, 6, 9, 12, and 14.

Solution. The copy network has $r = 2 \log_d n - 1 = 3$ stages and $k = 2 \log_d n$. For stages $1 \leq j \leq r - k = 1$ or at stage $j = 1$, we have $F_1 = 5$ and the local number of copies $f_1 = 1$, so the packet gets distributed. For stages $(r - k) + 1 = 2 \leq j \leq r = 3$, starting at stage $j = 2$, $F_2 = \lceil F_1/d \rceil = 5$, and $f_2 = \lceil F_2/d^{k-j} \rceil = 2$; thus, two copies are made at this stage and guided to two switches at the last stage ($j = 3$). At the third stage, $F_3 = \lceil F_2/d \rceil = 3$, and the local number of copies $f_3 = \lceil F_3/d^{k-j} \rceil = 3$. Therefore, for each of these switches, three copies of the packet are made. Note that the sum of these copies (6) has exceeded the requested global number of copies (5), so the one additional copy is thrown away by the corresponding BTC (see Figure 15.11).

15.4.2 Boolean Splitting Multicast Algorithm

The *Boolean splitting multicast algorithm* is a method of copying packets in any space-division switch fabric with n inputs or outputs constructed with 2×2 switch elements and therefore with $\log_2 n$ stages. Consider a Banyan network in which switching nodes replicate packets based on 2-bit header information. The following algorithm presents multicast steps within a copy network.

Define:

$$k = \log_d n$$

r = Number of stages

j = Stage number, $j \in \{1, 2, \dots, r\}$

f_j = Local number of copies made at stage j

$a(j)$ = Lowest requested output address at stage j : $a_k a_{k-1} \dots a_1$

$A(j)$ = Highest requested output address at stage j : $A_k A_{k-1} \dots A_1$

Begin Boolean Splitting Multicast Algorithm

For Stages $1 \leq j \leq r - k \Rightarrow$

$$F_j = F$$

$$f_j = 1$$

For Stages $(r - k) + 1 \leq j \leq r \Rightarrow$

Sort the output addresses

Compare a_j with A_j

If $a_j = A_j = 0 \Rightarrow$

Send the packet on crossbar output 0 (link 0).

For the packet forwarded to link 0 \Rightarrow

$$a_{j+1} = a_j$$

$$A_{j+1} = A_k \dots A_{k-(j-1)} 01 \dots 1$$

If $a_j = A_j = 1 \Rightarrow$

Send the packet on crossbar output 1 (link 1),

For the packet forwarded to link 1 \Rightarrow

$$a_{j+1} = a_k \dots a_{k-(j-1)} 10 \dots 0$$

$$A_{j+1} = A_j$$

If $a_j = 0$ and $A_j = 1$, or $a_j = 1$ and $A_j = 0$, \Rightarrow

Replicate the packet.

Send one copy of the packet carrying upper half of addresses on link 0.

Send one copy of the packet carrying lower half of addresses on link 1.

■

The switch fabric replicates packets according to a Boolean interval-splitting algorithm, based on the address interval in the new header. An output port address of the switch fabric that receives a copy of the packet is represented by two k -bit binary numbers. Suppose that a packet arrives at an arbitrary crossbar at stage $j - 1$ from a previous crossbar j . If we represent the minimum number by a k -bit address $a_k \dots a_1$ and the maximum number by a k -bit address $A_k \dots A_1$, we can make the following observations: If $a_j = A_j = 0$ or $a_j = A_j = 1$, the packet is sent out on crossbar output

0 or 1, respectively. If $a_j = 0$ and $A_j = 1$, the network replicates the packet and sends both packets out on both links 0 and 1.

If replication takes place, the packet header needs to be modified by splitting the original address interval into two subintervals, which can be expressed by the two following main recursions: $a_{j+1} = a_j$ and $A_{j+1} = A_k \cdots A_{k-(j-1)}01 \cdots 1$ for the packet sent out on Link 0, and $a_{j+1} = a_k \cdots a_{k-(j-1)}10 \cdots 0$, and $A_{j+1} = A_j$ for the packet sent out on Link 1. This modification is a routine operation, meaning that it is independent of the header contents and can be considered built-in hardware.

Example. Show the detail of the Boolean splitting multicast algorithm applied to copy a packet in a Banyan network $Y_{16,2}$. Find all numbers of copies in every stage of a switching fabric, where the global number of copies of an incoming packet is $F = 6$, and copies of the packet are to be routed to output port numbers 5, 6, 7, 8, 9, and 10.

Solution. The switch fabric has $k = \lceil \log_{16} 2 \rceil = 4$ stages. At stage $j = 1$, we have $F_1 = 6$ $a(1) = 0101$ and $A(1) = 1010$; consequently, $a_1 = 0$ is compared with $A_1 = 1$. Since these two bits are different, 0101 to 0111 are assigned to link 0, and 1000 to 1010 are assigned to link 1. If we continue the multicast process, the results of this example will be as shown in Figure 15.12.

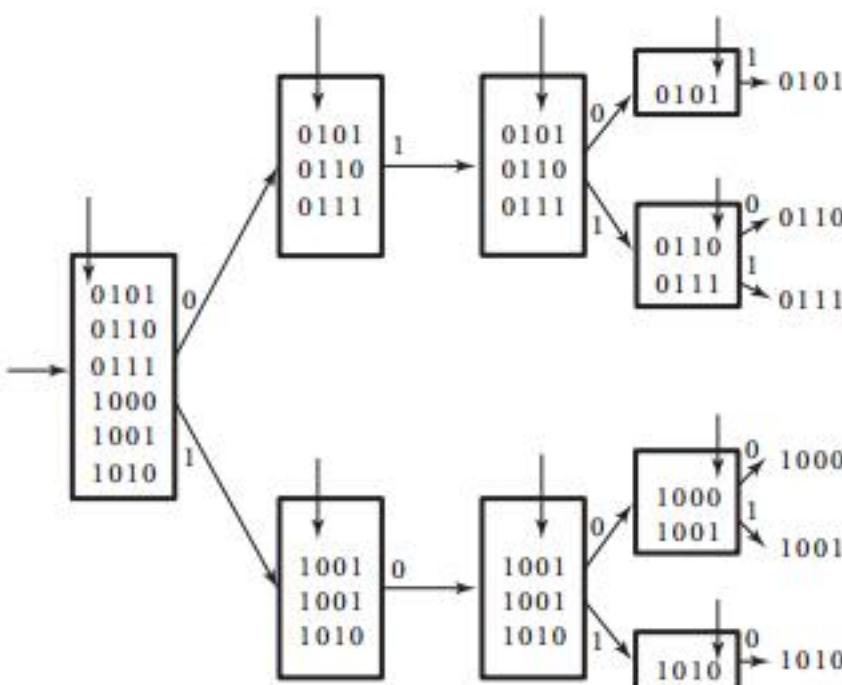


Figure 15.12 The Boolean splitting multicast algorithm in a multistage switch

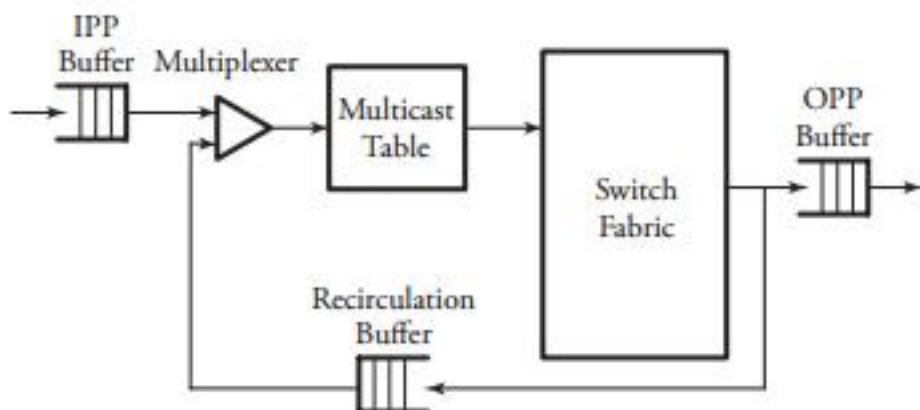


Figure 15.13 Recirculation multicasting

15.4.3 Packet Recirculation Multicast Algorithm

Packet recirculation (recycling) is another feasible method for constructing large switching fabrics for broadband switching applications. To implement a multicast connection, a binary tree is constructed with the source port at its root and the destination switch port at its leaves. This technique can be used for almost all kinds of space-division switch fabrics.

If a switch fabric is constructed by a multistage interconnected network, internal crossbars can act as relay points to accept packets and recycle them back into the switch fabric after packet relabeling. New labels carry the information about the destination pair, identifying the next two switch ports to which they are to be sent. Figure 15.13 illustrates the recirculation technique. A *multicast table* provides an output port/IP address pair. This pair is added to the packet header; thus, two additional bits indicate whether the pair is to be recirculated again. An *IPP buffer* holds packets received from the input link and forwards them to the switching network. An *OPP buffer* holds packets waiting to be transmitted on outgoing links. A *recirculation buffer* provides buffering to the packets that need to be recycled.

Packet recirculation and copying within the switch fabric is straightforward. Figure 15.14 illustrates packet recirculation and copying. Suppose that a is a source that generates packets to be delivered to output ports b , c , d , and e and that x and y are relay points. Suppose that a packet entering at input port a must have several copies. For such a packet, there may be entries of the type (x, j) and (e, k) in the multicast table. This can be interpreted as making two copies of the packet: one to be *recirculated* back to port x on address j ; the other one, to port e with address k . The multicast table entry at x may now have entries (b, n) and (y, m) , as seen. Typically, each packet header has 2 bits to indicate whether a packet needs to be recirculated.

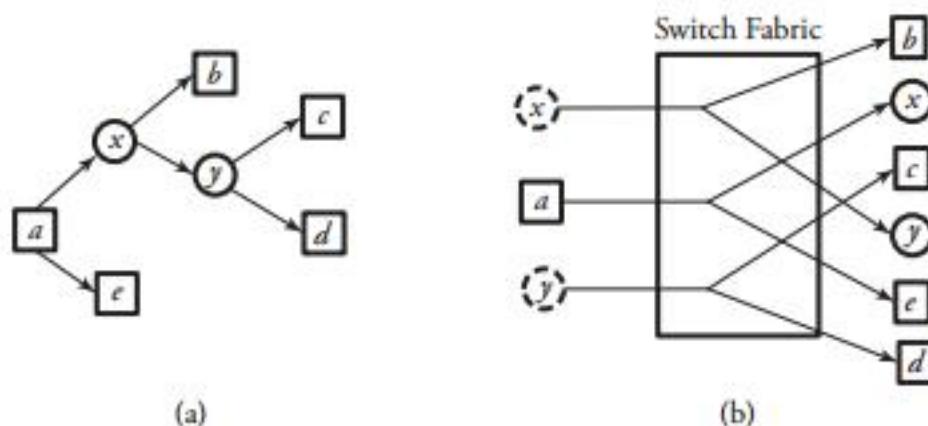


Figure 15.14 Within the switch fabric of routers: (a) packet recirculation and (b) copying

In summary, an end point that is added needs to have a parent. A good candidate is the one that does not have heavy recirculation traffic. Like the parent of the new end point, the intermediate point also is new, so it has to be put into the multicast table entry of another node, which becomes the grandparent of the new end point, whose child will now become a sibling of the new end point.

To remove a connection, we can consider two cases. We should examine whether the output to be removed has no grandparent but its sibling has children; in that case, we replace the parent's multicast table entry with the sibling's children. If the output to be removed has no grandparent and its sibling has no children, we simply drop the output to be removed from its parent's multicast table entry, and the connection reverts to a simple point-to-point connection. Note that since packets must be resequenced when they exit from the system, the resequencing buffer at the output port processor (OPP) of a router must be dimensioned to delay packets long enough so that slow packets have a chance to catch up with fast packets.

15.4.4 Multicasting in Three-Dimensional Switches

This switch uses a Clos network as a building block module. An n -port Clos network is a nonblocking network if $k \geq 2d - 1$, where d and k are the sizes of the first-stage crossbar switch elements. The three-dimensional structure of the Clos network consists of m parallel planes of the same type and same size Clos network in Figure 15.15. An incoming packet can be demultiplexed among m planes. The demultiplexer does the work of distributing the input packets. In other words, the three-dimensional switching system accepts packets coming from different planes and time multiplexes them onto the same output port. The multicast algorithm for each plane is described as follows.

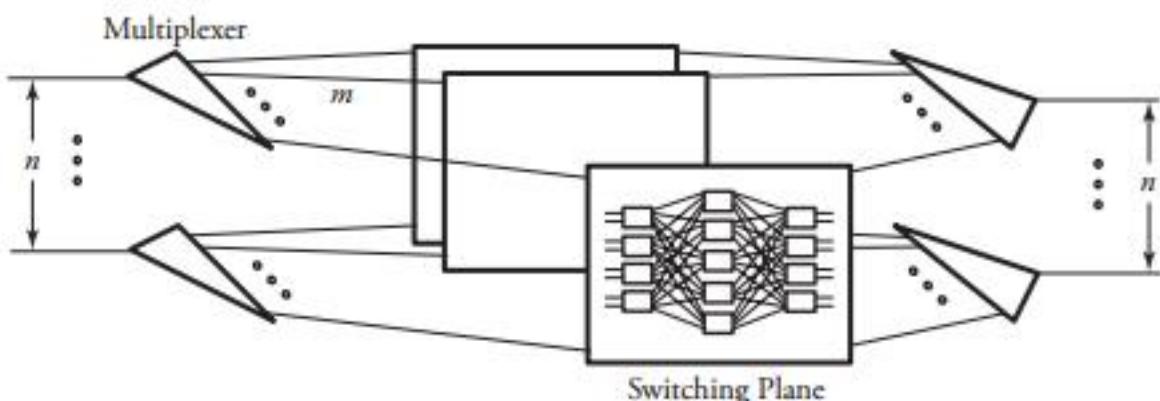


Figure 15.15 Three-dimensional multiplexed Clos network

At the first stage, no routing decision needs to be made. The first-stage switch elements maintain a table that indicates utilization of the center-stage arrays. This table is created on the basis of back-pressure information obtained from the center-stage arrays. A center-stage element least used at any particular time gets the highest priority. In fact, this table can be shared by all the first-stage arrays. Once a multicast packet arrives at the center-stage array, the switch looks at the first g bits of every address, where $g = \log_2 n/d$. The packet is then split according to the number of distinct k bits found. The maximum number of new packets that can be created is n/d . All addresses with the same first g bits are sent to the corresponding switch element in the last stage. Also, since four distinct addresses are identified by the first 2 bits in our example, the packet is split four ways.

The final stage (stage 3) looks at the remaining l bits, where $l = \log_2 n$, in each address of the multicast packet received by the last-stage array. The packet is again split according to the number of different sets of l bits found. The maximum number of packets at this stage is l . The packets are then sent to the appropriate output port corresponding to the last l bits. If a packet is split, the header has to be modified. The new header contains the addresses with the same initial g bits.

A multicast packet on a particular input port requesting F copies of a packet can distribute the request equally between the parallel planes. This reduces the load on any particular plane, leading to a reduction in the use of buffers. An analysis of a three-dimensional Clos network is based on the blocking probability $P_b(i)$ for input number i out of n :

$$P_b(i) = \sum_{j=\lfloor nm/F_{\max} \rfloor + 1}^i \binom{i}{j} \rho^j (1-\rho)^{i-j}, \quad (15.1)$$

where ρ is the offered load per input, and F_{max} is the maximum number of copies requested by any packet. For better performance, a priority is assigned to each plane on the basis of usage. Therefore, planes that are being used less, as determined by the back-pressure information, are given higher priorities. The distribution of multicast packets among planes is done by using the following algorithm.

Begin Multicast Algorithm in 3-D Switches

1. If $F \leq m$: Send a packet to the first F available planes.
2. Each plane makes one copy of the packet and routes it to the appropriate output port, depending on the header information. Thus, we get the required F copies.
3. If $F > m$: Find $\frac{F}{m}$, the number of copies equally distributed among the planes, Q , and the copies left over, R .
4. Choose the highest-priority plane. For the first R planes, add 1 to the number of copies desired to be made by that plane, which is simply the value of Q . That is, the remaining R copies are further divided among the least-busy planes (or the highest-priority planes). The remaining $m - R$ planes now have to make Q copies.
5. The first Q (or $Q + 1$ for those planes that take care of the extra requests) addresses form the address space and add the new header to it, which has information about the destination plane, number of copies, and so on. Repeat this for all addresses.

A distribution circuit at the input has the processing ability to take the packet header and modify it according to the algorithm for distribution of packets to the planes. Some fields are added on top of the original packet. In fact, an original packet is encapsulated into this new packet with the new packet header. ■

15.5 Summary

Communication systems must be able to multicast a message to many users and even route many messages to one user. In a multipoint environment, traditional networks are subject to severe blocking, since they are not typically designed for high-bandwidth communications. Under both multirate and multipoint conditions, the performance of switching networks is closely related to the network architecture.

This chapter focused on multicast techniques, protocols, and algorithms in all levels: from the network layer to the physical layer. After defining some basic terms and techniques—including dense-mode and sparse-mode multicast tree algorithms—we discussed two main classes of protocols. *Intradomain* multicast routing protocols multicasts packets within a domain, and the *interdomain* routing protocol multicasts

packets among domains. Among the intradomain protocols is MOSPF, an extension to the unicast OSPF and based on the dense-mode tree algorithm. PIM protocols use sparse-mode tree algorithms.

The inclusion of multicasting capability into modern networks and routers requires considerable sophistication in the architecture and requires a great deal of performance evaluation in both analysis and simulation. Simplistic approaches at the node level can fail to meet the required network performance. A set of multicast algorithms were presented for the router level. The tree-based technique is appropriate for multistage switch fabrics. Multicasting by packet recirculation is a method that reduces switching-network complexity, resulting in less memory required for routing packets in multicast virtual circuits than with tree-based network architecture. However, extra hardware is needed to implement recirculation.

The next chapter looks at some other special-purpose routing situations. One major topic is how two branches of a geographically separate organization can create a secure route by tunneling and establishing a virtual private network (VPN). Other, related topics covered are multiprotocol label switching, point-to-point communications, and overlay networks.

15.6 Exercises

1. For sending a message from a point to multiple points, compare the trade-offs of using multiple unicast connections, and using any multicast approach, such as DVMRP, explained in this chapter.
2. Discuss the efficiency of MOSPF if the sparse-mode algorithm is used.
3. Consider Figure 15.5, and assume that we want to implement MOSPF from a server in LAN 1 to a multicast group of five members, three group members in LAN 3 and the two others in LAN 4. Also assume that the cost of each link is indicated by the sum of two ending routers' subscripts. For example, the cost of link R₃ to R₈ is 11. If a router ends a LAN, the cost of its associated link is assumed to be 1.
 - (a) Show the cost of all links on the network.
 - (b) Form the least-cost tree for this multicast action, and find the root of copying routers.
4. To optimize the routing delay in a sparse-mode PIM:
 - (a) Find the optimal place for the rendezvous point.
 - (b) Give an example of a spanning tree, and justify your answer.

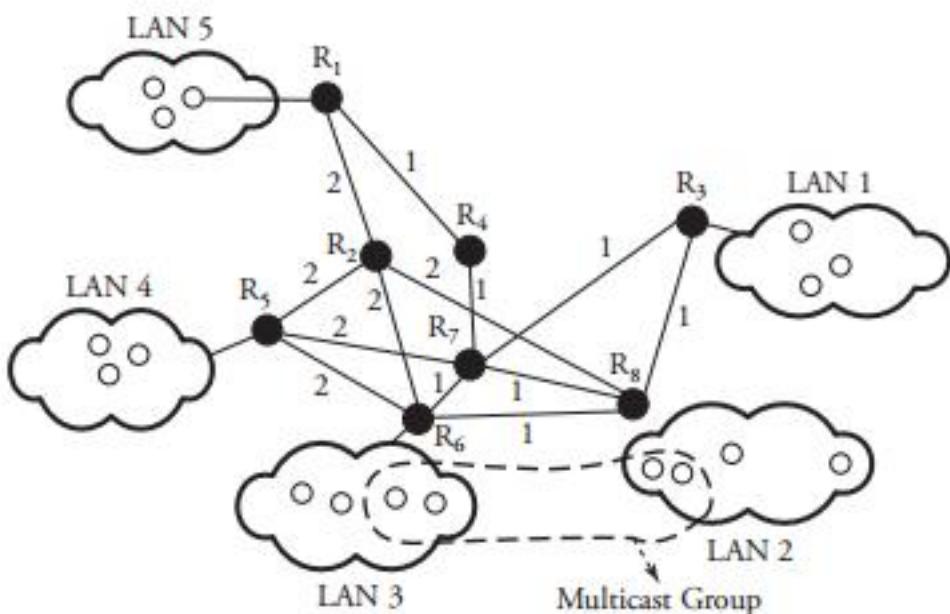


Figure 15.16 Exercise 7 example for multicast protocols

5. Consider Figure 15.6, and assume that we want to implement sparse-mode PIM from a server in LAN 1 to a multicast group of five members, with three group members in LAN 3 and the two others in LAN 4. Also assume that the cost of each link is indicated by the sum of two ending routers' subscripts. For example, the cost of link R₃ to R₈ is 11. If a router ends a LAN, the cost of its associated link is assumed to be 1.
 - (a) Find a reasonably good location for the rendezvous point for this multicast action.
 - (b) Form the least-cost tree for this multicast action, and find the root of copying routers.
6. Repeat exercise 5, but this time use the CBT protocol.
7. In Figure 15.16, five LANs, 1 through 5, are networked through their associated routers, R₃, R₈, R₆, R₅, and R₁. The cost of each link is indicated in the figure. A server in LAN 1 wants to send messages to the indicated multicast group.
 - (a) For deploying the MOSPF protocol, find the multicast tree.
 - (b) For deploying the sparse-mode PIM protocol and using the same tree you obtained in part (a), find the location of the rendezvous point and its associated costs to each multicast group member.

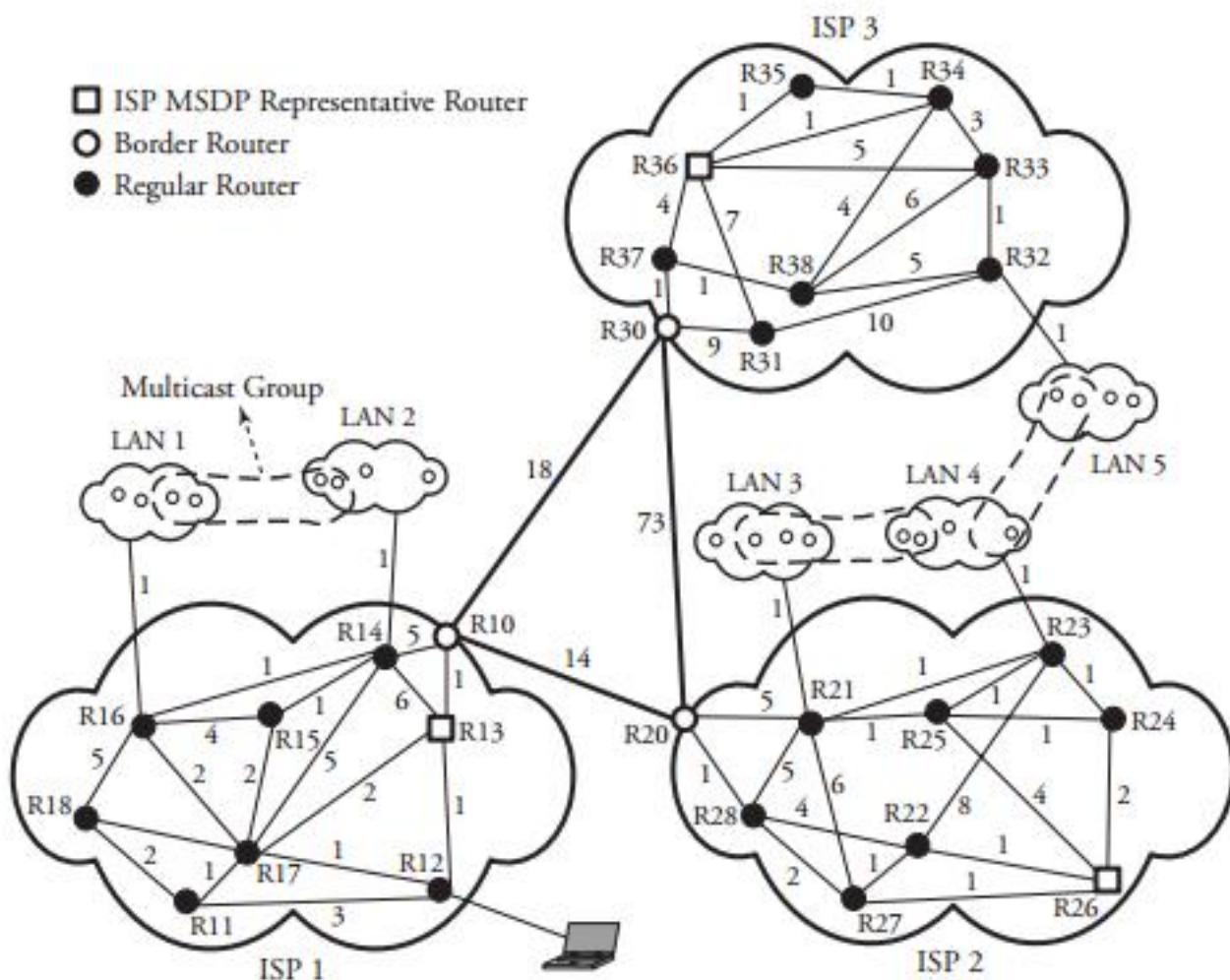


Figure 15.17 Exercise 9: using MSDP for interdomain multicasting and OSPF and sparse-mode PIM protocols for intradomain multicasting

- For a global number of copies $F = 7$, use the tree-based multicast algorithm in a $B_{16,2}$ copy network, and show details of stage-by-stage copying.
- Three major Internet service provider domains—ISP 1, ISP 2, and ISP 3—are connected through three border routers: R10, R20, and R30, respectively, as shown in Figure 15.17. All three ISPs agree to use MSDP for interdomain multicasting. Each domain has selected a router as its ISP MSDP representative, which can also be used as a rendezvous point, if needed. A source in ISP 1 wants to multicast messages to three groups located in various geographical locations, as shown. ISP 1 uses OSPF, and ISP 2 and ISP 3 use sparse-mode PIM for intradomain multicasting. The cost of each link (equal in both directions) is indicated in the figure.

- (a) Indicate all the involving routers in ISP 1, and find the total multicast cost to LAN 1 and LAN 2.
 - (b) Indicate all involving routers in ISP 2 and find the total multicast cost to LAN 3 and LAN 4.
 - (c) Indicate all involving routers in ISP 3 and find the total multicast cost to LAN 5.
10. For a global number of copies $F = 7$, use the Boolean splitting multicast algorithm in a $D_{16,2}$ copy network. Output ports 1, 2, 3, 7, 10, 12, and 15 receive a copy of packet.
- (a) Show details of stage-by-stage copying.
 - (b) Compare the tree-based multicast algorithm with the Boolean splitting multicast algorithm in terms of speed and performance.
11. We want to show the copying mechanism in the switch fabric of routers whose global number of copies $F = 5$. The switch fabric is an Omega network $\Omega_{8,2}$ that has $r = 6$ stages.
- (a) Use the Boolean splitting multicast algorithm. Output ports 1, 2, 4, 5, and 7 receive a copy of the packet. Show the details of stage-by-stage copying.
 - (b) Use the tree-based multicast algorithm to make five copies of a packet, and show the detail of stage-by-stage copying.
12. Consider a small switch fabric of 4×4 crossbar. Design the multicasting portion of its input port processor. Present all the hardware details.
13. *Computer Simulation Project.* Use the computer program you developed for the simulation of the seven-node network in Chapter 7 and modify it to model the network shown in Figure 15.16. Simulate a sparse-mode PIM for multicasting from router R_1 to four recipients located in a multicast group spread over a wide geographic area.
- (a) Construct a multicast group.
 - (b) Write a computer program to do a sparse-mode multicast tree.
 - (c) Assign a rendezvous point.
 - (d) Demonstrate the formation of four packet copies made for the multicast group.

CHAPTER 16

VPNs, Tunneling, and Overlay Networks

Moving on to layers 3 and 4 of the protocol reference model, this chapter introduces some special-purpose networking features: how networks can be *overlaid* or *tunneled*. In networking, tunneling is the encapsulation of a packet from one protocol to another one at the same or higher layer. Among application-specific communication networking cases, *virtual private networks* (VPNs) and *multiprotocol label switching* (MPLS) networks are two important and popular ones discussed in this chapter. These two network infrastructures can also be tied together for certain applications. The topics covered in of this chapter are

- *Virtual private networks (VPNs)*
- *Multiprotocol label switching (MPLS)*
- *Overlay networks*

A VPN is a networking infrastructure whereby a private network makes use of the public network. A VPN maintains privacy by using tunneling protocols and security procedures. The two types of VPNs, each determined by its method of tunneling, are *remote access* and *site to site*.

MPLS networks are a good example of VPNs. In MPLS, multiple labels can be combined in a packet to form a header for efficient tunneling. The *Label Distribution Protocol* (LDP) is a set of rules by which routers exchange information effectively. MPLS uses *traffic engineering* for efficient link bandwidth assignments. Such networks

operate by establishing a secure *tunnel* over a public network. Finally, we look at *overlay networks*. An overlay network is a computer network that creates a virtual topology on top of the physical topology of the public network.

16.1 Virtual Private Networks (VPNs)

A *virtual private network* (VPN) is a data network having connections that make use of public networking facilities. The (VPN) part of public network is set up “virtually” by a private-sector entity to provide public networking services to small entities. With the globalization of businesses, many companies have facilities across the world and use VPNs to maintain fast, secure, and reliable communications across their branches.

VPNs are deployed with privacy through the use of a tunneling protocol and security procedures. Figure 16.1 shows two organizations, 1 and 3, connected through their corresponding routers, forming a tunnel in the public network, such as the Internet. Such a structure gives both private organizations the same capabilities they have on their own networks but at much lower cost. They can do this by using the shared public infrastructure. Creating a VPN benefits an organization benefits by providing

- Extended geographical communication
- Reduced operational cost
- Enhanced organizational management

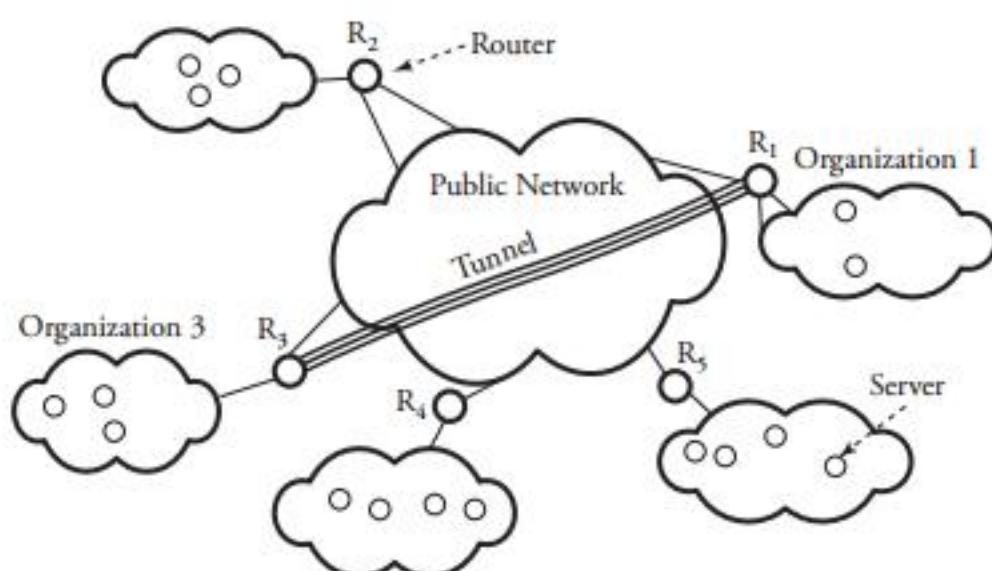


Figure 16.1 Two organizations connected through a tunnel using public facilities

- Enhanced network management with simplified local area networks
- Improved productivity and globalization

But since each user has no control over wires and routers, one of the issues with the Internet is still its lack of security, especially when a tunnel is exposed to the public. Thus, VPNs remain susceptible to security issues when they try to connect between two private networks using a public resource. The challenge in making a practical VPN, therefore, is finding the best security for it. Before discussing VPN security, we focus on types of VPNs. There are two types of VPNs each determined by its method of tunneling, *remote-access* and *site-to-site*. We will explain these two approaches in the next two sections.

16.1.1 Remote-Access VPN

Remote-access VPN is a user-to-LAN connection that an organization uses to connect its users to a private network from various remote locations. Large remote-access VPNs are normally outsourced to an Internet service provider to set up a *network-access server*. Other users, working off campus, can then reach the network-access server and use the VPN software to access the corporate network. Remote-access VPNs allow encrypted connections between an organization's private network and remote users through a third-party service provider.

Tunneling in a remote-access VPN uses mainly the *Point-to-Point Protocol* (PPP). PPP is the carrier for other Internet protocols when communicating over the network between a host computer and a remote point. Besides IPsec, other types of protocols associated with PPP are L2F, PPTP, and L2TP. The *Layer 2 Forwarding* (L2F) protocol uses the authentication scheme supported by PPP. The *Point-to-Point Tunneling Protocol* (PPTP) supports 40-bit and 128-bit encryption and uses the authentication scheme supported by PPP. The *Layer 2 Tunneling Protocol* (L2TP) combines features of both PPTP and L2F.

16.1.2 Site-to-Site VPN

By using effective security techniques, an organization can connect multiple fixed sites over a public network. *Site-to-site* VPNs can be classified as either intranets or extranets.

- *Intranet* VPNs connect an organization's remote-site LANs into a single private network.
- *Extranet* VPNs allow two organizations to work in a shared environment through a tunnel built to connect their LANs.

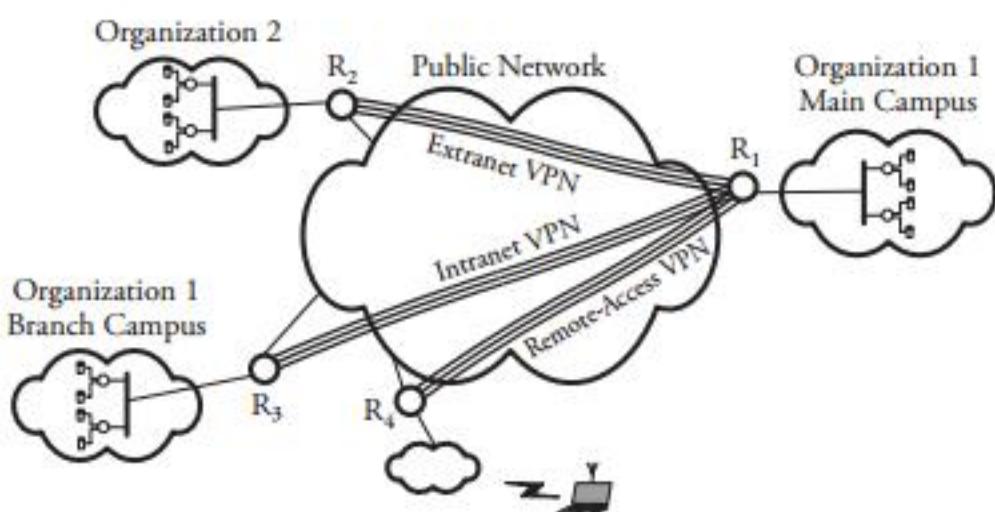


Figure 16.2 Three types of VPNs to and from a headquarter organization

Figure 16.2 shows the three types VPNs discussed so far. Organization 1's main campus and branch campus are connected through an intranet VPN tunnel. The main campus can also be connected to organization 2 through an extranet VPN tunnel. The employees of organization 1 can also access their corporation through a remote-access VPN. Each remote-access member must communicate in a secure medium. The main benefit of using a VPN is *scalability* with a reasonable cost. However, the physical and virtual distances of two communicating organizations have a great impact on the overall cost of building a VPN.

In a site-to-site VPN, *generic routing encapsulation* (GRE) is normally the encapsulating protocol. GRE provides the framework for the encapsulation over an IP-based protocol. IPsec in tunnel mode is sometimes used as the encapsulating protocol. IPsec works well on both remote-access and site-to-site VPNs but must be supported at both tunnel interfaces. The *Layer 2 Tunneling Protocol* (L2TP) can be used in site-to-site VPNs. L2TP fully supports IPsec regulations and can be used as a tunneling protocol for remote-access VPNs.

16.1.3 Tunneling and Point-to-Point Protocol (PPP)

A *tunnel* is a connection that forms a virtual network on top of a physical network. In computer networking, a tunnel resembles a telephone line in a public switched telephone network. VPNs typically rely on tunneling to create a private network that reaches across a public network. Tunneling is a process of encapsulating packets and sending them over the public network. Employees who are located outside an organization's main building can use point-to-point connections to create tunnels

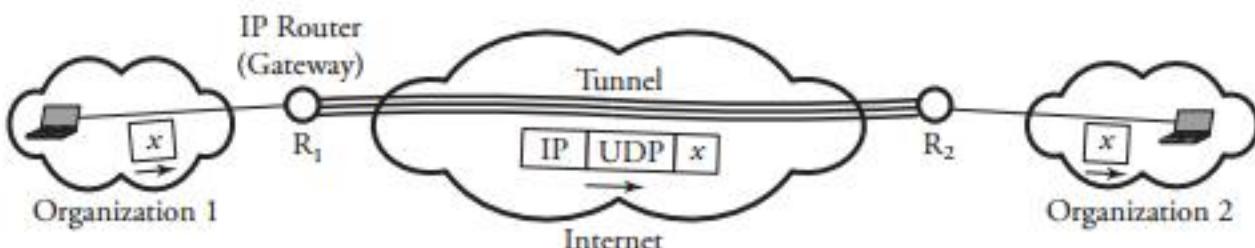


Figure 16.3 A customized protocol packet tunneling through the Internet

through the Internet. Since tunneling connections normally run over the Internet, they need to be secure. A tunnel is a relatively inexpensive connection, since it uses the Internet as its primary form of communication. Besides Internet protocols, tunneling requires two other types of protocols:

1. *Carrier protocols*, through which information travels over the public network
2. *Encapsulating protocols*, through which data is wrapped, encapsulated, and secured

One of the amazing implications of VPNs is that packets that use a protocol not supported on the Internet, such as NetBeui, can be placed inside an IP packet and sent safely over the Internet. VPNs can put a packet that uses a nonroutable IP address inside a packet to extend a private network over the Internet.

Consider the two LANs of the two organizations shown in Figure 16.3. We want to connect these two LANs through the Internet by using tunnels. Assume that the two LANs, as organization 1 and organization 2, want to use their own customized networking protocols, denoted by *x*, using connectionless datagram IP services. The IP resources can be at the scale of the Internet. Therefore, *x*-type packets cannot run over the Internet directly. The IP gateway R₁ listens for *x*-type packets on organization 1, encapsulates *x*-type packets in the transport-layer UDP datagrams, and transmits them over the Internet to R₂. When R₂ receives the encapsulated *x* packets, it decapsulates and feeds them into organization 2. This connection—in fact, a tunnel made through the Internet—resembles a direct physical link between the two LANs.

Point-to-Point Protocol (PPP)

The basic notion in tunneling is packet encapsulation from one protocol into the same or higher-layer protocol. Thus, a tunnel can also be defined as an encapsulating protocol for protocols at the lower layers. Tunneling protocols, such as the *Point-to-Point Protocol* (PPP) or the *Point-to-Point Tunneling Protocol* (PPTP) are encapsulating protocols that allow an organization to establish secure connections from one point to another while

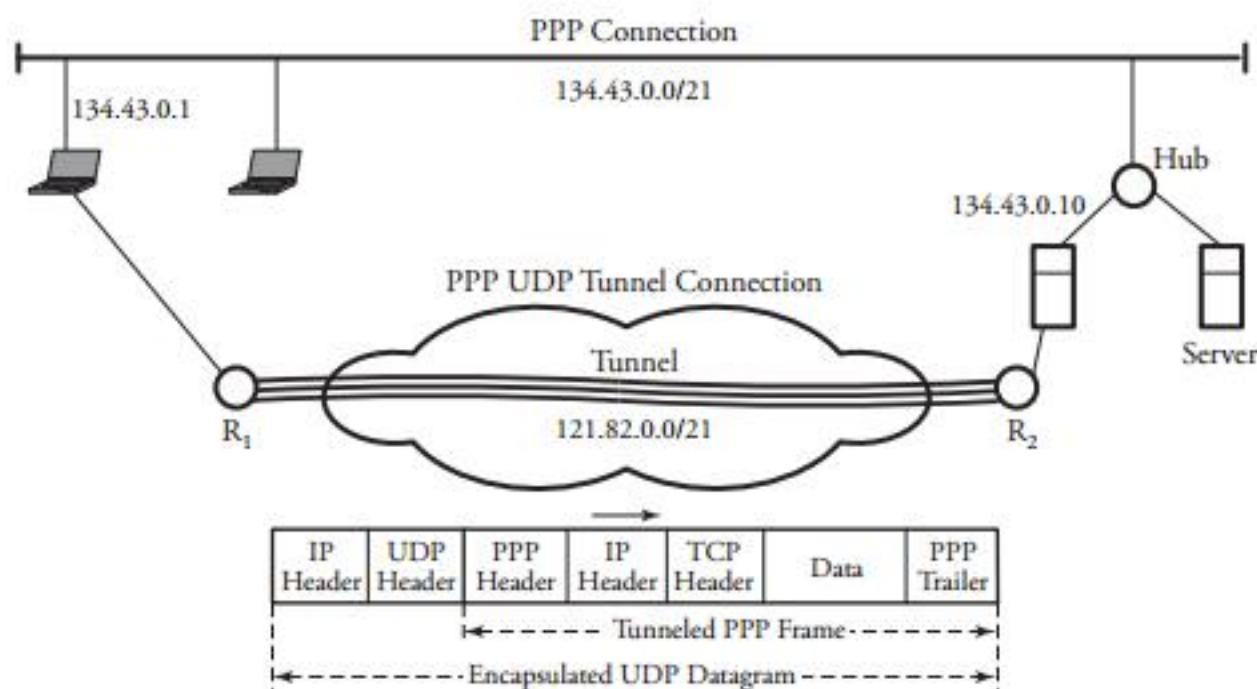


Figure 16.4 A point-to-point protocol (PPP) UDP tunnel connection

using public resources. A PPP connection is a serial connection between a user and an Internet service provider.

Example. In Figure 16.4, a point-to-point protocol (PPP) UDP tunnel connection is established while another virtual PPP connection exists. In this scenario, a user at 134.43..0.1 is communicating with a server at 134,43.0.10. These two end points are connected through their own PPP connection of 134.43.0.0/21, but the transmitted data flows through the tunnel on the 121.82.0.0/21 segment. This tunnel is established at an interface layer in a UDP transport-layer protocol as it appears from the frame format in the figure. Tunneling can also be formed at network- and transport-layer protocols, where equal layers are involved, such as IP-in-IP tunnels.

16.1.4 Security in VPNs

Without using dedicated hardware, a VPN uses virtual connections routed through the Internet from the company's private network to the remote site. Companies can create their own VPNs to accommodate the needs of remote employees and distant offices. This section looks at methods for keeping VPN connections secure. A well-protected VPN uses firewalls, encryption systems, IPsec features, and an authentication server.

A firewall provides an effective barrier between a private network and the Internet. Firewalls can be set up to restrict the number of open ports to monitor what types of

packets are passed through and which protocols are allowed through. The authentication servers performs authentication, authorization, and accounting for more secure access in a remote-access environment. When a request to establish a session comes in, the request is loaded onto this server. The server then checks who the sender is (authentication), what it is allowed to do (authorization), and what it actually does (accounting and bills).

16.2 Multiprotocol Label Switching (MPLS)

Multiprotocol label switching (MPLS) improves the overall performance and delay characteristics of the Internet. MPLS transmission is a special case of tunneling and is an efficient routing mechanism. Its connection-oriented forwarding mechanism, together with layer 2 label-based lookups, enables *traffic engineering* to implement peer-to-peer VPNs effectively.

MPLS adds some traditional layer 2 capabilities and services, such as traffic engineering, to the IP layer. The separation of the MPLS control and forwarding components has led to multilayer, multiprotocol interoperability between layer 2 and layer 3 protocols. MPLS uses a small label or stack of labels appended to packets and typically makes efficient routing decisions. Another benefit is flexibility in merging IP-based networks with fast-switching capabilities. This technology adds new capabilities to IP-based networks:

- Connection-oriented QoS support
- Traffic engineering
- VPN support
- Multiprotocol support

Traditional IP routing has several limitations, ranging from scalability issues to poor support for traffic engineering. The IP backbone also presents a poor integration with layer 2 existing in large service provider networks. For example, a VPN must use a service provider's IP network and build a private network and run its own traffic shielded from prying eyes. In this case, VPN membership may not be well engineered in ordinary IP networks and can therefore result in an inefficient establishment of tunnels.

MPLS network architectures also support other applications, such as IP multicast routing and QoS extensions. The power of MPLS lies in the number of applications made possible with simple label switching, ranging from traffic engineering to peer-to-peer VPNs. One of the major advantages of MPLS is integration of the routing and

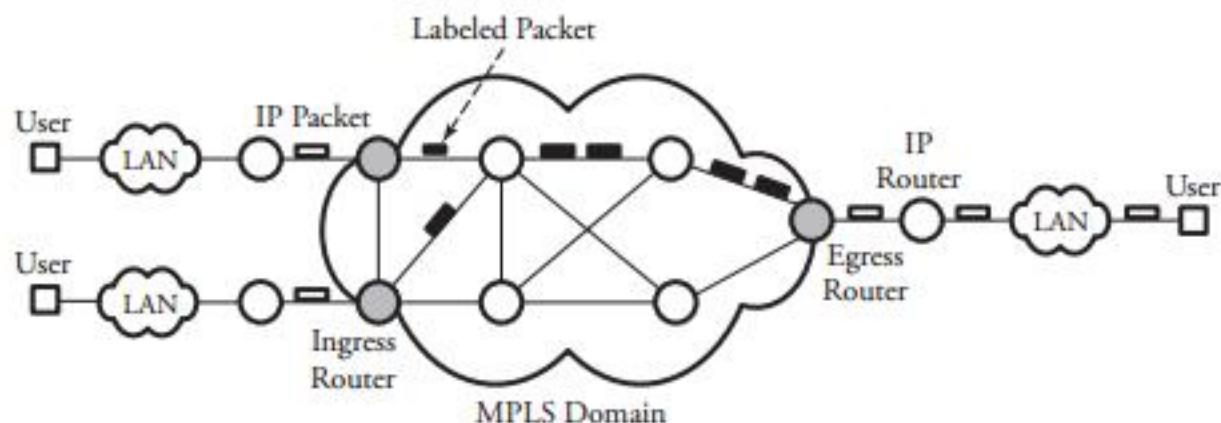


Figure 16.5 An MPLS network

switching layers. The development of the label-switched protocol running over all the existing layer 2 and layer 3 architectures is a major networking development.

16.2.1 MPLS Operation

MPLS is based on the assignment of *labels* to packets. Assigning labels to each packet makes a label-swapping scheme perform its routing process much more efficiently. An MPLS network consists of nodes called *label switch routers* (LSR). An LSR switches labeled packets according to particular switching tables. An LSR has two distinct functional components: a control component and a forwarding component. The control component uses routing protocols, such as OSPF and the *border gateway protocol* (BGP). The control component also facilitates the exchange of information with other LSRs to build and maintain the forwarding table.

A label is a header used by an LSR to forward packets. The header format depends on the network characteristics. LSRs read only labels and do not engage in the network-layer packet headers. One key to the scalability of MPLS is that labels have only local significance between two devices that communicate. When a packet arrives, the forwarding component uses the label of the packet as an index to search the forwarding table for a match. The forwarding component then directs the packet from the input interface to the output interface through the switching fabric.

MPLS Packet Format

MPLS uses *label stacking* to become capable of multilevel hierarchical routing. A label enables the network to perform faster by using smaller forwarding tables, a property that ensures a convenient scalability of the network. Figure 16.6 shows the MPLS header

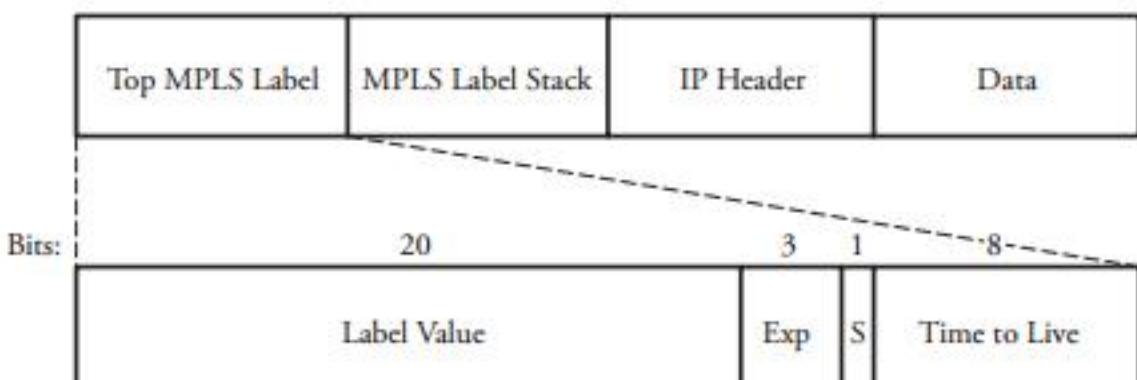


Figure 16.6 MPLS header encapsulation for an IP packet

encapsulation for an IP packet. An MPLS label is a 32-bit field consisting of several fields as follows.

- *Label value* is a 20-bit field label and is significant only locally.
- *Exp* is a 3-bit field reserved for future experimental use.
- *S* is set to 1 for the oldest entry in the stack and to 0 for all other entries.
- *Time to live* is an 8-bit field used to encode a hop-count value to prevent packets from looping forever in the network.

16.2.2 Routing in MPLS Domains

Figure 16.7 shows the label-switching paradigm in an MPLS network. An *ingress LSR* is an edge device that performs the initial packet processing and classification and applies the first label. An ingress LSR creates a new label. A *core LSR* swaps the incoming label with a corresponding next-hop label found from a forwarding table. At the other end of the network, another edge router, the *egress LSR*, is an outbound edge router and pops the label from the packet. It should be noted that multiple labels may be attached to a packet, forming a stack of labels. Label stacking enables multilevel hierarchical routing. For example, BGP labels are used for higher-level hierarchical packet forwarding from one BGP speaker to the other, whereas Interior Gateway Protocol (IGP) labels are used for packet forwarding within an autonomous system. Only the label at the top of the stack determines the forwarding decision.

Once an IP packet enters an MPLS domain, the ingress LSR processes its header information and maps that packet to a *forward equivalence class* (FEC). At this point, a *label switch path* (LSP) through the network must be defined, and the QoS parameters along that path must be established. The QoS parameters define how many resources

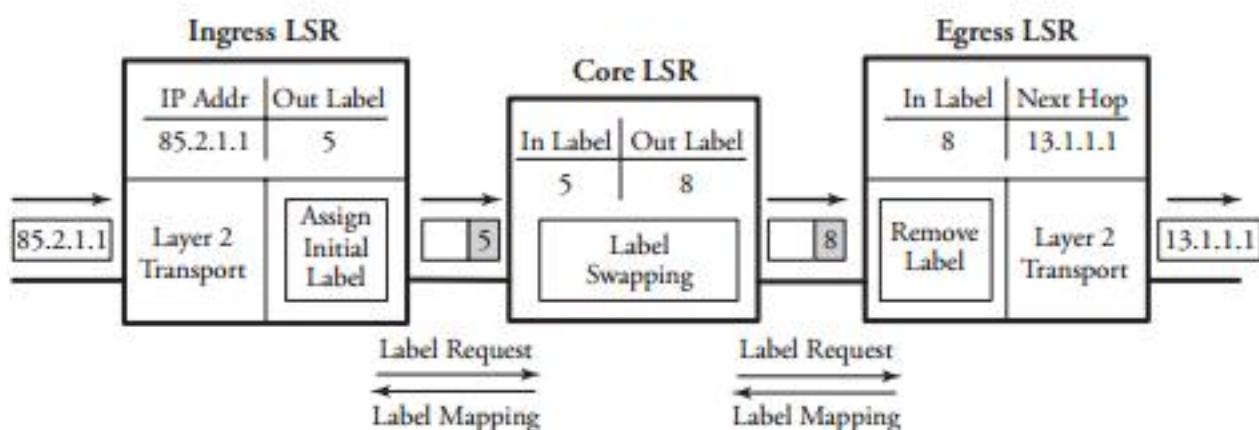


Figure 16.7 Multiple layer 2 switching example in MPLS

are to be used for the path and what queueing and discarding policy are to be used. For these functions, two protocols are used to exchange necessary information among routers: An intradomain routing protocol, such as OSPF, is used to exchange routing information, and the *Label Distribution Protocol* (LDP) assigns labels. At the end of the process, the router appends an appropriate label for FEC purposes and forwards the packet through.

Example. As shown in Figure 16.5, an IP packet with the destination address 85.2.1.1 enters an MPLS domain, and the ingress LSR processes its header and assigns a label, numbered 5. The label swapping from 5 to 8 takes place in the core router; finally, the next-hop IP address is attached to the packet on its way out of the egress router.

Packet forwarding at the core LSR is based on a label-swapping mechanism. Once it receives a labeled packet, the core LSR reads the label as an index to search in the *incoming label map table* for the corresponding next-hop label. The label in the MPLS header is swapped with the out-label and sent on the next hop. This method of packet forwarding simplifies the routing process by replacing the longest-prefix match of IP routing with simple short-label exact-match forwarding. The real benefit of this method is that instead of processing IP headers for forwarding packets, routers process a short label. Once a packet arrives at the egress LSR, its MPLS header is decapsulated, and the stripped packet is routed to its destination.

In summary, an MPLS domain has three label manipulation instructions: An *ingress* LSR creates a new label and pushes it to the label stack of a packet, a *core* LSR swaps the incoming label with a corresponding next-hop label found from the forwarding table, and an *egress* LSR (outbound edge router) pops a label from the label stack. Only the

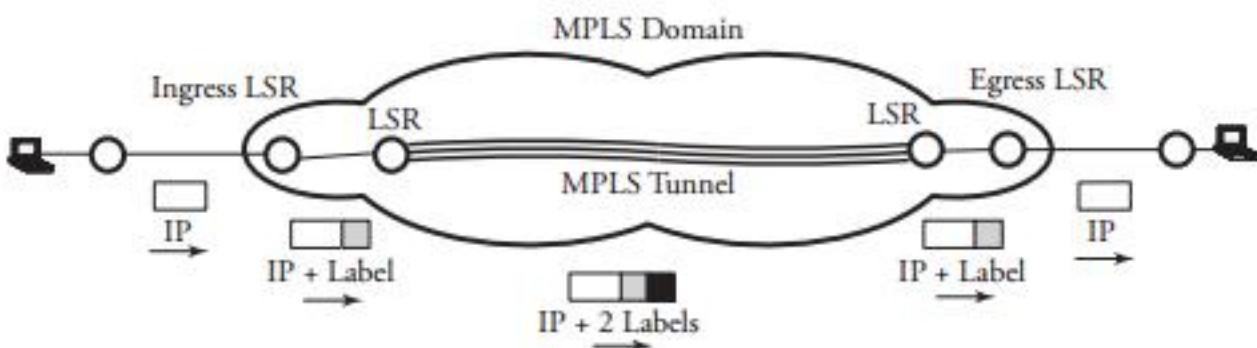


Figure 16.8 An IP packet labeled in an MPLS domain and tunneled to reach the other end of the domain

label at the top of the stack determines the forwarding decision. The egress LSR strips the label, reads the IP packet header, and forwards the packet to its final destination.

16.2.3 Tunneling and Use of FEC

In an MPLS operation, any traffic is grouped into FECs. FEC implies that a group of IP packets are forwarded in the same manner—for example, over the same path or with the same forwarding treatment. A packet can be mapped to a particular FEC, based on the following criteria:

- Source and/or destination IP address or IP network addresses
- TCP/UDP port numbers
- Class of service
- Applications

As mentioned earlier, labels have only local significance. This fact removes a considerable amount of the network-management burden. An MPLS packet may carry as many labels as required by a network sender. The process of labeled packets can always be performed based on the top label. The feature of label stack allows the aggregation of LSPs into a single LSP for a portion of the route, creating an **MPLS tunnel**. Figure 16.8 shows an IP packet moving through an MPLS domain. When the labeled packet reaches the ingress LSR, each incoming IP packet is analyzed and classified into different FECs. This traffic-classification scheme provides the capability to partition the traffic for service differentiation.

Route selection can be done either hop by hop or by *explicit routing*. With hop-by-hop routing, each LSR can independently choose the next hop for each FEC. Hop-by-hop routing does not support traffic engineering, owing to limited available resources.

Explicit routing can provide all the benefits of traffic engineering. With explicit routing, a single LSR determines the LSP for a given FEC. For explicit routing, LSRs in the LSP are identified, whereas in an explicit routing, only some of the LSRs in an LSP are specified.

With the introduction of constraint-based routing, FEC can segregate the traffic into different levels of QoS, each with different service constraints, to support a variety of services, such as latency-based voice traffic and security-based VPN. At the beginning of the tunnel, an LSR assigns the same label to packets from a number of LSPs by pushing the label onto each packet's stack. At the other side of the tunnel, another LSR pops the top element from the label stack, revealing the inner label.

Label Distribution Protocol (LDP)

The *Label Distribution Protocol* (LDP) is a set of rules by which an LSR informs another LSR of an FEC. LDP enables two LSRs to understand each other's MPLS capabilities. LSP schemes are either *downstream on demand* or *downstream unsolicited*. With the downstream-on-demand scheme, an upstream node explicitly requests a label from a downstream node, and the downstream node forms the requested label. With the downstream-unsolicited scheme, a downstream node advertises a label mapping even without receiving any advance requests. Both types of LDPs can be used in explicit and hop-by-hop routing; however, a simple LDP can function using the routing protocol, such as OSPF, to design routes, since hop-by-hop routing does not follow the traffic engineering.

16.2.4 Traffic Engineering

High-quality connections can be expensive in an Internet service provider domain. *Traffic engineering* enables an ISP to route high-quality traffic to offer the best service to users in terms of throughput and delay. This way, traffic engineering reduces the cost of a network connection. Traffic engineering substitutes the need to manually configure network devices to set up explicit routes. In MPLS, traffic engineering is an automated scheme for control signaling and link bandwidth assignment and has a dynamic adaptation mechanism.

Traffic engineering can be either *traffic oriented* or *resource oriented*. Traffic-oriented traffic engineering relates to the optimization of such traffic performance parameters as the minimization of packet loss and delay and quick fault recovery when a node or a link fails. The resource-oriented technique engages in the optimization of network resource utilization.

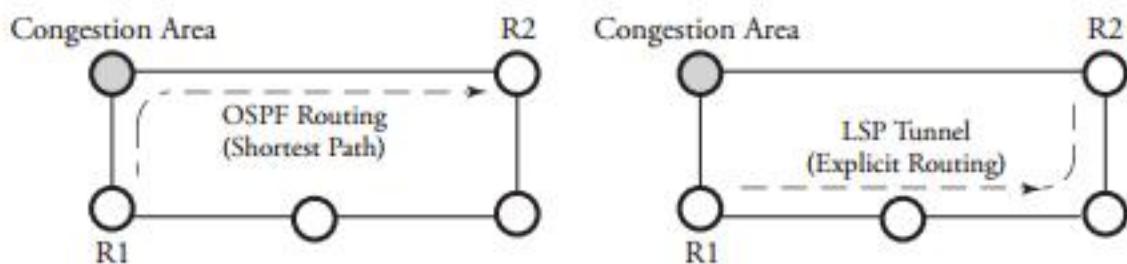


Figure 16.9 A traffic engineering scenario

Example. Figure 16.9 shows an example of traffic engineering in MPLS. Assume that router R1 has a packet to send to R2. OSPF routes the packet through the shortest path, regardless of whether R3 is experiencing congestion. In an MPLS network, an LSP can be set up explicitly to avoid the congested node; if a constraint-based routing algorithm is used, an LSP avoiding the congested node is set up dynamically, even though the routing path is longer. This path-management capability is very appealing for traffic engineering.

Example. Figure 16.10 shows an example of traffic engineering in an MPLS network. The routing table is shown for router R2, where R2 advertises to R3, R5, and R6 that it can route to all three destinations C, D, and E. Hence, any frame with labels 31, 4, and 21, respectively, are switched toward these destinations.

16.2.5 MPLS-Based VPNs

Routine operations of virtual private networks require the use of both wide-area intradomain routing and interdomain routing schemes. A VPN's request to form a tunnel can be processed at the edge routers. For example, multiprotocol-based Border Gateway Protocol (BGP) makes MPLS-based VPN easier to manage VPN sites and VPN membership, mainly owing to the traffic engineering feature of MPLS. In an MPLS network, VPNs can be deployed by delivering the service using MPLS-aware subscriber equipment on the same infrastructure used for deploying Internet services.

An MPLS network domain acts as a backbone network between VPN users. Also, core LSRs act as *providing routers*, and edge routers act as *customer edge routers*. Customer edge routers distribute VPN information through MPLS-BGP to other providing routers. In order to forward an IP packet encapsulated for VPN through an MPLS backbone, the top label of the MPLS label stack is used to indicate the outgoing interface, and the second-level label is used to indicate the BGP next hop. When it receives a normal encapsulated IP packet from a router, an ingress customer edge router

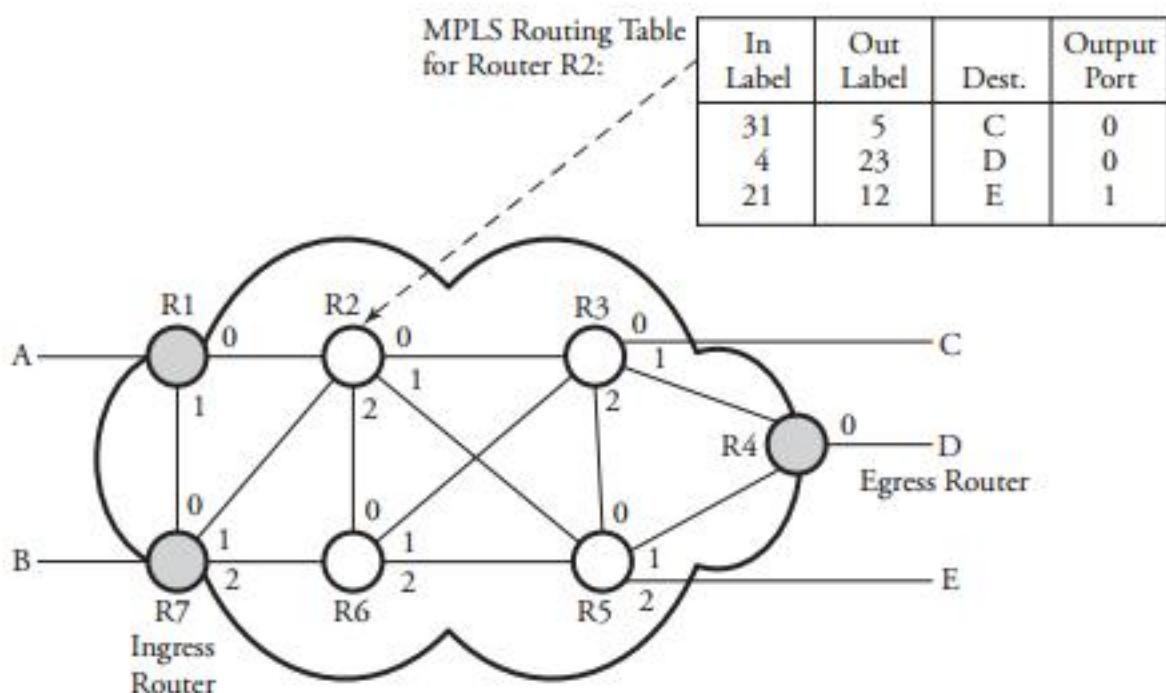


Figure 16.10 Layer 3 routing in an MPLS network

performs an “IP longest match” and finds the next hop corresponding to the packet’s home VPN. The second-to-last MPLS router forwards the packet and pops the top label so that the customer edge router can forward the packet, based on the second-level label, which gives the VPN.

16.3 Overlay Networks

An *overlay network* is an application-specific computer network built on top of another network. In other words, an overlay network creates a virtual topology on top of the physical topology. This type of network is created to protect the existing network structure from new protocols whose testing phases require Internet use. Such networks protect packets under test while isolating them from the main networking infrastructure in a test bed.

Figure 16.11 shows an overlay network configured over a wide area network. Nodes in an overlay network can be thought of as being connected by logical links. In Figure 16.11, for example, routers R₄, R₅, R₆, and R₁ are participating in creating an overlay network where the interconnection links are realized as overlay logical links. Such a logical link corresponds to a path in the underlying network. An obvious example of these networks is the *peer-to-peer network*, which runs on top of the Internet. Overlay networks have no control over how packets are routed in the underlying network

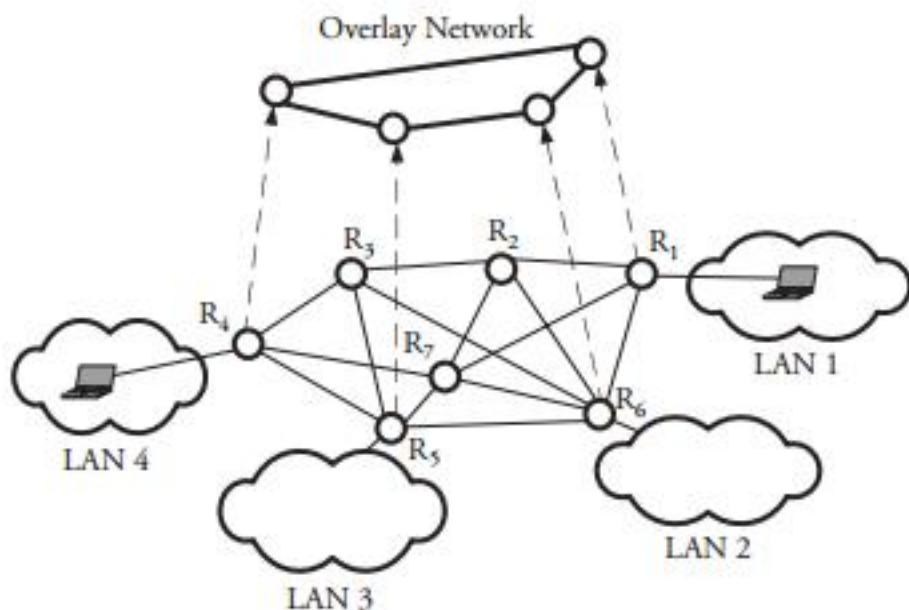


Figure 16.11 An overlay network for connections between two LANs associated with routers R_1 and R_4

between a pair of overlay source/destination nodes. However, these networks can control a sequence of overlay nodes through a message-passing function before reaching the destination.

For various reasons, an overlay network might be needed in a communication system. An overlay network permits routing messages to destinations when the IP address is not known in advance. Sometimes, an overlay network is proposed as a method to improve Internet routing as implemented in order to achieve higher-quality streaming media. Sometimes, for the implementation of such techniques as DiffServ and IP multicast, modification of all routers in the network is required. In such cases, an overlay network can be deployed on end hosts running the overlay protocol software, without cooperation from Internet service providers.

Overlay networks are *self-organized*. When a node fails, the overlay network algorithm should provide solutions that let the network recover and recreate an appropriate network structure. Another fundamental difference between an overlay network and an unstructured network is that overlays' look-up routing information is on the basis of identifiers derived from the content of moving frames.

16.3.1 Peer-to-Peer (P2P) Connection

As an overlay network resembles a system consisting of various applications running on a single operating system, it could also resemble a set of tunnels that interconnect resources and users. The interconnects are carried out by *peer-to-peer* (P2P) protocols.

Let δ be the time required to establish a connection and t_f be the time to finish the service as soon as the connection establishes. Assuming that the requests arrive at random to a peer node, the service time s is

$$s = \begin{cases} t_f & \text{if the peer is connected} \\ t_f + \delta & \text{if the peer is not connected} \end{cases}. \quad (16.1)$$

When a request arrives, the peer can be connected with probability p_c and not connected with probability $1 - p_c$. Realizing that s is a continuous random variable (see Appendix C) and is discrete for its two cases, discussed in Equation (16.1), the expected value of average service time is derived by

$$E[S] = (1 - p_c)t_f + p_c(t_f + \delta) = t_f + p_c\delta. \quad (16.2)$$

Let $\rho = \frac{\lambda}{\mu}$ be the utilization of the peer node, where λ is the request arrival rate, and μ is the average service rate. Thus, a fraction ρ of any given time that either “the peer uses for connecting” or “the connection is used for service” is expressed by

$$u_s = \rho \left(\frac{(1 - p_c)\delta}{E[S]} \right). \quad (16.3)$$

Similarly, the fraction $1 - \rho$ that the same mentioned given time is idle can be derived by

$$u_i = (1 - \rho)p_c, \quad (16.4)$$

where the idle time occurs when the peer is either disconnected or connected but not using the connection. We can now derive an expression for the peer connection efficiency, u , as follows:

$$u = 1 - (u_s + u_i). \quad (16.5)$$

The connection efficiency of the peer can be used to determine the overall efficiency of the P2P connection.

16.4 Summary

Our discussion of *tunneling* issues began with *virtual private networks* (VPNs), which are virtually used by a private-sector entity over a public network. Tunneling is an encapsulation of a packet data segment to move from one protocol to another protocol at the same or higher layer. An organization using a VPN uses a service provider's IP network and builds a private network and runs its own traffic.

Tunneling has two forms: *remote-access* tunneling, which is a user-to-LAN connection, and *site-to-site* tunneling, whereby an organization can connect multiple fixed sites over a public network. Employees who are located beyond their main campus can use *point-to-point* (PPP) connections to create tunnels through the Internet into the organization's resources. Both PPTP and L2TP depend on PPP to frame tunneled packets.

Multiprotocol label switching (MPLS) improves the overall performance of traditional IP routing, especially for the establishment of more effective VPNs. In MPLS, multiple labels can be combined in a packet to form a header used by an LSR for efficient tunneling. The *Label Distribution Protocol* (LDP) is a set of rules by which an LSR informs another LSR. The MPLS *traffic engineering* feature is an automated scheme for a control-signaling process and link-bandwidth assignment to improve the quality of network management.

Overlay networks create a virtual topology on top of an existing physical topology on a public network. Overlay networks are *self-organized*; thus, if a node fails, the overlay network algorithm can provide solutions that let the network recreate an appropriate network structure.

The next chapter starts a new topic. We focus on multimedia networking, starting with the introductory concept of *compression of digital voice and video*.

16.5 Exercises

1. Label routing in MPLS networks is similar to VPN tunneling. Compare these two schemes in terms of
 - (a) Traffic engineering capability
 - (b) Security
2. Traffic engineering in MPLS networks can be done in the following two locations. Compare the advantages of each approach.
 - (a) Egress nodes estimate routing to and from all ingress nodes.
 - (b) A preassigned router estimates routing and propagates to all LSRs.
3. Consider the MPLS network shown in Figure 16.10; suppose that we want to use traffic engineering. Assume that the table of egress router R4 contains 26 (In Label), (Out Label), D (dest.), and 0 (Output Port). We want to route packets received at the ingress router R7 to egress router R4 through R1-R2-R5-R4.
 - (a) Show the MPLS routing table for R1.
 - (b) Show the MPLS routing table for R3.

4. Consider an overlay network that consists of five connected nodes. Compare two methods for a peer-to-peer connection on top of the Internet, using
 - (a) A ring topology
 - (b) A star topology
5. Suppose that two peers are connected to a four-node overlay network interconnected with a ring topology, as shown in Figure 16.11. There are 200 requests per second arriving at one of the peers, and the average service rate is 230 requests per second. Assuming that the time required to establish a connection is 10 ms, that the time to finish the service as soon as the connection establishes is 120 ms, and that there is an equal chance that the peer is connected or not connected:
 - (a) Find the average service time of the peer.
 - (b) Find the total utilization of the peer.
6. *Computer simulation project.* Carry the computer program you developed for the simulation of the seven-node network in Chapter 7 and add a subprogram to simulate the operation of overlay networks. To do this, use the computer program, and construct the seven-router network shown in Figure 16.11.
 - (a) Simulate a packet transfer through the original network from R_1 to R_4 .
 - (b) Compare the results of part (a) to the case, using the overlay network shown in the figure.

CHAPTER 17

Compression of Digital Voice and Video

This chapter looks at a new advanced topic: *multimedia networking*. This important topic requires a thorough understanding of the process of preparing compressed voice and video. A raw piece of information, whether voice or video, must be converted to digital form and then compressed to save link bandwidth. This chapter discusses methods of preparing digital voice and video for multimedia networks, including the analysis of information sources, source coding, and limits of data compression. Typical voice and video streaming compression techniques, such as JPEG, MPEG, and MP3, are explained. The major topics of this chapter are

- *Overview of data compression*
- *Digital voice and compression*
- *Still images and JPEG compression*
- *Moving images and MPEG compression*
- *Limits of compression with loss*
- *Compression process without loss*
- *Case study: FAX compression for transmission*

Our discussion starts with an overview of data preparation and compression, focusing on voice, image, and moving-image data. Considerable effort is required to turn analog voice into digital form. This process of converting from raw voice to

compressed binary form—sampling, quantization, and encoding—is known as *pulse code modulation* (PCM).

We then take a quick look at some practical compression methods without loss. We review the compression schemes that are used after the binary voice or image is prepared. We discuss both JPEG and MPEG compression techniques for still images and moving images, respectively. Still images or moving images might contain redundant or repeated elements that can be eliminated and codes substituted for future decoding process. We also summarize Shannon's limits of compression with loss. At the end of the discussion on compression, we review lossless compression techniques, such as *Huffman encoding* and *run-length encoding*. The chapter ends with a case study on FAX compression for transmission.

17.1 Overview of Data Compression

The benefits of data compression in high-speed networks are obvious. Following are those that are especially important for the compressed version of data.

- Less transmission power is required.
- Less communication bandwidth is required.
- System efficiency is increased.

There are, however, certain trade-offs with data compression. For example, the encoding and decoding processes of data compression increase the cost, complexity, and delay of data transmission. Both of the two processes of data compression are required for producing multimedia networking information: *compression with loss* and *compression without loss*.

In the first category of data compression, some less valuable or almost similar data must be eliminated permanently. The most notable case of compression with loss is the process of signal sampling. In this category, for example, is voice sampling (Section 17.2). With data compression without data loss, the compressed data can be recovered and converted back to its original form when received. This method of compression is typically applied to digital bits after sampling.

Figure 17.1 shows the basic information process in high-speed communication systems. Any type of “source” data is converted to digital form in a long *information-source* process. The outcome is the generation of digital words. Words are encoded in the *source coding* system to result in a compressed form of the data.

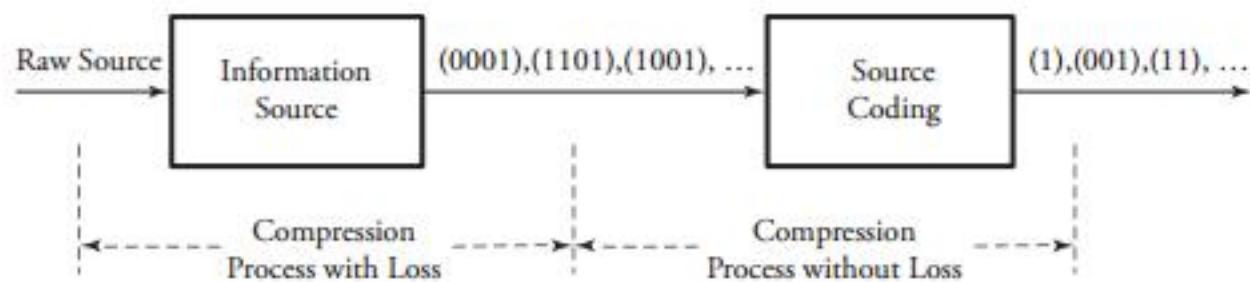


Figure 17.1 Overview of information process and compression in multimedia networks

17.2 Digital Voice and Compression

Our discussion starts with the voice as a simple real-time signal. We first review the fundamentals of voice digitization and sampling.

17.2.1 Signal Sampling

In the process of digitalizing a signal, analog signals first go through a *sampling process*, as shown in Figure 17.2. The sampling function is required in the process of converting an analog signal to digital bits. However, acquiring samples from an analog signal and eliminating the unsampled portions of the signal may result in some permanent loss of information. In other words, the sampling resembles an *information-compression* process with loss.

Sampling techniques are of several types:

- *Pulse amplitude modulation* (PAM), which translates sampled values to pulses with corresponding amplitudes
- *Pulse width modulation* (PWM), which translates sampled values to pulses with corresponding widths
- *Pulse position modulation* (PPM), which translates sampled values to identical pulses but with corresponding positions to sampling points

PAM is a practical and commonly used sampling method; PPM is the best modulation technique but is expensive. PWM is normally used in analog remote-control systems. The sampling rate in any of these schemes obeys the *Nyquist theorem*, according to which at least two samples on all components of the spectrum are needed in order to reconstruct a spectrum:

$$f_S \geq 2f_H, \quad (17.1)$$

where f_H is the highest-frequency component of a signal, and f_S is the sampling rate.

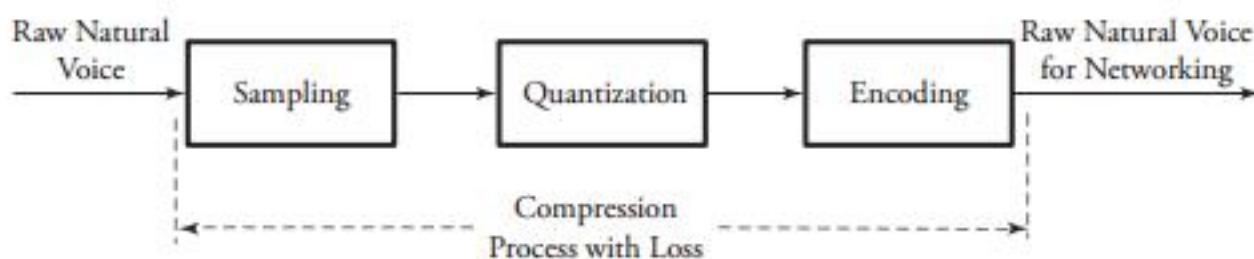


Figure 17.2 Overview of digital voice process

17.2.2 Quantization and Distortion

Samples are real numbers—decimal-point values and integer values—and, thus, up to infinite bits are required for transmission of a raw sample. The transmission of infinite bits occupies infinite bandwidth and is not practical for implementation. In practice, sampled values are rounded off to available quantized levels. However, rounding off the values loses data and generates *distortion*. A measure is needed to analyze this distortion. The distortion measure should show how far apart a signal denoted by $x(t)$ is to its reproduced version, denoted by $\hat{x}(t)$. The distortion measure of a single source is the difference between source sample X_i and its corresponding quantized value \hat{X}_i , denoted by $d(X, \hat{X})$, and is widely known as *squared-error distortion*:

$$d(X, \hat{X}) = (x - \hat{x})^2. \quad (17.2)$$

Note that \hat{X}_i is noninvertible, since lost information cannot be recovered. The distortion measure of n samples is based on the values of source samples obtained at the sampler output. As a result, the collection of n samples forms a random process:

$$X_n = \{X_1, X_2, \dots, X_n\}. \quad (17.3)$$

Similarly, the reconstructed signal at the receiver can be viewed as a random process:

$$\hat{X}_n = \{\hat{X}_1, \hat{X}_2, \dots, \hat{X}_n\}. \quad (17.4)$$

The distortion between these two sequences is the average between their components:

$$d(X_n, \hat{X}_n) = \frac{1}{n} \sum_{i=1}^n d(X_i, \hat{X}_i). \quad (17.5)$$

Note that $d(\mathbf{X}_n, \hat{\mathbf{X}}_n)$ itself is a random variable, since it takes on random numbers. Thus, the total distortion between the two sequences is defined as the expected value of $d(\mathbf{X}_n, \hat{\mathbf{X}}_n)$:

$$\begin{aligned} D &= E[d(\mathbf{X}_n, \hat{\mathbf{X}}_n)] = E\left[\frac{1}{n} \sum_{i=1}^n d(X_i, \hat{X}_i)\right] \\ &= \frac{1}{n} E[d(X_1, \hat{X}_1) + d(X_2, \hat{X}_2) + \cdots + d(X_n, \hat{X}_n)]. \end{aligned} \quad (17.6)$$

If all samples are expected to have approximately the same distortion denoted by $d(X, \hat{X})$, $d(X_1, \hat{X}_1) = d(X_2, \hat{X}_2) = \cdots = d(X_n, \hat{X}_n) = d(X, \hat{X})$. By using squared-error distortion, we obtain the total distortion:

$$D = \frac{1}{n} \left(n E[d(X, \hat{X})] \right) = E[d(X, \hat{X})] = E[(X - \hat{X})^2]. \quad (17.7)$$

Let R be the minimum number of bits required to reproduce a source and guarantee that the distortion be less than a certain distortion bound D_b . Clearly, if D decreases, R must increase. If X is represented by R bits, the total number of different values X_i takes is 2^R . Each single-source output is quantized into N levels. Each level $1, 2, \dots, N$ is encoded into a binary sequence. Let \mathfrak{N} be the set of real numbers $\mathfrak{N}_1, \dots, \mathfrak{N}_k, \dots, \mathfrak{N}_N$, and let \hat{X}_k be the quantized value belonging to subset \mathfrak{N}_k . Note that \hat{X}_k is a quantized version of X_k . Apparently, $R = \log_2 N$ bits are required to encode N quantized levels. Figure 17.3 shows a model of N -level quantization: For the subsets $\mathfrak{N}_1 = [-\infty, a_1]$, $\mathfrak{N}_2 = [a_1, a_2]$, \dots , $\mathfrak{N}_8 = [a_{N-1}, \infty]$, the quantized values are $\hat{X}_1, \hat{X}_2, \dots, \hat{X}_n$, respectively. We can use the definition of expected value to obtain D , as follows:

$$D = \int_{-\infty}^{+\infty} (x - \hat{x})^2 f_X(x) dx = \sum_{i=1}^N \int_{\mathfrak{N}_i} (x - \hat{x})^2 f_X(x) dx. \quad (17.8)$$

Typically, a distortion bound denoted by D_b is defined by designers to ensure that $D_b \leq D$.

Example. Consider that each sample of a sampled source is a Gaussian random variable with a given probability distribution function $f_X(x) = 0.01e^{-\frac{1}{800}x^2}$. We want eight levels of quantization over the regions $\{a_1 = -60, a_2 = -40, \dots, a_7 = 60\}$ and $\{\hat{x}_1 = -70, \hat{x}_2 = -50, \dots, \hat{x}_8 = 70\}$. Assuming that the distortion bound for this signal is $D_b = 7.2$, find out how far the real distortion, D , is to D_b .

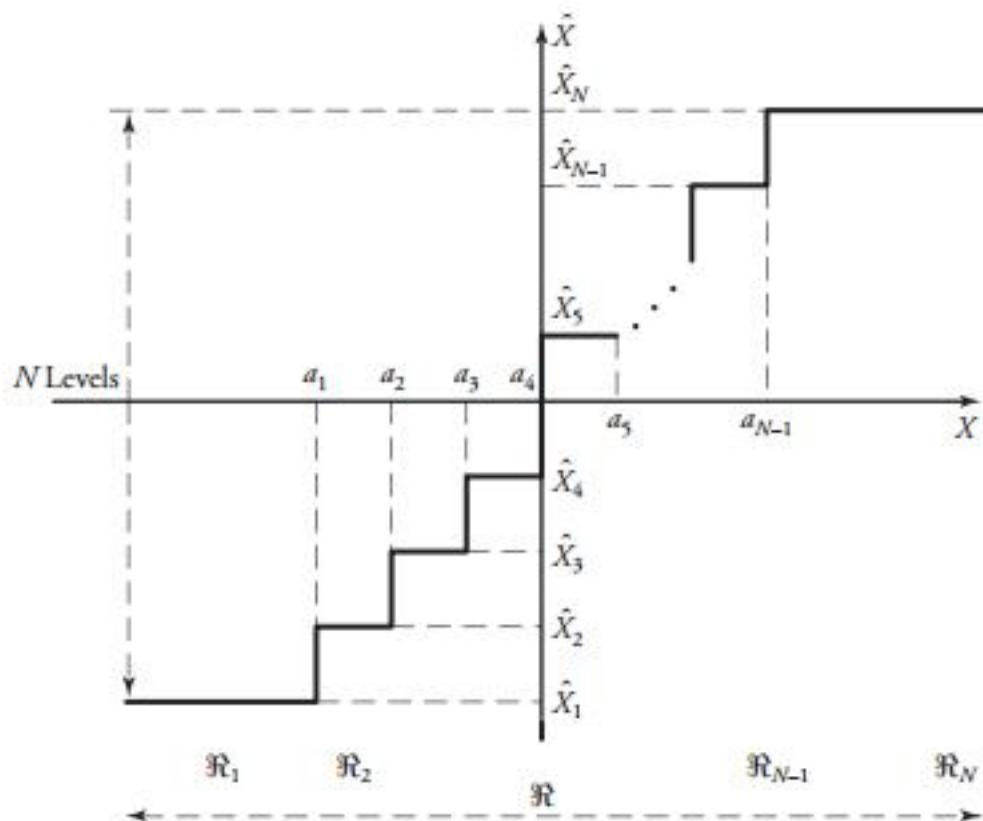


Figure 17.3 N-level quantization

Solution. Since $N = 8$, the number of bits required per sample is $R = \log_2 8 = 3$. Using this, D can be developed further:

$$\begin{aligned}
 D &= \sum_{i=1}^8 \int_{R_i} (X - \hat{X}_i)^2 f_X(x) dx \\
 &= \int_{-\infty}^{a_1} (x - \hat{x}_1)^2 (0.01e^{-\frac{1}{800}x^2}) dx + \sum_{i=2}^7 \int_{a_{i-1}}^{a_i} (x - \hat{x}_i)^2 (0.01e^{-\frac{1}{800}x^2}) dx \\
 &\quad + \int_{a_7}^{\infty} (x - \hat{x}_8)^2 (0.01e^{-\frac{1}{800}x^2}) dx = 16.64.
 \end{aligned}$$

We note here that the total distortion as a result of eight-level quantization is 16.64, which is considerably greater than the given distortion bound of 7.2.

The conclusion in the example implies that the quantization chosen for that source may not be optimal. A possible reason may be inappropriate choices of (a_1, \dots, a_7) and/or $(\hat{x}_1, \dots, \hat{x}_8)$. This in turn means that R is not optimal.

Optimal Quantizers

Let Δ be the length of each region equal to $a_{i+1} - a_i$. Thus, regions can be restated as $(-\infty, a_1) \cdots (a_{N-1}, +\infty)$. Clearly, the limits of the upper region can also be shown by $(a_1 + (N-2)\Delta, +\infty)$. Then, the total distortion can be rewritten as

$$D = \int_{-\infty}^{a_1} (x - \hat{x})^2 f_X(x) dx + \sum_{i=1}^{N-2} \int_{a_1+(i-1)\Delta}^{a_1+i\Delta} (x - \hat{x}_{i+1})^2 f_X(x) dx \\ + \int_{a_1+(N-2)\Delta}^{\infty} (x - \hat{x}_N)^2 f_X(x) dx. \quad (17.9)$$

For D to be optimized, we must have $\frac{\partial D}{\partial a_1} = 0$, $\frac{\partial D}{\partial \Delta} = 0$ and also $\frac{\partial D}{\partial \hat{x}_1} = 0, \dots, \frac{\partial D}{\partial \hat{x}_N} = 0$. The result of solving the $N+2$ equations can be summarized as follows. For $N = \text{even}$:

$$a_i = -a_{N-i} = -\left(\frac{N}{2} - i\right)\Delta \quad 1 \leq i \leq \frac{N}{2} \quad (17.10)$$

and

$$\hat{x}_i = \hat{x}_{N+1-i} = -\left(\frac{N}{2} - i + \frac{1}{2}\right)\Delta. \quad (17.11)$$

For $N = \text{odd}$:

$$a_i = -a_{N-i} = -\left(\frac{N}{2} + i\right)\Delta \quad 1 \leq i \leq \frac{N+1}{2} \quad (17.12)$$

and

$$\hat{x}_i = -\hat{x}_{N+1-i} = -\left(\frac{N}{2} + i + \frac{1}{2}\right)\Delta. \quad (17.13)$$

17.3 Still Images and JPEG Compression

This section investigates algorithms that prepare and compress still and moving images. The compression of such data substantially affects the utilization of bandwidths over the multimedia and IP networking infrastructures. We begin with a single visual image, such as a photograph, and then look at video, a motion image. The *Joint Photographic Experts Group* (JPEG) is the compression standard for still images. It is used for gray-scale and quality-color images. Similar to voice compression, JPEG is a lossy process. An image obtained after the decompression at a receiving end may not be the same as the original.

Figure 17.4 gives an overview of a typical JPEG process, which consists of three processes: *discrete cosine transform* (DCT), *quantization*, and *compression*, or encoding.

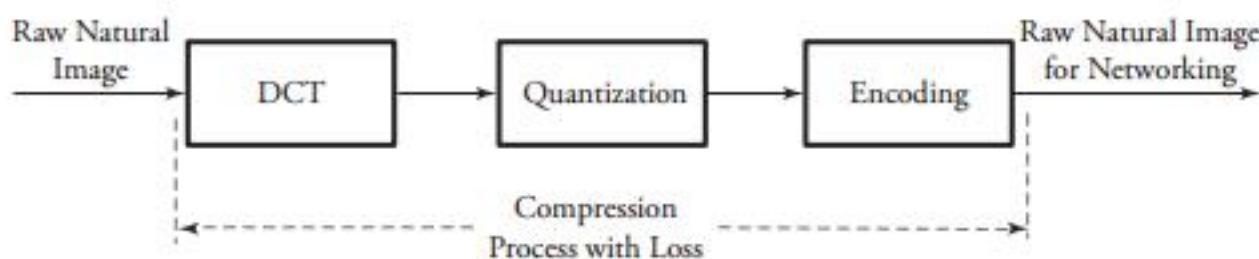


Figure 17.4 A typical JPEG process for production and compression of still images

The DCT process is complex and converts a snapshot of a real image into a matrix of corresponding values. The quantization phase converts the values generated by DCT to simple numbers in order to occupy less bandwidth. As usual, all quantizing processes are lossy. The compression process makes the quantized values as compact as possible. The compression phase is normally lossless and uses standard compression techniques. Before describing these three blocks, we need to consider the nature of a digital image.

17.3.1 Raw-Image Sampling and DCT

As with a voice signal, we first need samples of a raw image: a *picture*. Pictures are of two types: *photographs*, which contain no digital data, and *images*, which contain digital data suitable for computer networks. An image is made up of $m \times n$ blocks of picture units, or *pixels*, as shown in Figure 17.5. For FAX transmissions, images are made up of 0s and 1s to represent black and white pixels, respectively. A *monochrome image*, or *black-and-white image*, is made up of various shades of gray, and thus each pixel must be able to represent a different shade. Typically, a pixel in a monochrome image consists of 8 bits to represent $2^8 = 256$ shades of gray, ranging from white to black, as shown in Figure 17.6.

JPEG Files

Color images are based on the fact that any color can be represented to the human eye by using a particular combination of the base colors red, green, and blue (RGB). Computer monitor screens, digital camera images, or any other still color images are formed by varying the intensity of the three primary colors at pixel level, resulting in the creation of virtually any corresponding color from the real raw image. Each intensity created on any of the three pixels is represented by 8 bits, as shown in Figure 17.6. The intensities of each 3-unit pixel are adjusted, affecting the value of the 8-bit word to produce the desired color. Thus, each pixel can be represented by using 24 bits, allowing 2^{24} different colors. However, the human eye cannot distinguish all colors

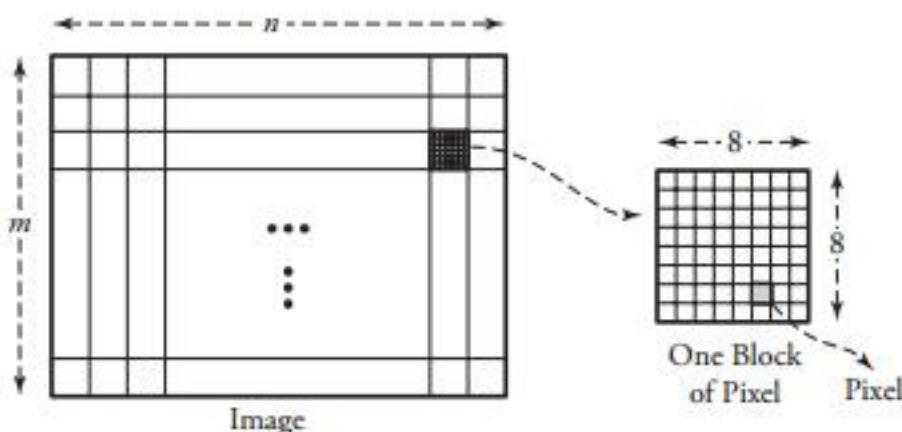


Figure 17.5 A digital still image

		Red	Green	Blue
White	0000,0000 0000,0001 ⋮	0000,0000 0000,0001 ⋮	0000,0000 0000,0001 ⋮	0000,0000 0000,0001 ⋮
Black	1111,1111	1111,1111	1111,1111	1111,1111
	(a)	(b)		

Figure 17.6 Still image in bits: (a) monochrome codes for still image; (b) color codes for still image

among the 2^{24} possible colors. The number of pixels in a typical image varies with the image size.

Example. A JPEG-based computer screen can consist of $1,024 \times 1,280$ pixels. Consequently, this computer image requires $(1,024 \times 1,280) \times 24 = 31,457,280$ bits. If a video consists of 30 images per second, a 943 Mb/s bandwidth is required.

GIF Files

JPEG is designed to work with full-color images up to 2^{24} colors. The *graphics interchange format* (GIF) is an image file format that reduces the number of colors to 256. This reduction in the number of possible colors is a trade-off between the quality of the image and the transmission bandwidth. GIF stores up to $2^8 = 256$ colors in a table and covers the range of colors in an image as closely as possible. Therefore, 8 bits are used to represent a single pixel. GIF uses a variation of Lempel-Ziv encoding (Section 17.6.3) for compression of an image. This technology is used for images whose color detailing is not important, such as cartoons and charts.

DCT Process

The *discrete cosine transform* (DCT) is a lossy compression process that begins by dividing a raw image into a series of standard $N \times N$ pixel blocks. For now, consider a monochrome block of pixels. With a standard size of $N = 8$, N is the number of pixels per row or column of each block. Let x, y be the position of a particular pixel in a block where $0 \leq x \leq N - 1$ and $0 \leq y \leq N - 1$. Hence, a gray scale of a given pixel x, y can get an integer value in $\{0, 1, 2, \dots, 255\}$. For an $N \times N$ pixel block, the DCT process is summarized in two steps

1. Form a $P[x][y]$ matrix to represent the collection of light-intensity values taken from various points of a real raw image.
2. Convert the values of $P[x][y]$ matrix to matrix with normalized and reduced values denoted by $T[i][j]$ obtained as follows.

The objective of matrix $T[i][j]$ is to create as many 0s as possible instead of small numbers in the $P[x][y]$ matrix in order to reduce the overall bandwidth required to transmit the image. Similarly, matrix $T[i][j]$ is a two-dimensional array with N rows and N columns, where $0 \leq i \leq N - 1$ and $0 \leq j \leq N - 1$. The elements of $T[i][j]$ are known as *spatial frequencies* and are obtained from

$$T[i][j] = \frac{2}{N} C(i) C(j) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \cos\left(\frac{\pi i(2x+1)}{2N}\right) \cos\left(\frac{\pi j(2y+1)}{2N}\right), \quad (17.14)$$

where

$$C(i) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } i = 0 \\ 0 & \text{otherwise} \end{cases}$$

$$C(j) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } j = 0 \\ 0 & \text{otherwise} \end{cases}$$

Example. An 8×8 matrix $P[x][y]$ for an image is formed as shown in Figure 17.7 (a). This matrix is converted to matrix $T[i][j]$ by using Equation (17.15). This example clearly shows that a matrix as $P[x][y]$ consisting of 64 values can be converted to $T[i][j]$ with 9 values and 55 zeros. It is easy to figure out the advantages of this conversion. The 55 zeros can be compressed, conveniently resulting in significant reduction of transmission bandwidth.

22	31	41	50	60	69	80	91
29	42	52	59	71	80	90	101
40	51	59	70	82	92	100	110
51	62	70	82	89	101	109	119
60	70	82	93	100	109	120	130
70	82	90	100	110	121	130	139
79	91	100	110	120	130	140	150
91	99	110	120	130	140	150	160

716	-179	0	-19	0	-6	0	-1
-179	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
-19	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
-6	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
-1	0	0	0	0	0	0	0

(a)

(b)

Figure 17.7 Matrix examples: (a) $P[x][y]$; (b) $T[i][j]$

The spatial frequencies depend directly on how much the pixel values change as functions of their positions in a pixel block. Equation (17.15) is set up such that the generated matrix $T[i][j]$ contains many 0s, and the values of the matrix elements generally become smaller as they get farther away from the upper-left position in the matrix. The values of $T[i][j]$ elements at the receiver can be converted back to matrix $P[x][y]$ by using the following function:

$$P[x][y] = \frac{2}{N} C(i) C(j) \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \cos\left(\frac{\pi i(2i+1)}{2N}\right) \cos\left(\frac{\pi j(2j+1)}{2N}\right). \quad (17.15)$$

Note that as values become farther away from the upper-left position, they correspond to a fine detail in the image.

17.3.2 Quantization

To further scale down the values of $T[i][j]$ with fewer distinct numbers and more consistent patterns to get better bandwidth advantages, this matrix is *quantized* to another matrix, denoted by $Q[i][j]$. To generate this matrix, the elements of matrix $T[i][j]$ are divided by a standard number and then rounded off to their nearest integer. Dividing $T[i][j]$ elements by the same constant number results in too much loss. The values of elements on the upper-left portions of the matrix must be preserved as much as possible, because such values correspond to less subtle features of the image. In contrast, values of elements in the lower-right portion correspond to the highest details of an image. To preserve as much information as possible, the elements of $T[i][j]$ are

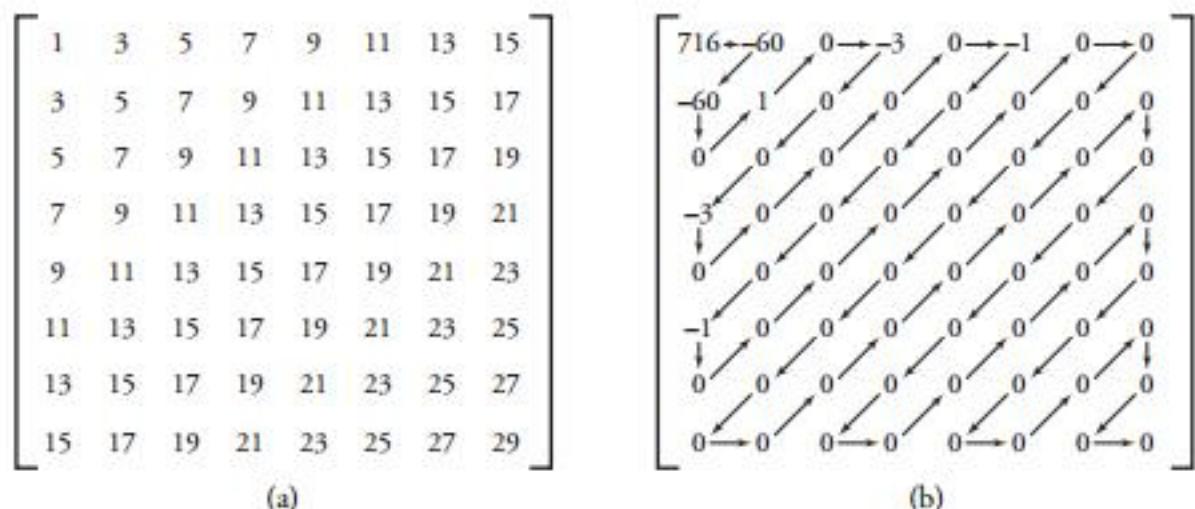


Figure 17.8 (a) Divisor matrix $D[i][j]$ and the quantization of a still image to produce; (b) matrix $Q[i][j]$ and the order of matrix elements for transmission

divided by the elements of an $N \times N$ matrix denoted by $D[i][j]$, in which the values of elements decrease from the upper-left portion to the lower-right portion.

Example. Consider again the 8×8 matrix $P[x][y]$ already converted to matrix $T[i][j]$ shown in Figure 17.7. Figure 17.8 (a) shows a divisor matrix $D[i][j]$. Figure 17.8 (b) shows the corresponding matrix $Q[i][j]$ resulting from the quantization process. Note particularly that big values, such as -179 have become smaller and that some such values, as -1 in the upper-right corner turn into 0, making it easier for compression.

We notice here that the process of quantization, as discussed at the beginning of this chapter, is not quite reversible. This means that the values of $Q[i][j]$ cannot be exactly converted back to $T[i][j]$, owing mainly to rounding the values after dividing them by $D[i][j]$. For this reason, the quantization phase is a lossy process.

17.3.3 Encoding

In the last phase of the JPEG process, encoding finally does the task of compression. In the quantization phase, a matrix with numerous 0s is produced. Consider the example shown in Figure 17.8 (b). The Q matrix in this example has produced 57 zeros from the original raw image. A practical approach to compressing this matrix is to use run-length coding (Section 17.6.1). If run-length coding is used, scanning matrix $Q[i][j]$ row by row may result in several phrases.

A logical way to scan this matrix is in the order illustrated by the arrows in Figure 17.8 (b). This method is attractive because the larger values in the matrix tend to collect

in the upper-left corner of the matrix, and the elements representing larger values tend to be gathered together in that area of the matrix. Thus, we can induce a better rule: Scanning should always start from the upper-left corner element of the matrix. This way, we get much longer runs for each phrase and a much lower number of phrases in the run-length coding.

Once the run-length coding is processed, JPEG uses some type of Huffman coding or arithmetic coding for the nonzero values. Note here that, so far, only a block of 8×8 pixels has been processed. An image consists of numerous such blocks. Therefore, the speed of processing and transmission is a factor in the quality of image transfer.

17.4 Moving Images and MPEG Compression

A *motion image*, or video is a rapid display of still images. Moving from one image to another must be fast enough to fool the human eye. There are different standards on the number of still images comprising a video clip. One common standard produces a motion image by displaying still images at a rate of 30 frames per second. The common standard that defines the video compression is the *Moving Pictures Expert Group* (MPEG), which has several branch standards:

- MPEG-1, primarily for video on CD-ROM
- MPEG-2, for multimedia entertainment and *high-definition television* (HDTV) and the satellite broadcasting industry
- MPEG-4, for object-oriented video compression and videoconferencing over low-bandwidth channels
- MPEG-7, for a broad range of demands requiring large bandwidths providing multimedia tools
- MPEG-21 for interaction among the various MPEG groups

Logically, using JPEG compression for each still picture does not provide sufficient compression for video as it occupies a large bandwidth. MPEG deploys additional compression. Normally, the difference between two consecutive frames is small. With MPEG, a base frame is sent first, and successive frames are encoded by computing the differences. The receiver can reconstruct frames based on the first base frame and the submitted differences. However, frames of a completely new scene in a video may not be compressed this way, as the difference between the two scenes is substantial. Depending on the relative position of a frame in a sequence, it can be compressed through one of the following types of frames:

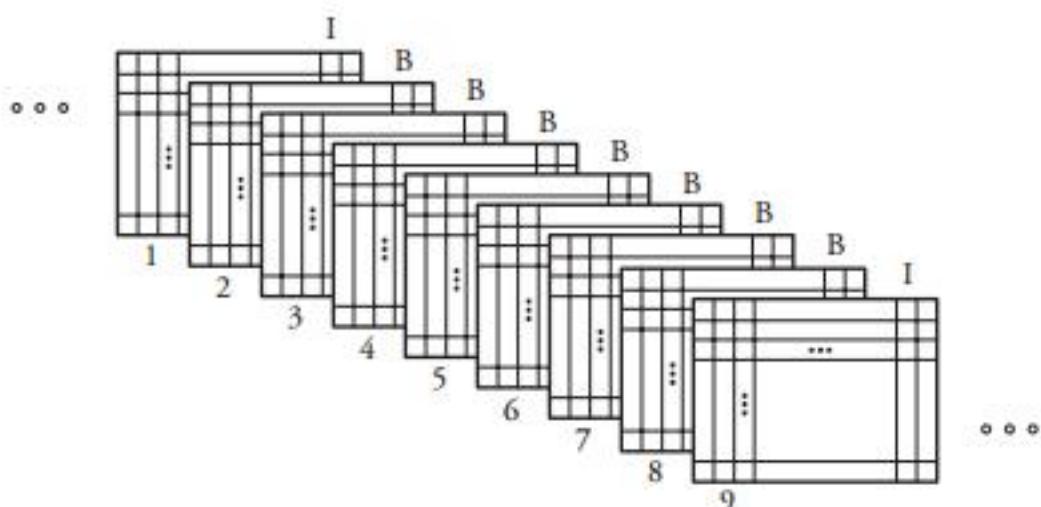


Figure 17.9 Snapshot of moving frames for MPEG compression

- *Interimage (I) frames.* An I frame is treated as a JPEG still image and compressed using DCT.
- *Predictive (P) frames.* These frames are produced by computing differences between a current and a previous I or P frame.
- *Bidirectional (B) frames.* A B frame is similar to a P frame, but the P frame considers differences between a previous, current, and future frames.

Figure 17.9 illustrates a typical grouping of frames, with I, P, and B frames forming a sequence. In any frame sequence, I frames appear periodically as the base of the scene. Normally, there is a P frame between each two groups of B frames.

17.4.1 MP3 and Streaming Audio

Section 17.2 explained how an audible sound or a human voice ranging between 20 Hz and 20 KHz can be converted into digital bits and eventually into packets for networking. A signal is sampled, quantized, and encoded, a process called PCM. A variety of methods of compressing such encoded products at the output of the PCM are available. However, Huffman compression of the processed signal may not be sufficient for transmission over IP networks.

The MPEG-1 layer 3 (MP3) technology compresses audio for networking and producing CD-quality sound. The sampling part of PCM is performed at a rate of 44.1 KHz to cover the maximum of 20 KHz of audible signals. Using the commonly used 16-bit encoding for each sample, the maximum total bits required for audio is $16 \times 44.1 = 700$ kilobits and 1.4 megabits for two channels if the sound is processed

in a stereo fashion. For example a 60-minute CD (3,600 seconds) requires about $1.4 \times 3,600 = 5,040$ megabits, or 630 megabytes. This amount may be acceptable for recording on a CD but is considered extremely large for networking, and thus a carefully designed compression technique is needed.

MP3 combines the advantages of MPEG with “three” layers of audio compressions. MP3 removes from a piece of sound all portions that an average ear may not be able to hear, such as weak background sounds. On any audio streaming, MP3 specifies what humans are not able to hear, removes those components, and digitizes the remaining. By filtering some part of an audio signal, the quality of compressed MP3 is obviously degraded to lower than the original one. Nonetheless, the compression achievement of this technology is remarkable.

17.5 Limits of Compression with Loss

Hartley, Nyquist, and Shannon are the founders of *information theory*, which has resulted in the mathematical modeling of information sources. Consider a communication system in which a source signal is processed to produce sequences of n words, as shown in Figure 17.10. These sequences of digital bits at the output of the information source can be compressed in the *source encoder* unit to save the transmission link bandwidth. An information source can be modeled by a *random process* $X_n = (X_1, \dots, X_n)$, where X_i is a random variable taking on values from a set of values as $\{a_1, \dots, a_N\}$, called *alphabet*. We use this model in our analysis to show the information process in high-speed networks.

17.5.1 Basics of Information Theory

The challenge of data compression is to find the output that conveys the most information. Consider a single source with random variable X , choosing values in $\{a_1, \dots, a_N\}$.

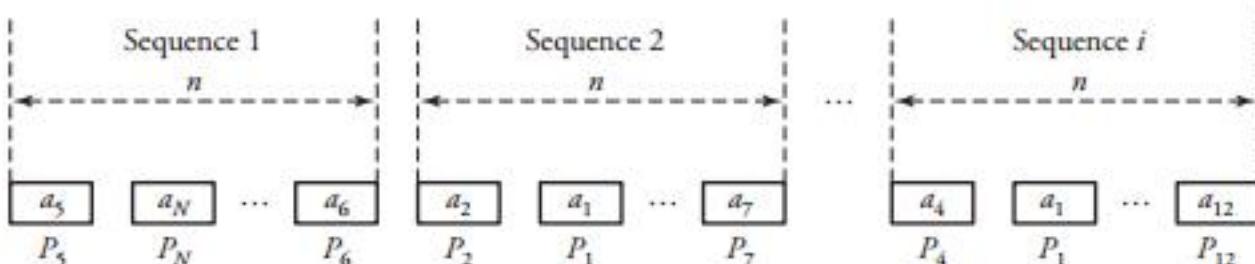


Figure 17.10 A model of data sequences

If a_i is the most likely output and a_j is the least likely output, clearly, a_j conveys the most information and a_i conveys the least information. This observation can be rephrased as an important conclusion: *The measure of information for an output is a decreasing and continuous function of the probability of source output.* To formulate this statement, let P_{k_1} and P_{k_2} be the probabilities of an information source's outputs a_{k_1} and a_{k_2} , respectively. Let $I(P_{k_1})$ and $I(P_{k_2})$ be the information content of a_{k_1} and a_{k_2} , respectively. The following four facts apply.

1. As discussed, $I(P_k)$ depends on P_k .
2. $I(P_k)$ = a continuous function of P_k .
3. $I(P_k)$ = a decreasing function of P_k .
4. $P_k = P_{k_1} \cdot P_{k_2}$ (probability of two outputs happen in the same time).
5. $I(P_k) = I(P_{k_1}) + I(P_{k_2})$ (sum of two pieces of information).

These facts lead to an important conclusion that can relate the probability of a certain data to its information content:

$$I(P_k) = -\log_2 P_k = \log_2 \left(\frac{1}{P_k} \right). \quad (17.16)$$

The log function has a base 2, an indication of incorporating the binary concept of digital data.

17.5.2 Entropy of Information

In general, *entropy* is a measure of uncertainty. Consider an information source producing random numbers, X , from a possible collection of $\{a_1, \dots, a_N\}$ with corresponding probabilities of $\{P_1, \dots, P_N\}$ and information content of $\{I(P_1), \dots, I(P_N)\}$, respectively. In particular, the entropy, $H(x)$, is defined as the average information content of a source:

$$\begin{aligned} H_X(x) &= \sum_{k=1}^N P_k I(P_k) \\ &= \sum_{k=1}^N -P_k \log_2 P_k = \sum_{k=1}^N P_k \log_2 \left(\frac{1}{P_k} \right). \end{aligned} \quad (17.17)$$

Example. A source with bandwidth 8 KHz is sampled at the Nyquist rate. If the result is modeled using any value from $\{-2, -1, 0, 1, 2\}$ and corresponding probabilities $\{0.05, 0.05, 0.08, 0.30, 0.52\}$, find the entropy.

Solution.

$$H_X(x) = - \sum_{k=1}^5 P_k \log_2 P_k = 0.522 \text{ bits/sample.}$$

The information rate in samples/sec = $8,000 \times 2 = 16,000$, and the rate of information produced by the source = $16,000 \times 0.522 = 8,352$ bits.

Joint Entropy

The joint entropy of two discrete random variables X and Y is defined by

$$H_{X,Y}(x, y) = - \sum_{x,y} P_{X,Y}(x, y) \log_2 P_{X,Y}(x, y), \quad (17.18)$$

where $P_{X,Y}(x, y) = \text{Prob}[X = x \text{ and the same time } Y = y]$ and is called the *joint probability mass function* of two random variables. In general, for a random process $\mathbf{X}_n = (X_1, \dots, X_n)$ with n random variables:

$$H_{\mathbf{X}_n}(x_n) = - \sum_{X_1, \dots, X_n} P_{X_1, \dots, X_n}(x_1, \dots, x_n) \log_2 P_{X_1, \dots, X_n}(x_1, \dots, x_n), \quad (17.19)$$

where $P_{X_1, \dots, X_n}(x_1, \dots, x_n)$ is the joint probability mass function (J-PMF) of the random process \mathbf{X}_n . For more information on J-PMF, see Appendix C.

17.5.3 Shannon's Coding Theorem

This theorem limits the rate of data compression. Consider again Figure 17.10, which shows a discrete source modeled by a random process that generates sequences of length n using values in set $\{a_1, \dots, a_N\}$ with probabilities $\{P_1, \dots, P_N\}$, respectively. If n is large enough, the number of times a value a_i is repeated in a given sequence = $n P_i$, and the number of values in a typical sequence is therefore $n(P_1 + \dots + P_N)$.

We define the *typical sequence* as one in which any value a_i is repeated $n P_i$ times. Accordingly, the probability that a_i is repeated $n P_i$ times is obviously $P_i P_i \cdots P_i = P_i^{n P_i}$, resulting in a more general statement: The probability of a typical sequence is the probability [(a_1 is repeated $n p_1$)] \times the probability [(a_2 is repeated $n p_2$)] $\times \dots$. This can be shown by $P_1^{n p_1} P_2^{n p_2} \cdots P_N^{n p_N}$, or

$$\text{Prob(Typical Sequence)} = \prod_{i=1}^N P_i^{n P_i}. \quad (17.20)$$

Knowing $P_i^{n P_i} = 2^{n P_i \log_2 P_i}$, we can obtain the probability of a typical sequence P_t as follows:

$$\begin{aligned} P_t &= \prod_{i=1}^N 2^{n P_i \log_2 P_i} \\ &= 2^{(n P_1 \log_2 P_1 + \dots + n P_N \log_2 P_N)} \\ &= 2^{n(P_1 \log_2 P_1 + \dots + P_N \log_2 P_N)} \\ &= 2^{n(\sum_{i=1}^N P_i \log_2 P_i)}. \end{aligned} \quad (17.21)$$

This last expression results in the well-known Shannon's theorem, which expresses the probability that a typical sequence of length n with entropy $H_X(x)$ is equal to

$$P_t = 2^{-nH_X(x)}. \quad (17.22)$$

Example. Assume that a sequence size of 200 of an information source chooses values from the set $\{a_1, \dots, a_5\}$ with corresponding probabilities $\{0.05, 0.05, 0.08, 0.30, 0.52\}$. Find the probability of a typical sequence.

Solution. In the previous example, we calculated the entropy to be $H_X(x) = 0.522$ for the same situation. With $n = 200$, $N = 5$, the probability of a typical sequence is the probability of a sequence in which a_1, a_2, a_3, a_4 , and a_5 are repeated, respectively, $200 \times 0.05 = 10$ times, 10 times, 16 times, 60 times, and 104 times. Thus, the probability of a typical sequence is $P_t = 2^{-nH_X(x)} = 2^{-200(0.52)}$.

The fundamental Shannon's theorem leads to an important conclusion. As the probability of a typical sequence is $2^{-nH_X(x)}$ and the sum of probabilities of all typical sequences is 1, the number of typical sequences is obtained by $= \frac{1}{2^{-nH_X(x)}} = 2^{nH_X(x)}$. Knowing that the total number of all sequences, including typical and nontypical ones, is N^n , it is sufficient, in all practical cases when n is large enough, to transmit only the set of typical sequences rather than the set of all sequences. This is the essence of data compression: The total number of bits required to represent $2^{nH_X(x)}$ sequences is $nH_X(x)$ bits, and the average number of bits for each source = $H_X(x)$.

Example. Following the previous example, in which the sequence size for an information source is 200, find the ratio of the number of typical sequences to the number of all types of sequences.

Solution. We had $n = 200$ and $N = 5$; thus, the number of typical sequences is $2^{nH_X(x)} = 2^{200 \times 0.522}$, and the total number of all sequences is 5^{200} . This ratio is almost zero, which may cause a significant loss of data if it is compressed, based on Shannon's

theorem. It is worth mentioning that the number of bits required to represent $2^{nH_X(x)}$ sequences is $nH_X(x) = 104$ bits.

17.5.4 Compression Ratio and Code Efficiency

Let \tilde{R} be the average length of codes, ℓ_i be the length of code word i , and P_i be the probability of code word i :

$$\tilde{R}_i = \sum_{i=0}^N P_i \ell_i. \quad (17.23)$$

A *compression ratio* is defined as

$$C_r = \frac{\tilde{R}}{\tilde{R}_x}, \quad (17.24)$$

where \tilde{R}_x is the length of a source output before coding. It can also be shown that

$$H_X(x) \leq \tilde{R} < H_X(x) + 1. \quad (17.25)$$

Code efficiency is a measure for understanding how close code lengths are to the corresponding decoded data and is defined by

$$\eta_{code} = \frac{H_X(x)}{\tilde{R}}. \quad (17.26)$$

When data is compressed, part of the data may need to be removed in the process.

17.6 Compression Methods Without Loss

Some types of data, including text, image, and video, might contain redundant or repeated elements. If so, those elements can be eliminated and some sort of codes substituted for future decoding. In this section, we focus on techniques that do not incur any loss during compression:

- Arithmetic encoding
- Run-length encoding
- Huffman encoding
- Lempel-Ziv encoding

Here, we ignore arithmetic encoding and consider only the last three encoding techniques.

Table 17.1 Statistics obtained for run-length compression of a 1,000-character piece of text

Number of Repeated Characters	Average Length of Repeated Characters	Compression Ratio C_r
10	4	0.99
10	10	0.93
20	4	0.98
20	10	0.85
30	4	0.97
30	10	0.79

17.6.1 Run-Length Encoding

One of the simplest data-compression techniques is *run-length encoding*. This technique is fairly effective for compression of plaintext and numbers, especially for facsimile systems. With run-length code, repeated letters can be replaced by a run length, beginning with C_c to express the compression letter count.

Example. Assume a system that represents b as a blank. Find the compressed version of the following sentence:

THISSSSSSbISbbbANbEXAMPLEbOFbRUN——LENGTHbCODE

Solution. According to the conventions stated, the compressed version of that sentence turns into:

THIS C_c 6SbISC C_c b4bANbEXAMPLEbOFbRUNC C_c -5LENGTHbCODE

It is obvious that the longer the text, the smaller the compression ratio becomes, as shown in Table 17.1. The statistics obtained in this table are based on a 1,000-character piece of text.

17.6.2 Huffman Encoding

Huffman encoding is an efficient frequency-dependent coding technique. With this algorithm, source values with smaller probabilities appear to be encoded by a longer word.

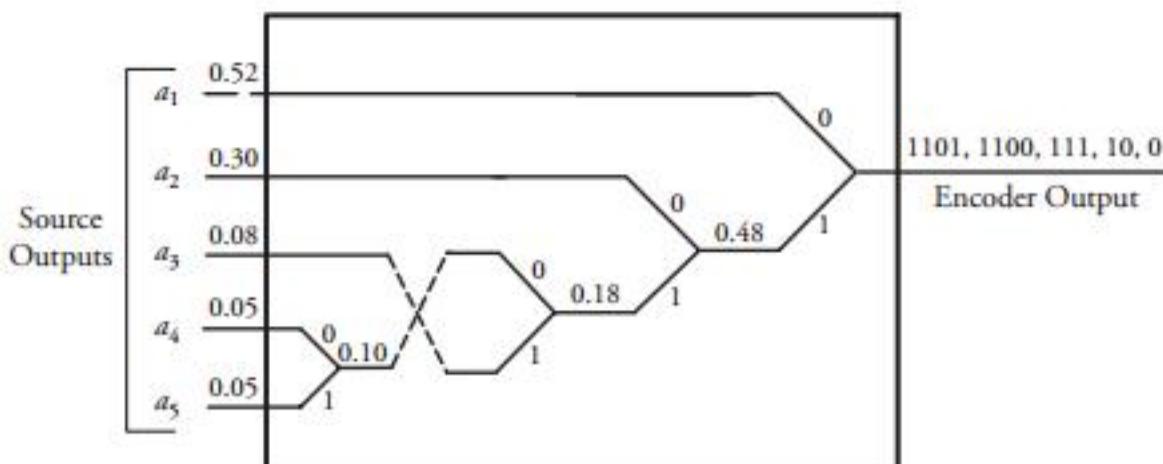


Figure 17.11 Huffman encoding

This technique reduces the total number of bits, leading to an efficient compression of data. The algorithm that implements such a technique is as follows.

Begin Huffman Encoding Algorithm

1. Sort outputs of the source in decreasing order of their probabilities. For example, 0.7, 0.6, 0.6, 0.59, ..., 0.02, 0.01.
2. Merge the two least probabilistic outputs into a single output whose probability is the sum of corresponding probability, such as $0.02 + 0.01 = 0.03$.
3. If the number of remaining outputs is 2, go to the next step; otherwise, go to step 1.
4. Assign 0 and 1 as codes on the diagram.
5. If a new output is the result of merging two outputs, append the code word with 0 and 1; otherwise, stop. ■

Example. Design a Huffman encoder for a source generating $\{a_1, a_2, a_3, a_4, a_5\}$ and with probabilities $\{0.05, 0.05, 0.08, 0.30, 0.52\}$.

Solution. Following the algorithm, the output of the information source shown in Figure 17.11, the information related to $\{a_1, a_2, a_3, a_4, a_5\}$ is compressed to 1100, 1101, 111, 10, 0, respectively.

17.6.3 Lempel-Ziv Encoding

Lempel-Ziv codes are independent of the source statistics. This coding technique is normally used for UNIX compressed files. The algorithm that converts a string of logical bits into a Lempel-Ziv code is summarized as follows.

Begin Lempel-Ziv Encoding Algorithm

1. Any sequence of source output is passed in a phrase of varying length. At the first step, identify phrases of the smallest length that have not appeared so far. Note that all phrases are different, and lengths of words grow as the encoding process proceeds.
2. Phrases are encoded using code words of equal length. If k_1 = number of bits are needed to describe the code word and k_2 = the number of phrases, we must have

$$k_1 = \log_2 [k_2]_2.$$

3. A code is the location of the prefix to the phrases.
4. A code is followed by the last bit of parser output to double-check the last bit. ■

Example. For the following string of bits, find the encoded Lempel-Ziv words:

11110111011000001010010001111010101100

Solution. Implementing step 1 on the string, there are 14 phrases, as follows:

1 – 11 – 10 – 111 – 0 – 110 – 00 – 001 – 01 – 0010 – 0011 – 1101 – 010 – 1100

Thus, $k_2 = 14$ and $k_1 = \log_2 [14]_2 = 4$. Table 17.2 shows the encoded words as steps 3 and 4 are applied on the parser output.

17.7 Case Study: FAX Compression for Transmission

A FAX picture is scanned and compressed in two steps: run-length encoding and then Huffman encoding. First, the transmission of the digital line scan is replaced by the transmission of a quantity count of each of the successive runs of black or white elements.

Consider a document of standard size 8.5 inches by 11 inches. The picture is first partitioned into *pixels*. If the desired resolution is 200×200 pixels per square inch, the total number of pixels per picture is exactly $200^2 \times (8.5 \times 11) = 37,400,000$ pixels.

Fax Process Algorithm

As mentioned earlier, processing a FAX picture requires both run-length coding and Huffman coding. Since black and white always alternate, no special characters are

Table 17.2 An example of Lempel-Ziv coding process

Parser Output	Location	Encoded Output
1	0001	00001
11	0010	00011
10	0011	00010
111	0100	00000
110	0110	00100
00	0111	01010
001	1000	01111
01	1001	01011
0011	0001	10000
0011	1011	10001
1101	1101	01101
011	1101	10010
1100	1110	01100
1	0001	00001

needed to specify that a run is black or white. Thus, the encoded data stream is a string of numbers indicating the lengths of the alternating black and white runs. The algorithm for the first phase is as follows.

1. Identify the first row out of the n -row document.
2. At any row i , start at the first pixel of the row. If the pixel is black, assign code 1; if the pixel is white, assign code 0.
3. At any step of counting j , let X_j be the number of consecutive 0s before a 1 appears. Then assign code $C_c X_j 0$ to this string of 0s. Do the same thing for 1s, and code it with $C_c X_j 1$.

At this point, the document is converted into a number of $C_c X_j 0$ s and $C_c X_j 1$. In phase 2 of coding, we need the statistics on the frequencies of a certain run-length code in order to compress it further, using the Huffman algorithm. Table 17.3 shows practical statistics for a black-and-white FAX document.

Table 17.3 Statistics on frequency of occurrences for strings obtained after run-length coding for black-and-white FAX document

Number of Repeated Pixels ($C_c X$)	Huffman Code for White Pixels	Huffman Code for Black Pixels
$C_c 1$	000111	010
$C_c 2$	0011	01
$C_c 10$	00011	1000011
$C_c 50$	00111001	000001101111

17.8 Summary

A number of algorithms effectively compress voice, still images, and moving images. A raw voice signal is converted to a binary-encoded form known as *pulse code modulation* (PCM). In this process, sampling and quantization both present some sort of loss. Compression schemes used for the preparation of an image include the *Joint Photographic Experts Group* (JPEG), a compression standard of still images. JPEG consists of three processes: *discrete cosine transform* (DCT), *quantization*, and *compression*. In particular, DCT converts a snapshot of a real image into a matrix of corresponding values, and the quantization phase converts the values generated by DCT to simple numbers.

A *motion image*, or video, is the rapid display of still images to fool the human eye. Standards differ on the number of still images that make a video clip. The common standard that defines video compression is the *Moving Pictures Expert Group* (MPEG). MPEG-1 layer 3 (MP3) is a technology for compressing audio for networking and producing CD-quality sound.

Compression has limits, as presented by Shannon. Shannon's theorem expresses the probability of a typical sequence of length n with entropy $H_X(x)$ to be equal to $2^{-nH_X(x)}$. Although some compression processes entail loss of data, others do not, such as Huffman or run-length encoding techniques.

The next chapter looks at multimedia networking, voice over IP (VIP), and streaming video.

17.9 Exercises

1. A sinusoidal signal $g(t)$ with period 10 ms is to be sampled by a sampler $s(t)$ with period $T_s = 1$ ms and pulse width $\tau = 0.5$ ms. The maximum voltages for both

signals are 1 volt. Let the sampled signal be $g_s(t)$; compute and sketch $g(t)$, $s(t)$, $g_s(t)$, $G(f)$, $S(f)$, and $G_s(f)$ (the range of nT_s in $s(t)$ is $[-2T_s, +2T_s]$).

2. Consider a pulse signal $g(t)$ being 1 volt in intervals: $\dots [-4 \text{ and } -2], [-1 \text{ and } +1], [+2 \text{ and } +4] \dots$ ms. This signal is to be sampled by an impulse sampler $s(t)$ being generated at $\dots, -3, 0, +3, \dots$ ms, and a convolving pulse $c(t)$ of $\tau = 1$ volt between -1 and $+1$ ms. Compute and sketch all the processes from analog signal $g(t)$ to sampled version $g_s(t)$ in both time and frequency domains.
3. Assume that a normal-distributed source with zero mean and variance of 2 is to be transmitted via a channel that can provide a transmission capacity of 4 bits/each source output.
 - (a) What is the minimum mean-squared error achievable?
 - (b) What is the required transmission capacity per source output if the maximum tolerable distortion is 0.05?
4. Let $X(t)$ denote a normal (Gaussian) source with $\sigma^2 = 10$, for which a 12-level optimal uniform quantizer is to be designed.
 - (a) Find optimal quantization intervals (Δ).
 - (b) Find optimal quantization boundaries (a_i).
 - (c) Find optimal quantization levels (\hat{x}_i).
 - (d) Find the optimal total resulting distortion.
 - (e) Compare the optimal total resulting distortion with the result obtained from the rate-distortion bound that achieves the same amount of distortion.
5. Consider the same information source discussed in exercise 4. This time, apply a 12-level optimal nonuniform quantizer.
 - (a) Find optimal quantization boundaries (a_i).
 - (b) Find optimal quantization intervals (Δ).
 - (c) Find optimal quantization levels (\hat{x}_i).
 - (d) Find the optimal total resulting distortion.
6. To encode two random signals X and Y that are uniformly distributed on the region between two squares, let the marginal PDF of random variables be

$$f_X(x) = \begin{cases} 0.2 & -2 \leq X < -1 \\ 0.2 & -1 \leq X < 0 \\ 0.2 & 0 \leq X < +1 \\ 0.2 & +1 \leq X < +2 \end{cases}$$

and

$$f_Y(y) = \begin{cases} 0.3 & -2 \leq X < -1 \\ 0.1 & -1 \leq X < 0 \\ 0.4 & 0 \leq X < +1 \\ 0.2 & +1 \leq X < +2 \end{cases}.$$

Assume that each of the random variables X and Y is quantized using four-level uniform quantizers.

- (a) Calculate the joint probability $P_{XY}(x, y)$.
 - (b) Find quantization levels x_1 through x_4 if $\Delta = 1$.
 - (c) Without using the optimal quantization table, find the resulting total distortion.
 - (d) Find the resulting number of bits per (X, Y) pair.
7. The sampling rate of a certain CD player is 80,000, and samples are quantized using a 16 bit/sample quantizer. Determine the resulting number of bits for a piece of music with a duration of 60 minutes.
8. The PDF of a source is defined by $f_X(x) = 2\Lambda(x)$. This source is quantized using an eight-level uniform quantizer described as follows:

$$Q(x) = \begin{cases} +3 & 1 < x \leq 2 \\ +3 & 0 < x \leq 1 \\ +3 & -1 < x \leq 0 \\ +3 & -2 < x \leq -1 \end{cases}.$$

Find the PDF of the random variable representing the quantization error $X - Q(X)$.

- 9. Using logic gates, design a PCM encoder using 3-bit gray codes.
- 10. To preserve as much information as possible, the JPEG elements of $T[i][j]$ are divided by the elements of an $N \times N$ matrix denoted by $D[i][j]$, in which the values of elements decrease from the upper-left portion to the lower-right portion. Consider matrices $T[i][j]$ and $D[i][j]$ in Figure 17.12.
 - (a) Find the quantized matrix $Q[i][j]$.
 - (b) Obtain a run-length compression on $Q[i][j]$.

$$\begin{bmatrix} 1 & 3 & 5 & 7 & 9 & 11 & 13 & 15 \\ 3 & 5 & 7 & 9 & 11 & 13 & 15 & 17 \\ 5 & 7 & 9 & 11 & 13 & 15 & 17 & 19 \\ 7 & 9 & 11 & 13 & 15 & 17 & 19 & 21 \\ 9 & 11 & 13 & 15 & 17 & 19 & 21 & 23 \\ 11 & 13 & 15 & 17 & 19 & 21 & 23 & 25 \\ 13 & 15 & 17 & 19 & 21 & 23 & 25 & 27 \\ 15 & 17 & 19 & 21 & 23 & 25 & 27 & 29 \end{bmatrix}$$

(a)

$$\begin{bmatrix} 513 & -138 & 0 & -17 & 0 & -6 & 0 & -1 \\ -138 & 1 & 0 & 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -17 & 6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(b)

Figure 17.12 Exercise 10 matrices for applying (a) divisor matrix $D[i][j]$ on (b) matrix $T[i][j]$ to produce an efficient quantization of a JPEG image to produce matrix $Q[i][j]$

11. Find the differential entropy of the continuous random variable X with a PDF defined by

$$f_X(x) = \begin{cases} x + 1 & -1 \leq x \leq 0 \\ -x + 1 & 0 < x \leq 1 \\ 0 & \text{else} \end{cases}$$

12. A source has an alphabet $\{a_1, a_2, a_3, a_4, a_5\}$ with corresponding probabilities $\{0.23, 0.30, 0.07, 0.28, 0.12\}$.
- Find the entropy of this source.
 - Compare this entropy with that of a uniformly distributed source with the same alphabet.

13. We define two random variables X and Y for two random voice signals in a multi-media network, both taking on values in alphabet $\{1, 2, 3\}$. The joint probability mass function (JPMF), $P_{X,Y}(x, y)$, is given as follows:

$$\begin{cases} P_{X,Y}(1, 1) = P(X = 1, Y = 1) = 0.1 \\ P_{X,Y}(1, 2) = P(X = 1, Y = 2) = 0.2 \\ P_{X,Y}(2, 1) = P(X = 2, Y = 1) = 0.1 \\ P_{X,Y}(1, 3) = P(X = 1, Y = 3) = 0.4 \\ P_{X,Y}(2, 3) = P(X = 2, Y = 3) = 0.2 \end{cases}$$

- Find the two marginal entropies, $H(X)$ and $H(Y)$.

- (b) Conceptually, what is the meaning of the marginal entropy?
(c) Find the joint entropy of the two signals, $H(X, Y)$.
(d) Conceptually, what is the meaning of the joint entropy?
14. We define two random variables X and Y for two random voice signals in a multimedia network.
(a) Find the conditional entropy, $H(X|Y)$, in terms of joint and marginal entropies.
(b) Conceptually, what is the meaning of the joint entropy?
15. Consider the process of a source with a bandwidth $W = 50$ Hz sampled at the Nyquist rate. The resulting sample outputs take values in the set of alphabet $\{a_0, a_1, a_2, a_3, a_4, a_5, a_6\}$ with corresponding probabilities $\{0.06, 0.09, 0.10, 0.15, 0.05, 0.20, 0.35\}$ and are transmitted in sequences of length 10.
(a) Which output conveys the most information? item What is the information content of outputs a_1 and a_5 together?
(b) Find the least-probable sequence and its probability, and comment on whether it is a typical sequence.
(c) Find the entropy of the source in bits/sample and bits/second.
(d) Calculate the number of nontypical sequences.
16. A source with the output alphabet $\{a_1, a_2, a_3, a_4\}$ and corresponding probabilities $\{0.15, 0.20, 0.30, 0.35\}$ produces sequences of length 100.
(a) What is the approximate number of typical sequences in the source output?
(b) What is the ratio of typical sequences to nontypical sequences?
(c) What is the probability of a typical sequence?
(d) What is the number of bits required to represent only typical sequences?
(e) What is the most probable sequence, and what is its probability?
17. For a source with an alphabet $\{a_0, a_1, a_2, a_3, a_4, a_5, a_6\}$ and with corresponding probabilities $\{0.55, 0.10, 0.05, 0.14, 0.06, 0.08, 0.02\}$:
(a) Design a Huffman encoder.
(b) Find the code efficiency.
18. A voice information source can be modeled as a band-limited process with a bandwidth of 4,000 Hz. This process is sampled at the Nyquist rate. In order to provide a guard band to this source, 200 Hz is added to the bandwidth for which a Nyquist rate is not needed. It is observed that the result-

ing samples take values in the set $\{-3, -2, -1, 0, 2, 3, 5\}$, with probabilities $\{0.05, 0.1, 0.1, 0.15, 0.05, 0.25, 0.3\}$.

- (a) What is the entropy of the discrete time source in bits/output?
 - (b) What is the entropy in b/s?
 - (c) Design a Huffman encoder.
 - (d) Find the compression ratio and code efficiency for Part (c).
19. Design a Lempel-Ziv encoder for the following source sequence:

01010000100011111001010101111010010101010

20. Design a Lempel-Ziv encoder for the following source sequence:

1111100010101011101111100010101010001111010100001

21. *Computer simulation project.* Using a computer program, implement Equation (17.15) to obtain $T[i][j]$ matrix for a JPEG compression process.

This page intentionally left blank

CHAPTER 18

VoIP and Multimedia Networking

The discussion in the previous chapter on compressed voice and video sets the stage for the discussion in this chapter on multimedia networking. The communication industry has spent considerable effort in designing an IP-based media transport mechanism: *voice over IP* (VoIP), which can deliver voice-band telephony with the quality of telephone networks. Internet phone services are less expensive and have more features, such as video conferencing, online directory services, and Web incorporation. *Multimedia networking* is one of the most effective Internet developments. In addition to data, the Internet is used to transport phone calls, audio, and video. This chapter looks at the transportation of real-time signals along with the signaling protocols used in voice telephony, video streaming, and multimedia networking, covering the following major topics:

- *Overview of IP telephony*
- *VoIP signaling protocols*
- *Real-time media transport protocols*
- *Distributed multimedia networking*
- *Stream Control Transmission Protocol (SCTP)*
- *Self-similarity and non-Markovian streaming analysis*

This chapter focuses on transport mechanisms for the delivery of media streams with the highest possible quality. After reviewing how sampled and digitized streams

of voice are treated in networks, we look at two VoIP protocols—*Session Initiation Protocol* (SIP) and the *H.323 series of protocols*—and explain their session signaling and numbering. We then present *real-time media transport protocols*, by which a sender sends a stream of data at a constant rate. The most widely applied protocol for real-time transmission is the *Real-Time Transport Protocol* (RTP), which is available with its companion version, *Real-Time Control Protocol* (RTCP).

As detailed in our discussion on video streaming, a video in a single server can be streamed from a video server to a client for each client request. However, when a high-bit-rate video stream must pass through many Internet domains, a significant delay may result. We then present a solution by using *content distribution networks* (CDNs), which can bring multimedia content to users. The *Stream Control Transmission Protocol* (SCTP) provides a general-purpose protocol for transporting stream traffic. The chapter ends with a detailed streaming source modeling and analysis and a detailed traffic modeling of streaming sources, using *self-similarity* patterns.

18.1 Overview of IP Telephony

An IP *telephone* can be used to make telephone calls over IP networks. *Voice over IP* (VoIP), or IP telephony, uses packet-switched networks to carry voice traffic in addition to data traffic. The basic scheme of IP telephony starts with *pulse code modulation*, discussed in Chapter 17. The encoded data is transmitted as packets over packet-switched networks. At a receiver, the data is decoded and converted back to analog form. The packet size must be properly chosen to prevent large delays. The IP telephone system must also be able to handle the signaling function of the call setup, mapping of phone number to IP address, and proper call termination.

Basic components of an IP telephone system include IP *telephones*, the *Internet backbone*, and *signaling servers*, as shown in Figure 18.1. The IP telephone can also be a laptop or a computer with the appropriate software. An IP telephone connects to the Internet through a wired or a wireless medium. The signaling servers in each domain are analogous to the central processing unit in a computer and are responsible for the coordination between IP phones. The hardware devices required to deploy packet-switched networks are less expensive than those required for the connection-oriented public-switched telephone networks. On a VoIP network, network resources are shared between voice and data traffic, resulting in some savings and efficient use of the available network resources.

A VoIP network is operated through two sets of protocols: *signaling protocols* and *real-time packet-transport protocols*. Signaling protocols handle call setups and are

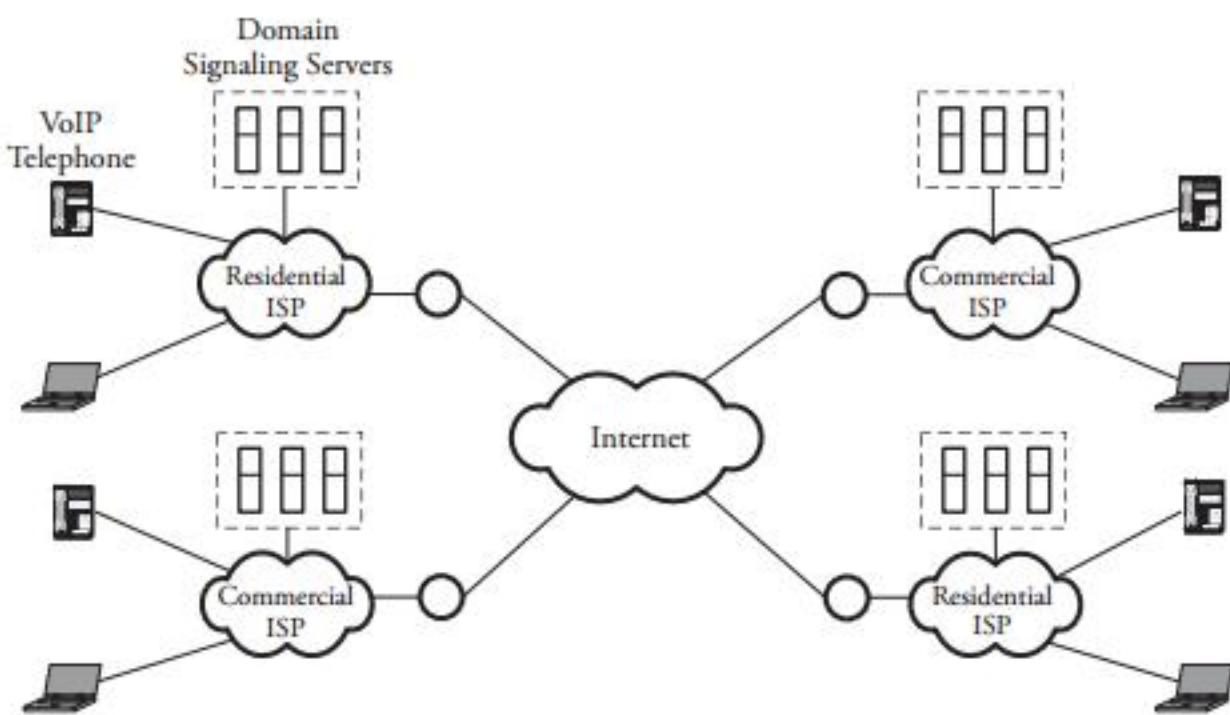


Figure 18.1 Voice over IP system

controlled by the signaling servers. Once a connection is set, RTP transfers voice data in real-time fashion to destinations. RTP runs over UDP because TCP has a very high overhead. RTP builds some reliability into the UDP scheme and contains a sequence number and a real-time clock value. The sequence number helps RTP recover packets from out-of-order delivery. Two RTP sessions are associated with each phone conversation. Thus, the IP telephone plays a dual role: an RTP sender for outgoing data and an RTP receiver for incoming data.

18.1.1 VoIP Quality-of-Service

A common issue that affects the QoS of packetized audio is *jitter*. Voice data requires a constant packet interarrival rate at receivers to convert data into a proper analog signal for playback. The variations in the packet interarrival rate lead to jitter, which results in improper signal reconstruction at the receiver. Typically, an unstable sine wave reproduced at the receiver results from the jitter in the signal. Buffering packets can help control the interarrival rate. The buffering scheme can be used to output the data packets at a fixed rate. The buffering scheme works well when the arrival time of the next packet is not very long. Buffering can also introduce a certain amount of delay.

Another issue having a great impact on real-time transmission quality is *network latency*, or delay, which is a measure of the time required for a data packet to travel

from a sender to a receiver. For telephone networks, a round-trip delay that is too large can result in an *echo* in the earpiece. Delay can be controlled in networks by assigning a higher priority for voice packets. In such cases, routers and intermediate switches in the network transport these high-priority packets before processing lower-priority data packets.

Congestion in networks can be a major disruption for IP telephony. Congestion can be controlled to a certain extent by implementing weighted random early discard, whereby routers begin to intelligently discard lower-priority packets before congestion occurs. The drop in packets results in a subsequent decrease in the window size in TCP, which relieves congestion to a certain extent.

A VoIP connection has several QoS factors:

- *Packet loss* is accepted to a certain extent.
- *Packet delay* is normally unacceptable.
- *Jitter*, as the variation in packet arrival time, is not acceptable after a certain limit.

Packet loss is a direct result of the queueing scheme used in routers. VoIP can use *priority queueing*, *weighted fair queuing*, or *class-based weighted fair queuing*, whereby traffic amounts are also assigned to classes of data traffic. Besides these well-known queueing schemes, voice traffic can be handled by a *custom queuing*, in which a certain amount of channel bandwidth for voice traffic is reserved.

Although the packet loss is tolerable to a certain extent, packet delay may not be tolerable in most cases. The variability in the time of arrival of packets in packet-switched networks gives rise to *jitter* variations. This and other QoS issues have to be handled differently than in conventional packet-switched networks. QoS must also consider connectivity of packet-voice environment when it is combined with traditional telephone networks.

18.2 VoIP Signaling Protocols

The IP telephone system must be able to handle signalings for call setup, conversion of phone number to IP address mapping, and proper call termination. Signaling is required for call setup, call management, and call termination. In the standard telephone network, signaling involves identifying the user's location given a phone number, finding a route between a calling and a called party, and handling the issue of call forwarding and other call features.

Layer	Protocol								
	SIP		H.323						
5	Other Signals	Media Transport	Registration	Media Transport		Security	Signaling	Data	
				Voice Codec	Video Codec				
4			H.225.0-RAS	G.711 H.263 G.722 G.723 G.728	H.261 H.323	H.235	H.225.9-Q.931 H.250 H.245	T.120	
3	IP, RSVP, and IGMP								
4	UDP					TCP			

Figure 18.2 Main protocols for VoIP and corresponding layers of operation

IP telephone systems can use either a distributed or a centralized signaling scheme. The distributed approach enables two IP telephones to communicate using a client/server model, as most Internet applications do. The distributed approach works well with VoIP networks within a single company. The centralized approach uses the conventional model and can provide some level of guarantee. Three well-known signaling protocols are

1. *Session Initiation Protocol (SIP)*
2. *H.323 protocols*
3. *Media Gateway Control Protocol (MGCP)*

Figure 18.2 shows the placement of VoIP in the five-layer TCP/IP model. SIP, H.323, and MGCP run over TCP or UDP; real-time data transmission protocols, such as RTP, typically run over UDP. Real-time transmissions of audio and video traffic are implemented over UDP, since the real-time data requires lower delay and less overhead. Our focus in this chapter is on the two signaling protocols SIP and H.323 and on RTP and RTCP.

18.2.1 Session Initiation Protocol (SIP)

The *Session Initiation Protocol (SIP)* is one of the most important VoIP signaling protocols operating in the application layer in the five-layer TCP/IP model. SIP can

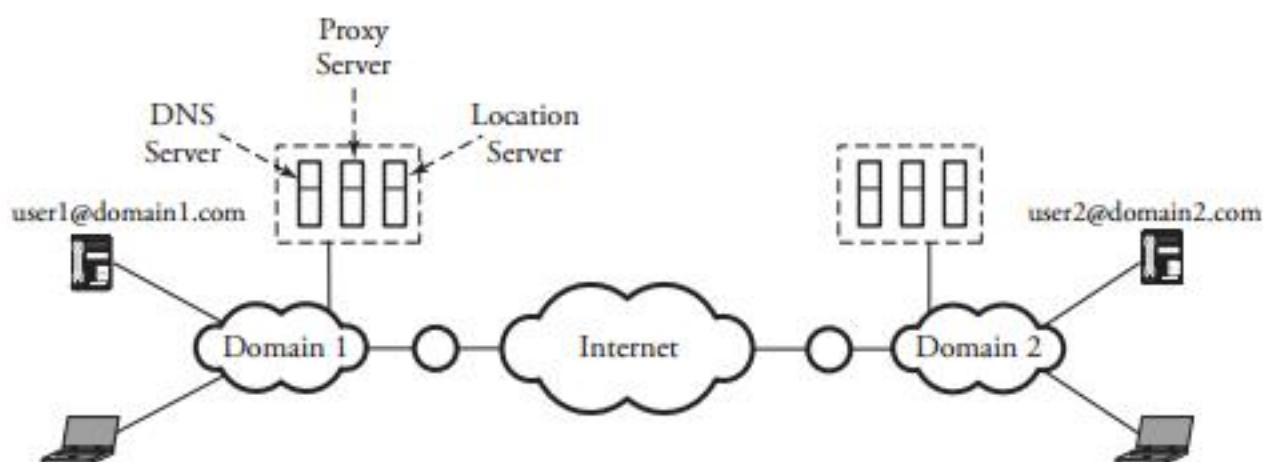


Figure 18.3 Overview of SIP

perform both unicast and multicast sessions and supports user mobility. SIP handles signals and identifies user location, call setup, call termination, and busy signals. SIP can use multicast to support conference calls and uses the *Session Description Protocol* (SDP) to negotiate parameters.

SIP Components

Figure 18.3 shows an overview of SIP. A call is initiated from a *user agent*: the user's IP telephone system, which is similar to a conventional phone. A user agent assists in initiating or terminating a phone call in VoIP networks. A user agent can be implemented in a standard telephone or in a laptop with a microphone that runs some software. A user agent is identified using its associated domain. For example, `user1@domain1.com` refers to user 1, who is associated with the `domain1.com` network.

SIP consists of the following five servers:

1. *DNS server*. The *Domain Name System* (DNS) server maps the domain name to an IP address in the *user information database* (UID). The UID database contains such user information as preferences and the services to which it has subscribed. The UID also stores information on the related IP addresses. Each user is normally configured with more than one DNS server.
2. *Proxy server*. The proxy server forwards requests from a user agent to a different location and handles authorizations by checking whether the caller is authorized to make a particular call.
3. *Location server*. This server is responsible for UID management. The location server interacts with the database during call setup. Each proxy server is normally configured with more than one location server.

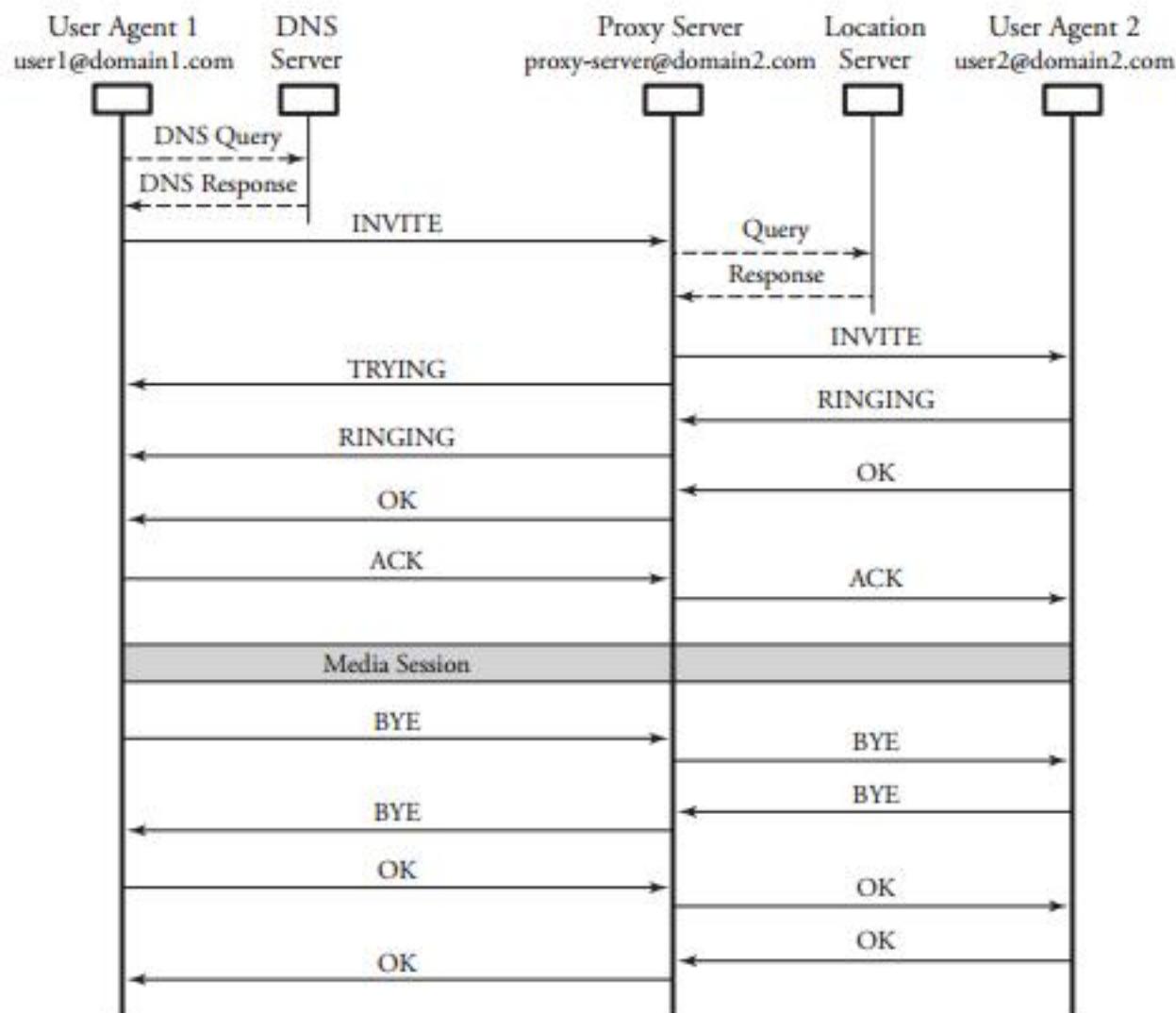


Figure 18.4 Overview of SIP signaling

4. *Redirect server.* This server performs call forwarding and provides alternative paths for the user agent.
5. *Registrar server.* This server is responsible for registering users in the system and updating the UID that the location server consults. Requests for registration must be authenticated before the registration is granted.

Session Signaling and Numbering

Figure 18.4 shows a SIP session between user agents 1 and 2: `user1@domain1.com` and `user2@domain2.com`, respectively. In this example, user 1 places a call to contact user 2. User 1 first communicates with its DNS server to map the domain name to an IP address in the SIP UID. The DNS server then communicates with the proxy server of the called party. At this point, user 1 has resolved the name of `user2@domain2.com` into an IP address through DNS query and response.

SIP uses IP addresses for numbering. Locating users in an integrated network consisting of different networks is a complex task. In addition, an optimal route to the user after the user is located must be found. The location servers use the *Telephone Routing Over IP* (TRIP) protocol to locate a user in an integrated network. TRIP advertises routes, exchanges routing information, and divides the globe into *IP telephone administrative domains* (ITAD). Location servers exchange information about routes with signaling gateways connecting to various ITADs.

The first signal to establish this call is the INVITE message, which is used for session creation. The INVITE message contains such information fields as *from*, *to*, *via*, and *call-id*, in addition to routing information. The proxy server in turn communicates with a location server of the called party. The end point of the called party, user2@domain2.com, is sent an INVITE message to initiate a session. Once user 2 receives a connection query, the TRYING signal is propagated to user 1 from the proxy server, indicating that the call is being routed. This signal is also used to keep track of the call process. A RINGING signal is transmitted from user 2 back to user 1.

When user 2 accepts the call, an OK signal is issued back to user 1 to indicate that the called party has accepted the call. This last signal is acknowledged by an ACK message without any response. The called party picks up the phone, and the two IP phones communicate directly through a *media session*, using the real-time protocol, as shown in the figure. At the end of the conversation, the BYE message is used for a session termination, ending the call.

A few other message signals can be used for other services. The CANCEL message is used to cancel a pending request. The REGISTER message is used to register the user's location. After registration, the user can be reached at a specific URL. Finally the OPTIONS message is used to learn about the parameters used by the called party.

18.2.2 H.323 Protocols

The H.323-series protocols are implemented in layer 5 of the TCP/IP model and run over either TCP or UDP. The H.323 protocols interact to provide ideal telephone communication, providing phone numbers to IP address mapping, handling digitized audio streaming in IP telephony, and providing signaling functions for call setup and call management. The H.323 series supports simultaneous voice and data transmission and can transmit binary messages that are encoded using *basic encoding rules*. From the security standpoint, the H.323 scheme provides a unique framework for security, user authentication, and authorization and supports conference calls and multipoint connections, as well as accounting and call-forwarding services.

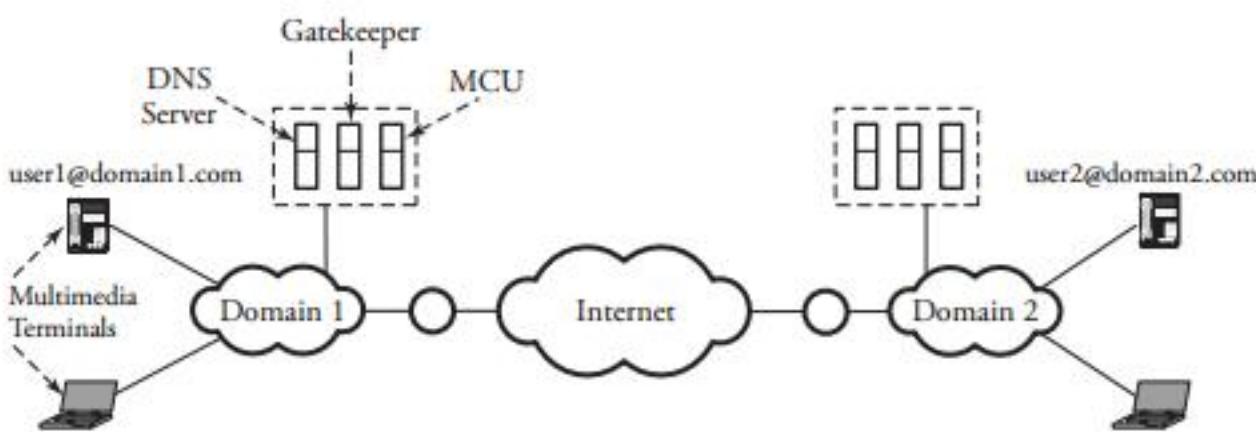


Figure 18.5 Overview of H.323 protocol connection

H.323 Components

Figure 18.5 shows an overview of the H.323 protocol connections. The H.323 scheme defines the following five components:

1. *Multimedia terminal.* A multimedia terminal is designed to support video and data traffic and to provide support for IP telephony.
2. *DNS server.* As in SIP, a DNS server maps a domain name to an IP address.
3. *Gateway.* The gateway is a router that serves as an interface between the IP telephone system and the traditional telephone network.
4. *Gatekeeper:* The gatekeeper is the control center that performs all the location and signaling functions. The gatekeeper monitors and coordinates the activities of the gateway. The gateway also performs some signaling functions.
5. *Multicast or multipoint control unit (MCU).* This unit provides some multipoint services, such as conference calls.

A gatekeeper can send signals to other gatekeepers of different zones to access users of those domains. This feature supports distributed locations of multinational systems.

Session Signaling and Numbering

Figures 18.6 and 18.7 show the details of two IP telephone communications using H.323-series protocols. Assume that two user agents 1 and 2: user1@domain1.com and user2@domain2.com, respectively. In this example, user 1 places a call to contact user 2.

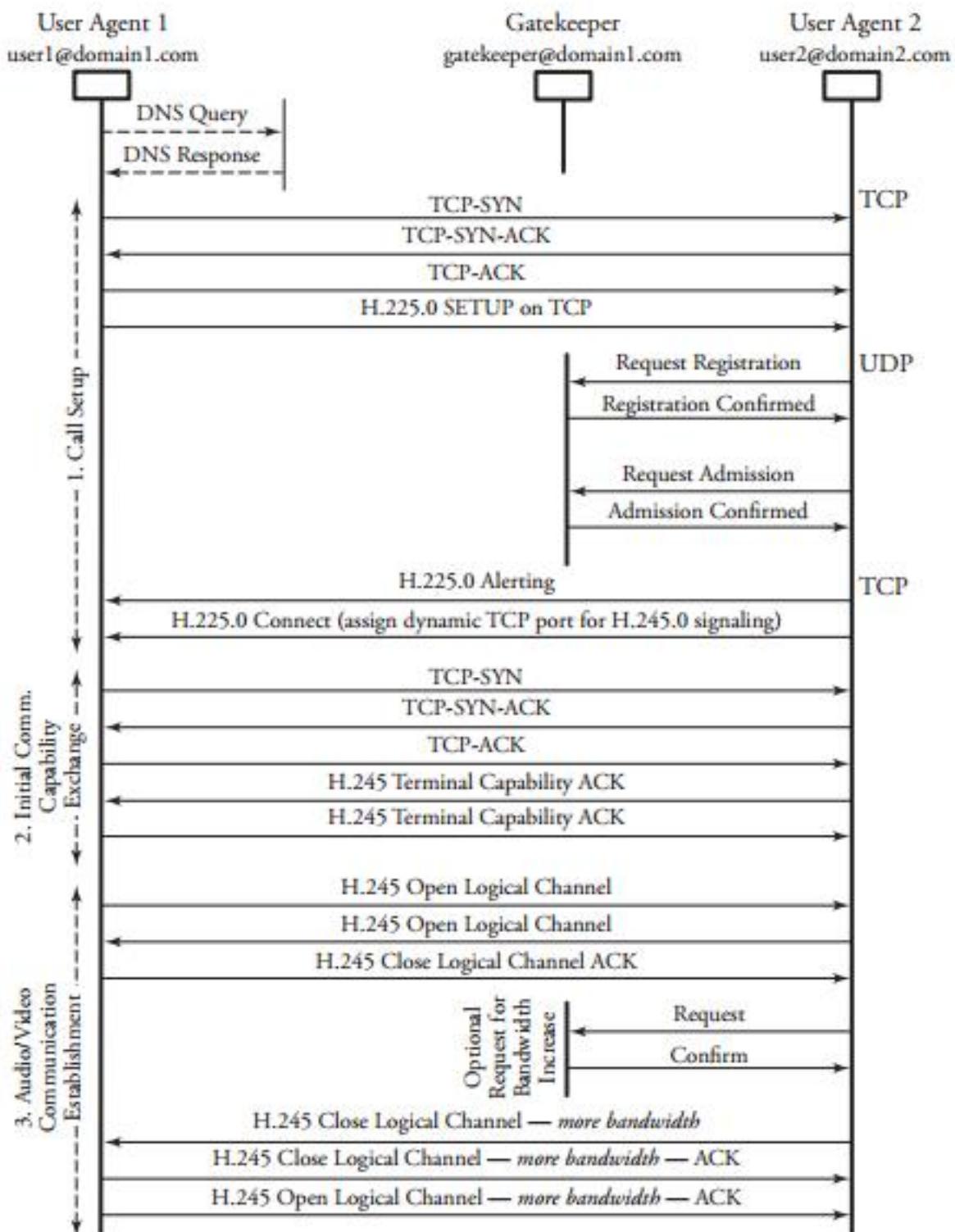


Figure 18.6 H.323 protocol signaling: steps 1, 2, and 3

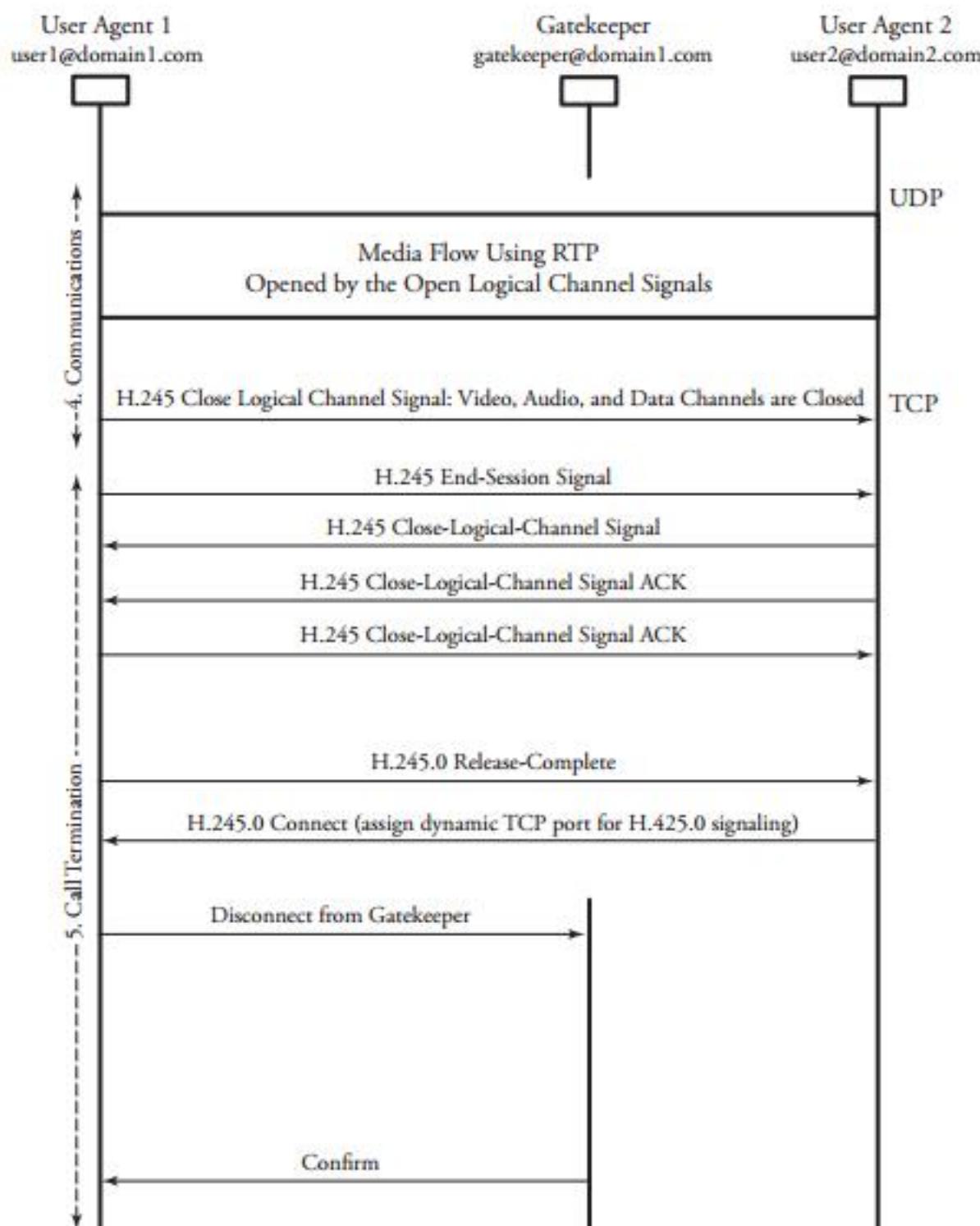


Figure 18.7 H.323 protocol signaling: steps 4 and 5

First, user 1 communicates with its DNS server, as explained earlier. The signaling uses both UDP and TCP, and partitioned into the following five steps:

1. Call setup
2. Initial communication capability
3. Audio/video communication establishment
4. Communications
5. Call termination

At step 1, when user1 dials user 2's telephone number, the first set of signals are exchanged between these two users in conjunction with opening a TCP connection. TCP-SYN, TCP-SYN-ACK, and then TCP-ACK signals are generated between the two users. At this point, the H225.0 SETUP ON TCP signal informs the called party that the connection can be set up on TCP. The users can now request a certain bandwidth from the associated gatekeeper server of the called party. The requested bandwidth is granted if sufficient bandwidth is left over on the connection; otherwise, the call has to find another gatekeeper to register with. This phase is handled by H.225 and H.245 protocols.

At step 2, all the end points' communication capabilities available over TCP are exchanged. This phase is necessary because the type of communication service requested depends on the communication capabilities of both end points. Step 3 implements the establishment of a logical channel, which in H.323 is unidirectional; therefore, a logical channel must be established in either direction in order to have two-way communications. At the end of this phase, two end points are set for communications. Meanwhile, more bandwidth can also be requested, as shown in the figure, before communications start.

Step 4 comprises the communications between the two users. This phase is handled using RTP over UDP. At this step, any kind of media flow can be considered, depending on the size and type of the channel established in step 4. Finally at step 5, the call is terminated by either user. In Figure 18.7, user 2 initiates the termination of the call by closing the logical channel in H.245 and disconnecting the call from the gatekeeper.

18.3 Real-Time Media Transport Protocols

In real-time applications, a stream of data is sent at a constant rate. This data must be delivered to the appropriate application on the destination system, using real-time protocols. The most widely applied protocol for real-time transmission is the *Real-Time*

Transport Protocol (RTP), including its companion version: *Real-Time Control Protocol* (RTCP).

UDP cannot provide any timing information. RTP is built on top of the existing UDP stack. Problems with using TCP for real-time applications can be identified easily. Real-time applications may use multicasting for data delivery. As an end-to-end protocol, TCP is not suited for multicast distribution. TCP uses a retransmission strategy for lost packets, which then arrive out of order. Real-time applications cannot afford these delays. TCP does not maintain timing information for packets. In real-time applications, this would be a requirement.

18.3.1 Real-Time Transport Protocol (RTP)

The *real-time transport protocol* (RTP) provides some basic functionalities to real-time applications and includes some specific functions to each application. RTP runs on top of the transport protocol as UDP. As noted in Chapter 8, UDP is used for port addressing in the transport layer and for providing such transport-layer functionalities as reordering. RTP provides application-level framing by adding application-layer headers to datagrams. The application breaks the data into smaller units, called *application data units* (ADUs). Lower layers in the protocol stack, such as the transport layer, preserve the structure of the ADU.

Real-time applications, such as voice and video, can tolerate a certain amount of packet loss and do not always require data retransmission. The mechanism RTP uses typically informs a source about the quality of delivery. The source then adapts its sending rate accordingly. If the rate of packet loss is very high, the source might switch to a lower-quality transmission, thereby placing less load on the network. A real-time application can also provide the data required for retransmission. Thus, recent data can be sent instead of retransmitted old data. This approach is more practical in voice and video applications. If a portion of an ADU is lost, the application is unable to process the data, and the entire ADU would have to be retransmitted.

Real-Time Session and Data Transfer

The TCP/IP and OSI models divide the network functionalities, based on a layered architecture. Each layer performs distinct functions, and the data flows sequentially between layers. The layered architecture may restrict the implementation on certain functions out of the layered order. *Integrated layer processing* dictates a tighter coupling among layers. RTP is used to transfer data among *sessions* in real time. A session is a

logical connection between an active client and an active server and is defined by the following entities:

- *RTP port number*, which represents the destination port address of the RTP session. Since RTP runs over UDP, the destination port address is available on the UDP header.
- *IP address* of the RTP entity, which involves an RTP session. This address can be either a unicast or a multicast address.

RTP uses two relays for data transmission. A *relay* is an intermediate system that acts as both a sender and a receiver. Suppose that two systems are separated by a firewall that prevents them from direct communication. A relay in this context is used to handle data flow between the two systems. A relay can also be used to convert the data format from a system into a form that the other system can process easily. Relays are of two types: *mixer* and *translator*.

A *mixer relay* is an RTP relay that combines the data from two or more RTP entities into a single stream of data. A mixer can either retain or change the data format. The mixer provides timing information for the combined stream of data and acts as the source of timing synchronization. Mixers can be used to combine audio streams in real-time applications and can be used to service systems that may not be able to handle multiple RTP streams.

The *translator* is a device that generates one or more RTP packets for each incoming RTP packet. The format of the outgoing packet may be different from that of the incoming packet. A *translator relay* can be used in video applications in which a high-quality video signal is converted to a lower-quality in order to service receivers that support a lower data rate. Such a relay can also be used to transfer packets between RTP entities separated by an application-level firewall. Translators can sometimes be used to transfer an incoming multicast packet to multiple destinations.

RTP Packet Header

RTP contains a fixed header and an application-specific variable-length header field. Figure 18.8 shows the RTP header format. The RTP header fields are:

- *Version (V)*, a 2-bit field indicating the protocol version.
- *Padding (P)*, a 1-bit field that indicates the existence of a padding field at the end of the payload. Padding is required in applications that require the payload to be a multiple of some length.
- *Extension (X)*, a 1-bit field indicating the use of an extension header for RTP.

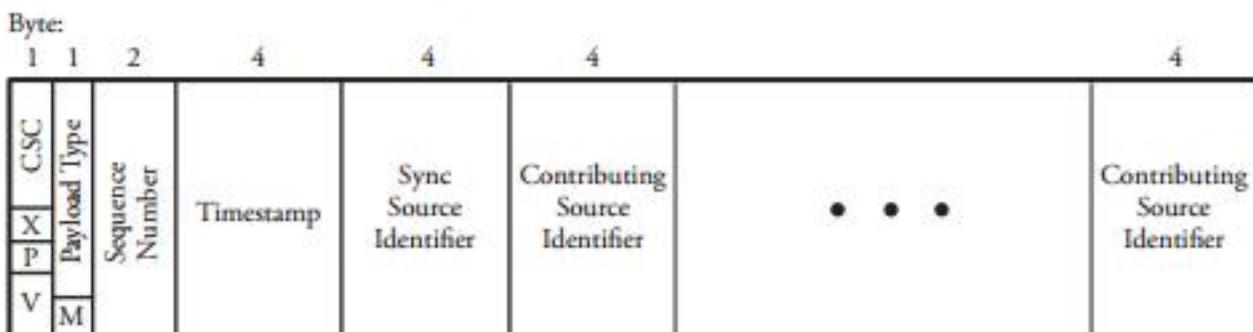


Figure 18.8 Packet format for the real-time transport protocol

- *Contributing source count* (CSC), a 4-bit field that indicates the number of contributing source identifiers.
- *Marker* (M), a 1-bit field indicating boundaries in a stream of data traffic. For video applications, this field can be used to indicate the end of a frame.
- *Payload type*, A 7-bit field specifying the type of RTP payload. This field also contains information on the use of compression or encryption.
- *Sequence number*, a 16-bit field that a sender uses to identify a particular packet within a sequence of packets. This field is used to detect packet loss and for packet reordering.
- *Timestamp*, a 32-bit field enabling the receiver to recover timing information. This field indicates the timestamp when the first byte of data in the payload was generated.
- *Synchronization source identifier*, a randomly generated field used to identify the RTP source in an RTP session.
- *Contributing source identifier*, an optional field in the header to indicate the contributing sources for the data.

Overall, the main segment of an RTP header includes 12 bytes and is appended to a packet being prepared for multimedia application.

18.3.2 Real-Time Control Protocol (RTCP)

The Real-Time Transport Protocol (RTCP) also runs on top of UDP. RTCP performs several functions, using multicasting to provide feedback about the data quality to all session members. The session multicast members can thus get an estimate of the

performance of other members in the current active session. Senders can send reports about data rates and the quality of data transmission. Receivers can send information about packet-loss rate, jitter variations, and any other problems they might encounter. Feedback from a receiver can also enable a sender to diagnose a fault. A sender can isolate a problem to a single RTP entity or a global problem by looking at the reports from all receivers.

RTCP performs *source identification*. RTCP packets contain some information to identify the source of the control packet. The rate of RTCP packets must also be kept to less than 5 percent of the total session traffic. Thus, this protocol carries out “rate control” of RTCP packets. At the same time, all session members must be able to evaluate the performance of all other session members. As the number of active members in a session increases, the transmission rates of the control packets must be reduced. RTCP is also responsible for *session control* and can provide some session-control information, if necessary.

Packet Type and Format

RTCP transmits control information by combining a number of RTCP packets in a single UDP datagram. The RTCP packet types are *sender reports* (SR), *receiver reports* (RR), *source descriptor* (SDES), *goodbye* (BYE), and *application-specific types*. Figure 18.9 shows some RTCP packet formats. The fields common to all packet types are as follows:

- *Version*, a 2-bit field that indicates the current version.
- *Padding*, a 1-bit field that indicates the existence of padding bytes at the end of the control data.
- *Count*, a 5-bit field that indicates the number of SR or RR reports or the number of source items in the SDES packet.
- *Packet type*, an 8-bit field that indicates the type of RTCP packet. (Four RTCP packet types were specified earlier.)
- *Length*, a 16-bit field that represents the length of packet in 32-bit words minus 1.
- *Synchronization source identifier*, a field common to the SR and RR packet types; it indicates the source of the RTCP packets.

Figure 18.9 also shows a typical format of a sender report. The report consists of the common header fields and a block of sender information. The sender report may

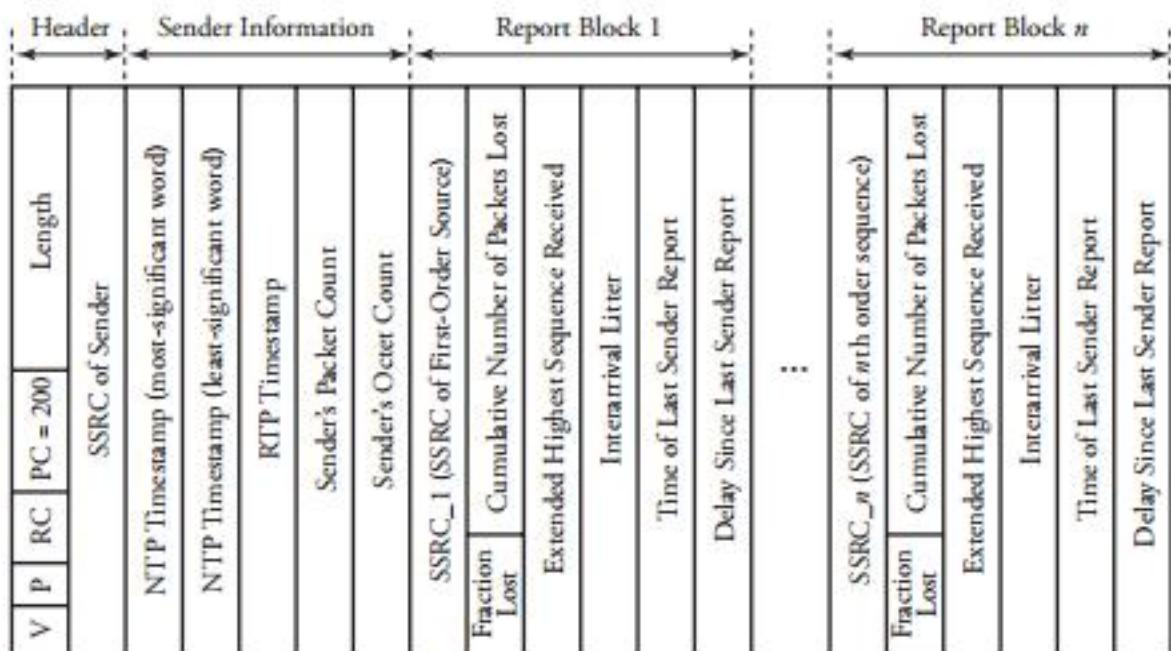


Figure 18.9 Format of the SR packet in RCTP

also contain zero or more receiver report blocks, as shown in the figure. The fields in the sender information block are:

- *NTP timestamp*, a 64-bit field that indicates when a sender report was sent. The sender can use this field in combination with the timestamp field returned in receiver reports to estimate the round-trip delay to receivers.
- *RTP timestamp*, a 32-bit field used by a receiver to sequence RTP data packets from a particular source.
- *Sender's packet count*, a 32-bit field that represents the number of RTP data packets transmitted by the sender in the current session.
- *Sender's byte count*, a 32-bit field that represents the number of RTP data octets transmitted by the sender in the current session.

The SR packet includes zeros or more RR blocks. One receiver block is included for each sender from which the member has received data during the session. The RR block includes the following fields:

- *SSRC_n*, a 32-bit field that identifies the source in the report block, where *n* is the number of sources.
- *Fraction lost*, an 8-bit field indicating the fraction of data packet loss from source *SSRC_n* since the last SR or RR report was sent.

- *Cumulative number of packets lost*, a 24-bit field that represents the total number of RTP data packets lost from the source in the current active session identified by SSRC_n.
- *Extended highest sequence number received*, the first 16 least-significant bits, used to indicate the highest sequence number for packets received from source SSRC_n. The first 16 most-significant bits indicate the number of times that the sequence number has been wrapped back to zero.
- *Interarrival jitter*, a 32-bit field used to indicate the jitter variations at the receiver for the source SSRC_n.
- *Last SR timestamp*, a 32-bit field indicating the timestamp for the last SR packet received from the source SSRC_n.
- *Delay since last SR*, a 32-bit field indicating the delay between the arrival time of the last SR packet from the source SSRC_n and the transmission of the current report block.

Receivers in RTCP can provide feedback about the quality of reception through a receiver report. A receiver that is also a sender during a session, it also sends the sender reports.

18.3.3 Estimation of Jitter in Real-Time Traffic

The jitter factor is a measure of the delay experienced by RTP packets in a given session. The average jitter can be estimated at the receiver. Let us define the following parameters at the receiver:

- t_i Timestamp of the RTP data packet i indicated by the source.
- a_i Arrival time of the RTP data packet i at the receiver.
- d_i Measure of difference between interarrival time of RTP packets at receiver and the one for packet departure from the source. This value represents the difference in packet spacing at source and receiver.
- $E[i]$ Estimate of average jitter until the time of packet i arrival.

The difference interval d_i is given by

$$d_i = (a_i - a_{i-1}) - (t_i - t_{i-1}). \quad (18.1)$$

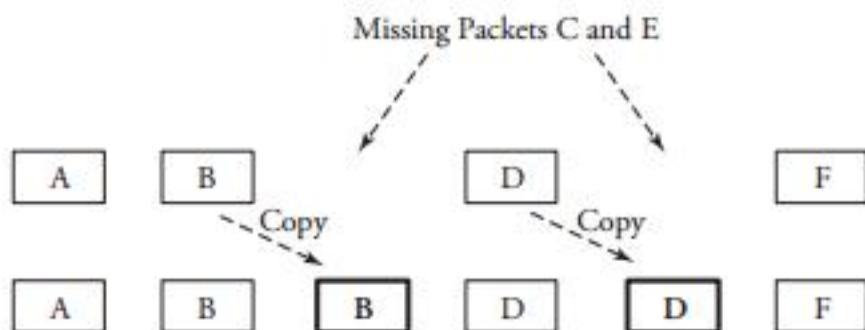


Figure 18.10 Missing voice packets and reconstructing the data stream

The estimated average jitter until the time of packet i arrival is given by

$$E[i] = k(E[i - 1] + |d_i|), \quad (18.2)$$

where k is a normalizing coefficient. The interarrival jitter value indicated in the sender report provides useful information on network conditions to the sender and the receiver. The jitter value can be used as an estimate to calculate the variation of network congestion.

RTP packet-sequence numbers are used to help a receiver sequence packets in order to recover lost packets. When packets arrive out of order at the receiver, the sequence number can be used to assemble data packets. Consider Figure 18.10. When certain packets are lost, the gaps are filled in with previously received data packets. As shown in the figure, packet D is replayed twice, since packet C is lost. This mechanism can help reduce the pips and clicks in voice signals owing to lost packets. This reconstructed data stream is sent to the receiver, with the lost packets replaced by previously received packets. This can significantly improve the latency.

18.4 Distributed Multimedia Networking

Video streaming presents a significant challenge to network designers. A video in a single server can be streamed from a video server to a client at the client request. The high-bit-rate video streaming must sometimes pass through many Internet service providers, leading to the likelihood of significant delay and loss on video. One practical solution to this challenge is to use *content distribution networks* (CDNs) for distributing stored multimedia content.

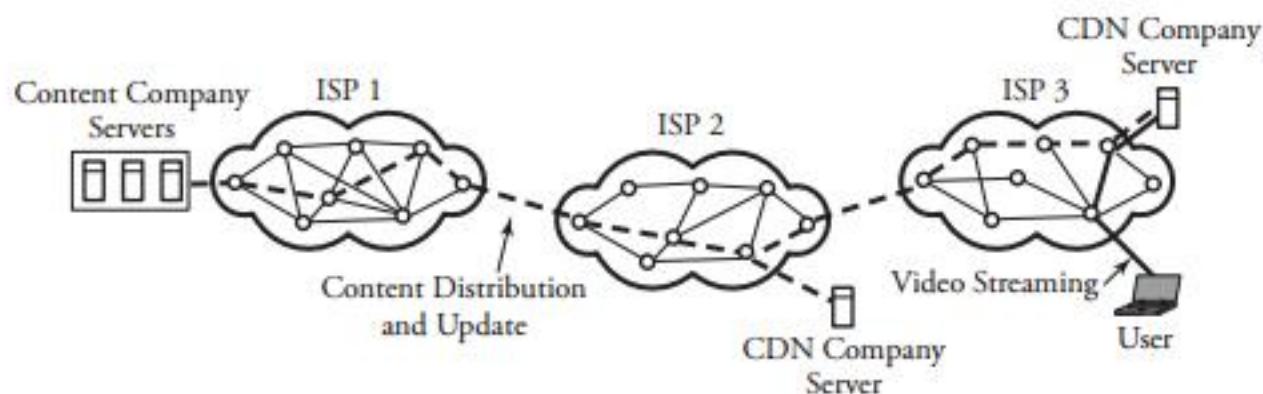


Figure 18.11 Video streaming provided to a user in ISP domain 3 through a branch of a CDN company in the same domain while it interacts with its other servers in ISP 2 and ISP 1 domains. The source of the requested content is updated by the content company in ISP 1 domain.

18.4.1 Content Distribution Networks (CDNs)

A *content distribution network* (CDN) is a group of proxy servers located at a certain strategic location around the Internet service provider. CDNs ensure that a download request can always be handled from the nearest server. With CDNs, the content of streaming is geographically brought to a user unable to access the content at the desired data rate in the original location. Therefore, a user deals with a *content provider*, such as private TV broadcast companies, not ISPs. The content company hires a CDN company to deliver its content (streaming video) to the user. This does not mean that a CDN company cannot expose its server to ISPs.

A CDN company has several CDN centers around the Internet. Each group of servers is installed in proximity to ISP access points. At the request of a user, the content is provided by the closest CDN server that can best deliver the content. Figure 18.11 shows video streaming being provided to a user in ISP 3 domain through a branch of a CDN company in the same domain while this company interacts with its other servers in ISP 2 and ISP 1 domains. The source of the requested content is updated by the content company, which in this case is located in ISP 1 domain.

18.4.2 CDN Interactions with DNS

CDNs use Domain Name System (DNS) servers to direct browsers to the correct server. For example, assume that the URL of a content company is www.contentco.com, as shown in Figure 18.12. Assume that this company has produced a video movie named `movie.mpg` with MPEG format and that the content of this movie is placed on one

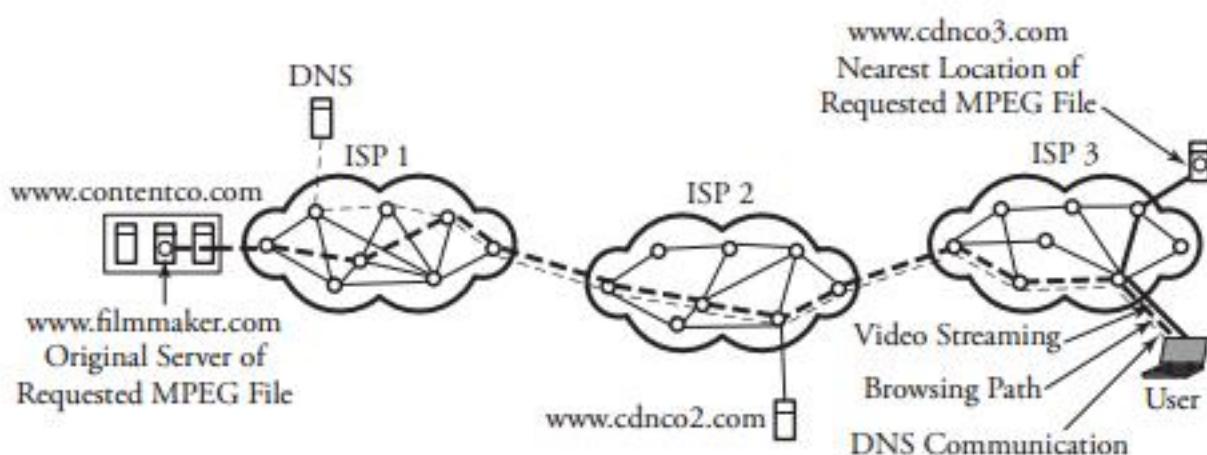


Figure 18.12 Video streaming provided to a user in ISP 3 domain using domain name system (DNS) servers

of its servers, named www.filmmaker.com. Therefore, this MPEG file has a reference as www.filmmaker.com/movies/movie.mpg. Also, suppose that a CDN company interacting with the content company is accessed through www.cdnco.com. The content company replaces this reference with one affected by the CDN company's URL domain name as www.cdnco.com/www.filmmaker.com/movies/movie.mpg. Now, when a user like the one in Figure 18.12 requests the movie.mpg video object, its browser sends its request for the base object to the origin server, www.filmmaker.com. Then the browser finds the object at www.cdnco.com/www.filmmaker.com/movies/movie.mpg.

Now, the browser knows that www.cdnco.com is also involved and requests the corresponding DNS server for the location of the CDN company. The DNS server is set up to transfer all movie queries about www.contentco.com to the corresponding DNS server. Finally, the corresponding DNS server extracts the IP address of the requesting browser and returns the IP address of the closest CDN server to the browser, located in the ISP 3 domain in the previous example. Consequently, the look-up table of the DNS server finds this best (nearest) CDN's server URL at www.cdnco3.com/www.filmmaker.com/movies/movie.mpg. Note that the CDN's link for this example is www.cdnco3.com, corresponding to the CDN company's server in the ISP 3, as seen in the figure. This way, the user communicates with the CDN server located nearby with minimal congestion.

18.4.3 Providing QoS to Streaming

Consider Figure 18.13, in which three users—User A, User B, and User C—all belong to a local area network and are using the Internet simultaneously. User A is contacting

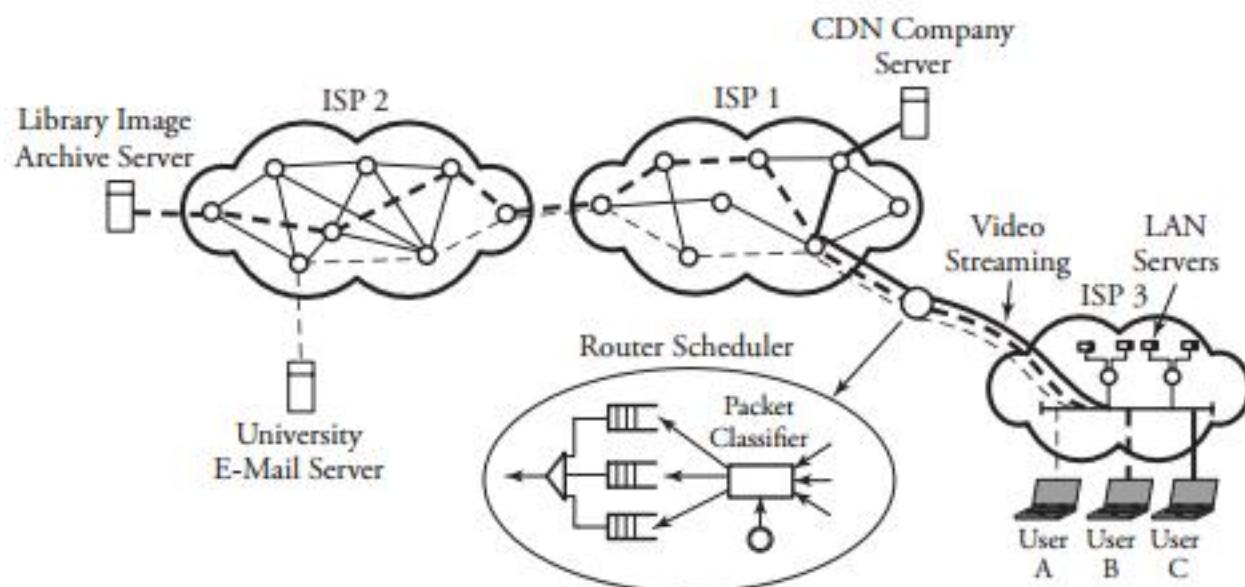


Figure 18.13 Providing QoS at the main router connecting to an Internet service provider

a university e-mail server and requires small bandwidth. User B is searching images in a library archive server and requires a non-real-time but modest bandwidth. User C is using a video streaming service from a CDN server and requires substantial high-quality bandwidth. Packet flows from these users need to be scheduled as shown in the main router between the LAN and ISP 1.

Video streaming, e-mail, and image packets in the best-effort Internet are mixed in the output queue of the main exit router of a domain. Under such circumstances, a burst of packets, primarily from the image file source, could cause IP video streaming packets to be excessively delayed or lost at the router. One solution in this case is to mark each packet as to which class of traffic it belongs to. This can be done by using the type-of-service (ToS) field in IPv4 packets. As seen in the figure, transmitted packets are first classified in terms of their priorities and are queued in a first in, first out (FIFO) order. The priority of an image file can be equal to or less than the one for video streaming, owing to the arrangement of purchased services.

18.5 Stream Control Transmission Protocol (SCTP)

The *Stream Control Transmission Protocol* (SCTP) provides a general-purpose transport protocol for message-oriented applications. It is a reliable transport protocol for transporting stream traffic, can operate on top of unreliable connectionless networks, and

offers acknowledged and nonduplicated transmission data on connectionless networks (datagrams). SCTP has the following features.

- The protocol is error free. A retransmission scheme is applied to compensate for loss or corruption of the datagram, using checksums and sequence numbers.
- It has ordered and unordered delivery modes.
- SCTP has effective methods to avoid flooding congestion and masquerade attacks.
- This protocol is *multipoint* and allows several streams within a connection.

In TCP, a stream is a sequence of bytes; in SCTP, a sequence of variable-sized messages. SCTP services are placed at the same layer as TCP or UDP services. Streaming data is first encapsulated into packets, and each packet carries several correlated chunks of streaming details. If an MPEG movie is displayed live over the Internet, a careful assignment of data per packet is required. An MPEG video consists of frames, each consisting of $n \times m$ blocks of pixels, with each pixel normally an 8×8 matrix. In this case, each block of pixels can be encapsulated into a chunk, where each row of the block is formatted as a packet.

18.5.1 SCTP Packet Structure

Figure 18.14 shows the structure of streaming packets used in SCTP. An SCTP packet is also called a *protocol data unit* (PDU). As soon as the streaming data is ready to be transmitted over IP, an SCTP packet forms the payload of an IP packet. Each packet consists of a *common header* and *chunks*. The streaming data is distributed over packets, and each packet carries correlated “chunks” of streaming data. Multiple chunks representing multiple portions of streaming information are in fact multiplexed into one packet up to the path-maximum packet size.

A chunk header starts with a chunk *type* field used to distinguish data chunks and any other types of control chunks. The *type* field is followed by a *flag* field and a chunk *length* field to indicate the chunk size. A chunk, and therefore a packet, may contain either control information or user data. The common header begins with the *source port number* followed by the *destination port number*. SCTP uses the same port concept as TCP or UDP does. A 32-bit *verification tag* field is exchanged between the end-point servers at startup to verify the two servers involved. Thus, two tag values are used in a connection. The common header consists of 12 bytes. SCTP packets are protected by a 32-bit checksum. The level of protection is more robust than the 16-bit checksum of TCP and UDP.

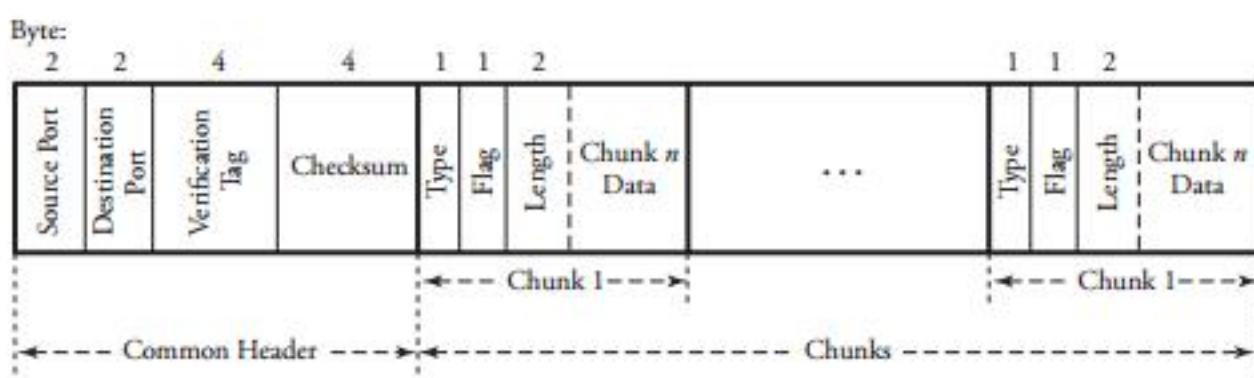


Figure 18.14 The structure of packets in the stream control transmission protocol (SCTP). Streaming data is encapsulated into packets and each packet carries several correlated chunks of streaming details.

Each packet has n chunks, and each chunk is of two types: *payload data chunk* for transmitting actual streaming data and *control chunks* for signaling and control. Signaling and control chunks are of several different types, as follows:

- *Initiation*, to initiate an SCTP session between two end points
- *Initiation acknowledgment*, to acknowledge the initiation of an SCTP session
- *Selective acknowledgment*, to be transmitted to a peer end point to acknowledge received data chunks
- *Heartbeat request*, to probe the reachability of a particular destination transport address defined in the session
- *Heartbeat acknowledgment*, to respond to the heartbeat request chunk
- *Abort*, to close a session
- *Shutdown*, to initiate a graceful close of a session
- *Shutdown acknowledgment*, to acknowledge receipt of the shutdown chunk once the shutdown process is completed
- *Operation error*, to notify the other party of a certain error
- *State cookie*, sent by the source to its peer to complete the initialization process
- *Cookie acknowledgment*, to acknowledge receipt of a state cookie chunk
- *Shutdown complete*, to acknowledge receipt of the shutdown acknowledgment chunk at the completion of the shutdown process

SCTP can easily and effectively be used to broadcast live video clips or full-color video movies. The SCTP exercises at the end of this chapter explore SCTP further.

18.6 Self-Similarity and Non-Markovian Streaming Analysis

Multimedia applications are delay sensitive and loss tolerant, unlike static-content communications, which are delay and loss intolerant. Distributed multimedia networks must be able to support the exchange of multiple types of information, such as voice, video, and data among users while also satisfying the performance requirements of each individual application. Consequently, the expanding diversity of high-bandwidth communication applications calls for a unified, flexible, and efficient network to prevent any congestion.

A network handling heavy video streaming must reserve some resources based on the source QoS. With reservation techniques, lossless transmission is guaranteed for the entire duration of the block; otherwise, the block is lost. However, sudden changes in the total volume of traffic at a node can impact the performance of streaming transmission. Multimedia networks are expected to support a large number of *bursty sources* with different characteristics. This fact enforces the use of processes other than Poisson for describing network traffic. The aggregated arrivals of packets are assumed to form *stream batches* of packets.

18.6.1 Self-Similarity with Batch Arrival Models

Section 11.5.3 explained that some traffic patterns indicate significant “burstiness,” or variations on a wide range of timescales. Bursty traffic, such as a video stream, can be viewed as a *batch* of traffic units and described statistically using *self-similarity patterns*. In a self-similar process, that packet loss and delay behavior are different from those in traditional network models using Poisson models. Self-similar traffic can be constructed by multiplexing a large number of ON/OFF sources that have ON and OFF intervals. This mechanism corresponds to a network of streaming servers, each of which is either silent or transferring video stream at a constant rate. Using this traffic, the distributions of transmission times and quiet times for any particular session are *heavy tailed*, which is an essential characteristic of traffic self-similarity.

The discrete-time representation of a communication system is the natural way to capture its behavior. In most communication systems, the input process to a queue is not *renewal* but correlated. A renewal process is a process in which the interval between consecutive occurrences of a certain event are independently and identically distributed. For example, the Poisson process is a renewal case with an exponential distribution. In a practical environment, the input process of packetized voice, data, or video traffic to a multiplexer does not form a renewal process but is bursty and correlated.

In streaming-traffic analysis, a batch of packets may arrive simultaneously. With the relatively more accurate traffic model being presented here, maximum and average traffic rates over *a number of given intervals* are determined. Using maximum rate and average rate parameters in our analysis can be efficient in capturing the burstiness characteristics of streaming sources. This method is especially effective for realistic sources, such as a compressed streaming video. In such situations, a source can even transmit at its peak rates when sending its large-size frames immediately followed by smaller frames. In the performance analysis, the packet-loss probability and the impact of increase in switching speed to link-speed ratio on the throughput are of particular interest.

In this analysis, consider a small buffered multiplexer or router, as large queuing delays are not expected in a multimedia network with a real-time transmission. A bursty arrival is modeled as a *batch*, or packets with identical interarrivals. This model captures the multirate burstiness characteristic of realistic sources. One property of self-similarity is that an object as an image is preserved with respect to scaling in space or time. In this environment, the traffic-relational structure remains unchanged at varying timescales. For any time $t > 0$ and a real number $a > 0$, a self-similar process, $X(t)$, is a continuous-time stochastic (random) process with parameter $0.5 < H < 1$ if it satisfies

$$X(t) = \frac{X(at)}{a^H}, \quad (18.3)$$

where parameter H is known as the *Hurst parameter*, or *self-similarity parameter*. The Hurst parameter is an important factor in bursty traffic, representing a measure of the dependence length in a burst. The closer H is to its maximum, 1, the greater the persistence of long-range dependence.

The expected values of both sides in Equation (18.3) must then be related as

$$E[X(t)] = \frac{E[X(at)]}{a^H}. \quad (18.4)$$

This result indicates that a self-similar process when $H = 0.5$ can also be obtained from the *Brownian random process* discussed in Section C.4.2. Based on Equation (C.34), and for any time increment δ , the increment of process, $X(t + \delta) - X(t)$, has the following distribution:

$$P[X(t + \delta) - X(t) \leq x] = \frac{1}{\sqrt{2\pi\delta}} \int_{-\infty}^x e^{-y^2/2\delta} dy. \quad (18.5)$$

To better understand the behavior of aggregated batch sequences of traffic, we can also present the self-similar process, $X(t)$, in terms of a discrete-time version, $X_{n,m}$,

defined at discrete points in time. In this process, $n \in \{1, 2, \dots\}$ is discrete time and m is the batch size. Thus:

$$X_{n,m} = \frac{1}{m} \sum_{i=(n-1)m+1}^{nm} X(i). \quad (18.6)$$

Note that for $X_{n,m}$, the corresponding aggregated sequence with a level of aggregation m , we divide the original series $X(t)$ into nonoverlapping blocks of size m and average them over each block where index n labels blocks.

Example. Consider traffic with batch size $m = 4$. Self-similar process averaging is expressed by

$$X_{n,4} = \frac{1}{4} (X(4n-3) + X(4n-2) + X(4n-1) + X(4n)). \quad (18.7)$$

Heavy-Tailed Distributions

Self-similarity implies that traffic has similar statistical properties in a timescale range, such as milliseconds, seconds, minutes, hours, or even days. In a practical situation, in which bursts of streams are multiplexed, the resulting traffic tend to produce a bursty aggregate stream. In other words, the traffic has a long-range dependence characterized by a *heavy-tailed distribution*. A random variable has heavy-tailed distribution if for $0 < \alpha < 2$, its cumulative distribution function (CDF)

$$F_X(x) = P[X \leq x] \sim 1 - \frac{1}{x^\alpha} \quad (18.8)$$

as $x \rightarrow 0$.

Heavy-tailed distributions are typically used to describe the distributions of burst lengths. A simple example of heavy-tailed distributions is the *Pareto distribution*, which is characterized by the following CDF and probability density function (PDF):

$$\begin{cases} F_X(x) = 1 - \left(\frac{k}{x}\right)^\alpha, \\ f_X(x) = \frac{\alpha k^\alpha}{x^{\alpha+1}} \end{cases}, \quad (18.9)$$

where k is the smallest possible value of the random variable. For this distribution, if $\alpha \leq 1$, the distribution has infinite mean and variance; if $\alpha \leq 2$, the distribution has infinite variance. To define *heavy tailed*, we can now use a comparison over PDFs of a Pareto distribution and exponential distribution. Making this comparison shows how

the tail of the curve in a Pareto distribution takes much longer to decay. A random variable that follows a heavy-tailed distribution may be very large with a probability that cannot be negligible.

18.7 Summary

This chapter explored transport mechanisms for application delivery with the highest possible quality. We focused on media applications, such as streaming audio and video, one-to-many transmission of real-time audio and video, and real-time interactive audio and video.

We discussed how sampled and digitized voice are treated in networks and investigated two VoIP signaling session protocols: *Session Initiation Protocol* (SIP) and the *H.323 series of protocols*, showing how calling and numbering for these protocols can be achieved. SIP identifies user location signals, call setup, call termination, and busy signals. User agents assist in initiating or terminating a phone call in VoIP networks. User agents can be implemented in a standard telephone or in a laptop with a microphone that runs some software. The signaling in H.323 uses both UDP and TCP and is partitioned into *call setup*, *initial communication capability*, *audio/video communication establishment*, *communications*, and *call termination*. Various scheduling-policy mechanisms—*priority queueing*, *weighted fair queuing*, and *class-based weighted fair queuing*—can provide the foundation of a QoS networking architecture.

Senders use various *real-time media transport protocols* to send a stream of data at a constant rate. One of the protocols for real-time transmission is the *Real-Time Transport Protocol* (RTP), which provides application-level framing. Real-time applications, such as voice and video, can tolerate a certain amount of packet loss and do not always require retransmission of data. But if the rate of packet loss is very high, the source might use a lower-quality transmission, thereby placing less load on the network.

A video in a single server can be streamed from a video server to a client for each client request. *Content distribution networks* (CDNs) can be used for streaming data. A video streaming provided to a user in an ISP domain can use Domain Name System (DNS) servers to direct browsers to the correct server. The *Stream Control Transmission Protocol* (SCTP) is a general-purpose transport protocol for transporting stream traffic. SCTP offers acknowledged and nonduplicated transmission of basic units of data, or datagrams, on connectionless networks.

Finally, a *non-Markovian analysis of streaming traffic* was presented. A stream of packets generated from a server source can be modeled as a discrete sequence of events and defined as a discrete-time 0-1 valued process called self-similar traffic.

The last two chapters of the book consider two advanced and related subjects: *mobile ad hoc networks* and *wireless sensor networks*.

18.8 Exercises

1. We want to transmit a speaker's voice through a digital radio broadcast system that uses 8-bit code PCM (explained in Chapter 17) placed on the Internet.
 - (a) How many bits are produced in each second?
 - (b) Estimate the size of an encapsulated RTP/IP datagram (the basic transmission unit of data in the Internet) that carries a half second of PCM-encoded audio using UDP.
2. Two voice sources come to one server for live transmission together, using RTP packets. Each source bandwidth has been compressed to 31 Kb/s.
 - (a) Show an overview of the encapsulated RTP/IP datagram, and estimate the size of each packet, utilizing an RTP packet using UDP.
 - (b) How many packets are produced each 5 minutes?
3. We have five packets to transmit in real time. The estimated average jitter until the time of the first packet is 0.02 ms. Table 18.1 shows the timestamps of RTP data packets indicated by the source, t_i , and the arrival times of RTP packets at the receiver a_i . Assume that the normalizing coefficient k is 0.2.
 - (a) Estimate the average jitter until every packet has arrived.
 - (b) What would be possible reason(s) that t_i increases at a different rate from i ?

Table 18.1 Exercise 3 example of source timestamps and receiver arrival times for five packets

i	a_i	t_i
1	43	69
1	45	74
1	47	73
1	49	91
1	51	99

4. SCTP is applied to transmit a color video clip between two points of an IP network requiring 4 minutes of network use. Each image of this video clip consists of $1,024 \times 1,280$ pixel blocks, and the video consists of 30 images per second. The video clip is not compressed but is passed through the quantization process, and each pixel can be a value among a sample space of 77 numbers. One-tenth of each row of a frame (image) is formatted by one packet.
 - (a) Find the size of each SCTP chunk, including its header.
 - (b) Find the total size of each SCTP packet, including its header.
 - (c) Find the total size of bits transmitted, based only on payload packets.
 - (d) Determine the required bandwidth between these two nodes.
5. Suppose that a live transmission of a compressed color video movie between two points of an IP network requires 2 hours of network use. We want to apply SCTP for this transmission. Each image of this video consists of $1,024 \times 1,280$ pixel blocks, and the video consists of 30 images per second. One option is to encapsulate each block of pixels in a chunk, allowing each row of a frame (image) to be formatted by one packet. Assume that each pixel block is compressed on average to 10 phrases and each phrase requires on average 5 bits.
 - (a) Find the size of each SCTP chunk, including its header.
 - (b) Find the total size of each SCTP packet, including its header.
 - (c) Find the total size of bits transmitted, based only on payload packets.
 - (d) Determine the required bandwidth between these two nodes.
6. Assume that a real-time bursty source is modeled using a Brownian motion process $X(t)$. Let $Z(t) = X(t) + 2t$.
 - (a) Find the probability distribution function (PDF) of $Z(t)$.
 - (b) Find the joint PDF of $Z(t)$ and $X(t + 1)$.
7. In Figure 18.15, a remote medical emergency unit is streaming the 70-cycle/minute heartbeat of a patient to a hospital, using SCTP. Each heart cycle has six peaks: P, Q, R, S, T, and U. Suppose that all information about each peak is formatted into one chunk of the SCTP data packet. Because of their importance and complexity, each of the Q, R, and S pulses requires four samples; P, T, or U require only one sample each.
 - (a) Determine the required bandwidth between the unit and the hospital.
 - (b) If multiple samples are included in a packet, find the maximum number of heartbeat cycles that can be formatted into a packet.

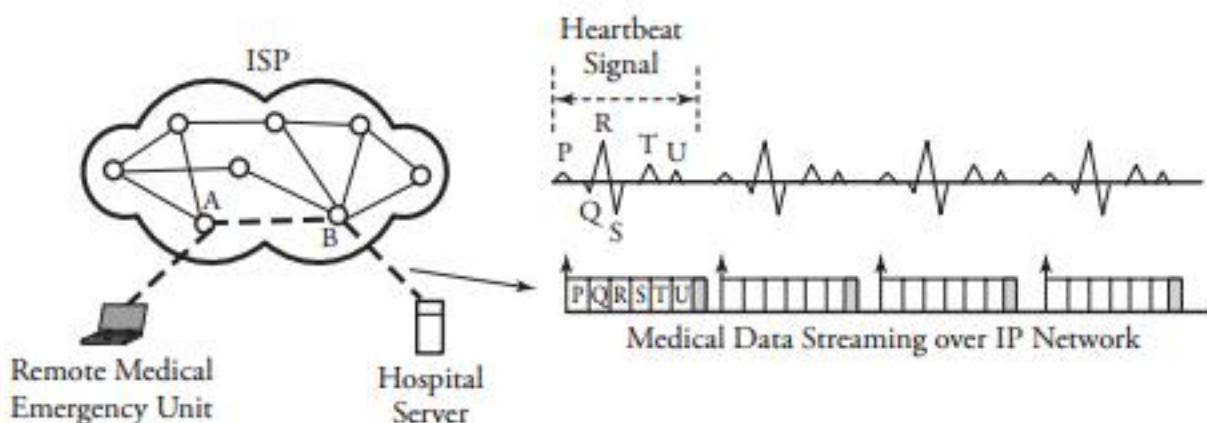


Figure 18.15 Exercise 7 example of remote medical emergency unit streaming a patient's heartbeat of to a hospital. The heartbeat signal is converted to stream of packets.

- (c) Evaluate the approach presented in part (b), and compare it with the original one.
- 8. For self-similar traffic, we have seen the relationship between the expected values in Equation (18.4), using the Hurst parameter, H .
 - (a) Derive a relationship between the variances of the two sides of Equation (18.3).
 - (b) Compare situations in which the Hurst parameter, H , takes values 0.5, 0.8, or 1 in Equation (18.3).
- 9. To better understand the behavior of bursty traffic, such as a video streaming source, assume that the traffic is described by a Pareto distribution with $k = 1$. Plot PDFs of the following two cases of α , and compare them with that of an exponential distribution. Comment on the heavy-tailed phenomenon.
 - (a) $\alpha = 0.8$
 - (b) $\alpha = 3.8$
- 10. *Computer simulation project.* We want to simulate self-similar traffic. First, develop a random number generator. Then, develop source code that defines the following components, and integrate the these two programs to simulate the self-similar traffic:
 - (a) Source ID
 - (b) Priority of the packets
 - (c) Elapsed time (in byte transmission times)
 - (d) Packet size (bytes)
 - (e) Number of packets remaining in current burst

This page intentionally left blank

CHAPTER 19

Mobile Ad-Hoc Networks

Mobile ad-hoc networks (MANETs) have had a profound impact in the world of computer networks. Characterized by anytime/anywhere untethered establishment of a wireless network, the MANET infrastructure enables location-independent services. Ad-hoc networks do not need any fixed infrastructure to operate and support dynamic topology scenarios in which no wired infrastructure exists. This chapter covers the following topics on wireless mobile ad-hoc networks:

- *Overview of wireless ad-hoc networks*
- *Routing in ad-hoc networks*
- *Ad-hoc routing protocols for ad-hoc networks*
- *Security of ad-hoc networks*

A mobile user can act as a routing node, and a packet can be routed from a source to its destination without having any static router in the network. Two classes of routing strategies in ad-hoc networks are *table-driven routing protocols* and *source-initiated routing protocols*. Security of ad-hoc networks is a key issue. Ad-hoc networks are, by their nature, vulnerable to attacks. An intruder can easily attack ad-hoc networks by loading available network resources and disturbing the normal operation of routing protocols by modifying packets.

19.1 Overview of Wireless Ad-Hoc Networks

Wireless *mobile ad-hoc network* (MANET) technology is designed for the establishment of a network anywhere and anytime, without any fixed infrastructure to support the mobility of the users in the network. In other words, a wireless ad-hoc network is a collection of mobile nodes with a dynamic network infrastructure forming a temporary network. Such networks no central server or base station for providing connectivity, and all network intelligence must be placed inside the mobile user devices. Figure 19.1 gives overview of an ad-hoc network, where wireless mobile nodes have formed a network, with one node too far to reach.

In such an environment, each mobile user acts as a routing node, and a packet is routed from a source to its destination by incorporating of other network users. Since the topology of an ad-hoc network can change quickly and unpredictably, it should be adaptable to changes, such as when a link breaks, a node leaves the network, or a new node is attached to the network. Thus, unlike intradomain routing algorithm for regular networks, if a node leaves an ad-hoc network, all affected nodes can discover new routes. Ad-hoc networks have several types of applications

- *Rescue operations.* In an emergency public disaster, such as an earthquake, ad-hoc networks can be set up at a location where no infrastructure is present. Ad-hoc networks can be used to support network connectivity when no fixed network is available.

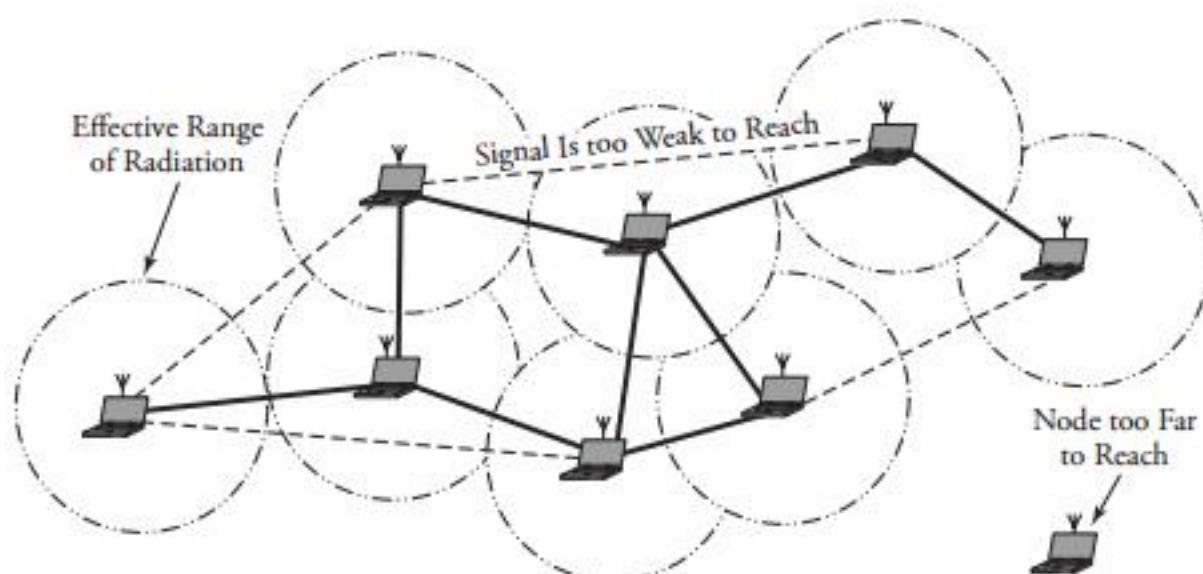


Figure 19.1 Overview of an ad-hoc network

- *Military.* Ad-hoc networks can be used in a battle zone, for a military command and mobile units.
- *Law enforcement and security operations.* An ad-hoc network can be used in a temporary security operation, acting as a mobile surveillance network.
- *Home networks.* An ad-hoc network can be used to support seamless connectivity among various devices.
- *Conferencing.* Ad-hoc networks can be set up for a presentation. An audience can download a presentation, browse the slides on a portable device, print them on the local printer, or e-mail the presentation to an absent colleague.

Ad-hoc networks must possess several unique features. One is *automatic discovery* of available services. Each time a new service becomes available, an ad hoc networking device has to configure use of the new service. As an ad-hoc network lacks centralized administration, the network must be able to prevent network collapse when one of the mobile nodes moves out of transmitter range. In general, nodes should be able to enter or leave the network as they wish. Thus, every node acts as both a host and a router, and the network must be intelligent enough to handle network dynamics. This property is called *self-stabilization*.

One of the most common tasks of an ad-hoc network is to multicast a message to many users efficiently. In such an environment, networks are subject to severe blocking. Thus, the performance of an ad hoc system depends on the stability of the network architecture. The inclusion of all these features in ad-hoc networks requires considerable architecture sophistication.

19.2 Routing in Ad-Hoc Networks

The lack of a backbone infrastructure makes packet routing in ad-hoc networks a challenging task. A routing protocol should be able to automatically recover from any problem in a finite amount of time without human intervention. Conventional routing protocols are designed for nonmoving infrastructures and assume that routes are bidirectional, which is not always the case for ad-hoc networks. Identification of mobile terminals and correct routing of packets to and from each terminal while moving are certainly challenging.

Since the topology of an ad-hoc network is dynamic, reserving resources and sustaining QoS are difficult. In an ad-hoc medium, it may not be possible to communicate bidirectionally, so ad-hoc routing protocols should assume that links are unidirectional. The power of a wireless device is another important factor. The routing protocol also

has to support node stand-by mode. Devices such as laptops are very limited in battery power; hence, the use of stand-by mode to save power is important.

19.2.1 Classification of Routing Protocols

Ad-hoc routing protocols can be classified into two broad categories:

1. *Centralized versus distributed*. In centralized routing protocols, the routing decision is made at a central node. In distributed routing protocols, the routing decision is made by all the network nodes. Routing protocols in most efficiently designed ad-hoc networks are distributed to increase the reliability of the network. In such cases, nodes can enter or leave the network easily, and each node can make routing decisions, using other collaborative nodes.
2. *Static versus adaptive*. In static routing protocols, a route of a source/destination pair does not change because of any traffic condition or link failure. In adaptive routing protocols, routes may change because of any congestion.

Whether a protocol is centralized, distributed, static, or adaptive, it can generally be categorized as either *table driven* or *source initiated*.

Table-Driven Routing Protocols

Table-driven, or proactive, routing protocols find routes to all possible destinations ahead of time. The routes are recorded in the nodes' routing tables and are updated within the predefined intervals. Proactive protocols are faster in decision making but need more time to converge to a steady state, causing problems if the topology of the network continually changes. However, maintaining routes can lead to a large overhead. Table-driven protocols require every node to maintain one or more tables to store updated routing information from every node to all other nodes. Nodes propagate updated tables all over the network such that the routing information in each table corresponds to topological changes in the network.

Source-Initiated Routing Protocols

Source-initiated, or reactive, routing protocols, are on-demand procedures and create routes only when requested to do so by source nodes. A route request initiates a *route-discovery process* in the network and is completed once a route is discovered. If it exists at the time of a request, a route is maintained by a route-maintenance procedure until either the destination node becomes irrelevant to the source or the route is no longer

needed. On-demand protocols are more suitable for ad-hoc networks. In most cases, reactive protocols are desired. This means that the network reacts only when needed and does not broadcast information periodically. However, the control overhead of packets is smaller than for proactive protocols.

19.3 Routing Protocols for Ad-Hoc Networks

This section discusses three table-driven protocols and four source-initiated protocols. The table-driven protocols are the *Destination-Sequenced Distance Vector* (DSDV) protocol, the *Cluster-Head Gateway Switch Routing* (CGSR) protocol, and the *Wireless Routing Protocol* (WRP). The source-initiated protocols are the *Dynamic Source Routing* (DSR) protocol, the *Associative-Based Routing* (ABR) protocol, *Temporally Ordered Routing Algorithm* (TORA), and *Ad-Hoc On-Demand Distance Vector* (AODV) protocol.

19.3.1 Destination-Sequenced Distance Vector (DSDV) Protocol

The *Destination-Sequenced Distance Vector* (DSDV) protocol is a table-driven routing protocol based on the improved version of classical Bellman-Ford routing algorithm. DSDV is based on the *Routing Information Protocol* (RIP), explained in Chapter 7. With RIP, a node holds a routing table containing all the possible destinations within the network and the number of hops to each destination. DSDV is also based on *distance vector routing* and thus uses bidirectional links. A limitation of DSDV is that it provides only one route for a source/destination pair.

Routing Tables

The structure of the routing table for this protocol is simple. Each table entry has a sequence number that is incremented every time a node sends an updated message. Routing tables are periodically updated when the topology of the network changes and are propagated throughout the network to keep consistent information throughout the network.

Each DSDV node maintains two routing tables: one for forwarding packets and one for advertising incremental routing packets. The routing information sent periodically by a node contains a new sequence number, the destination address, the number of hops to the destination node, and the sequence number of the destination. When the topology of a network changes, a detecting node sends an update packet to its

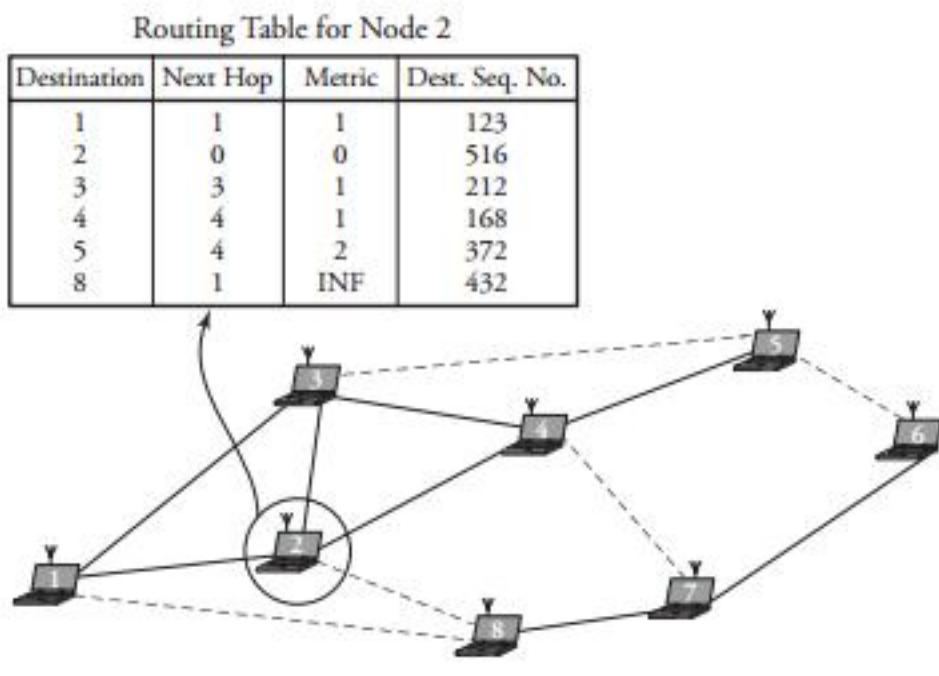


Figure 19.2 A DSDV routing table

neighboring nodes. On receipt of an update packet from a neighboring node, a node extracts the information from the packet and updates its routing table as follows

DSDV Packet Process Algorithm

1. If the new address has a higher sequence number, the node chooses the route with the higher sequence number and discards the old sequence number.
2. If the incoming sequence number is identical to the one belonging to the existing route, a route with the least cost is chosen.
3. All the metrics chosen from the new routing information are incremented.
4. This process continues until all the nodes are updated. If there are duplicate updated packets, the node considers keeping the one with the least-cost metric and discards the rest. ■

In case of a broken link, a cost of ∞ metric with a new sequence number (incremented) is assigned to it to ensure that the sequence number of that metric is always greater than or equal to the sequence number of that node. Figure 19.2 shows a routing table for node 2, whose neighbors are nodes 1, 3, 4, and 8. The dashed lines indicate no communications between any corresponding pair of nodes. Therefore, node 2 has no information about node 8.

The packet overhead of the DSDV protocol increases the total number of nodes in the ad-hoc network. This fact makes DSDV suitable for small networks. In large ad-hoc networks, the mobility rate—and therefore the overhead—increase, making the network unstable to the point that updated packets might not reach nodes on time.

19.3.2 Cluster-Head Gateway Switch Routing Protocol

The *Cluster-Head Gateway Switch Routing* (CGSR) protocol is a table-driven routing protocol. In a clustering system, each predefined number of nodes are formed into a *cluster* controlled by a *cluster head*, which is assigned using a distributed clustering algorithm. However, with the clustering scheme, a cluster head can be replaced frequently by another node, for several reasons, such as lower-level energy left in the node or a node moves out of contact.

With this protocol, each node maintains two tables: a *cluster-member table* and a *routing table*. The cluster-member table records the cluster head for each destination node, and the routing table contains the next hop to reach the destination. As with the DSDV protocol, each node updates its cluster-member table on receiving a new update from its neighbors.

Clustering and Routing Algorithms

CGSR routing involves cluster routing, whereby a node is required to find the best route over cluster heads from the cluster-member table. Figure 19.3 shows an example of routing in an area in which six clusters have been formed. A node in cluster A is transmitting a packet to a node in cluster F. Nodes within each cluster route their packets to their own associated clusters. The transmitting node then sends its packet to the next hop, according to the routing table entry associated with that cluster head. The cluster head transmits the packet to another cluster head until the cluster head of the destination node is reached. The routing is made through a series of available cluster heads from A to F. Packets are then transmitted to the destination.

19.3.3 Wireless Routing Protocol (WRP)

The *Wireless Routing Protocol* (WRP) is a table-based protocol maintaining routing information among all nodes in the network. This protocol is based on the distributed Bellman-Ford algorithm. The main advantage of WRP is that it reduces the number

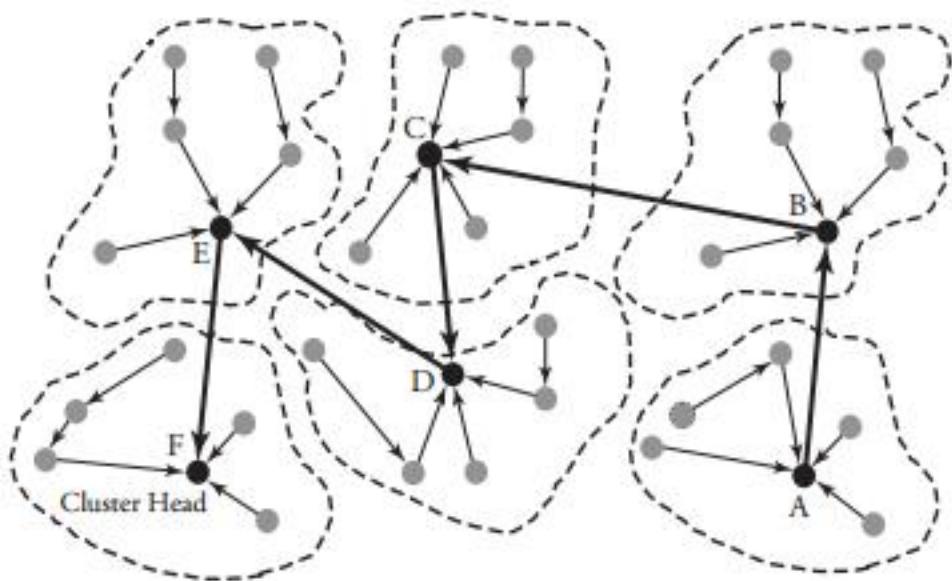


Figure 19.3 Communication with cluster-head gateway switch routing (CGSR) protocol

of routing loops. With this protocol, each node in a network maintains four tables, as follows:

1. *Distance table*, which holds the destination, next hop, distance, and predecessors of each destination and each neighbor.
2. *Routing table*, which saves the destination address, next hop, distance, predecessor, and a marker for each destination, specifying whether that entry corresponds to a simple path.
3. *Link-cost table*, which provides the link cost to each neighbor and also the number of periodic update periods elapsed since the node received any error-free message from it.
4. *Message transmission-list table*, which records which updates in an update message are to be retransmitted and which neighbors need to acknowledge the retransmission. The table provides the sequence number of the update message, a retransmission counter, acknowledgments, and a list of updates sent in the update message.

Nodes should either send a message including the update message or a HELLO message to their neighbors. If a node has no message to send, it should send a HELLO message to ensure connectivity. If the sending node is new, it is added to the node's routing table, and the current node sends the new node a copy of its routing table content.

Once it detects a change in a route, a node sends the update message to its neighbors. The neighboring nodes then change their distance entries and look for new possible paths through other nodes. This protocol avoids the count-to-infinity issue present in most ad-hoc network protocols. This issue is resolved by making each node perform consistency checks of predecessor information reported by all its neighbors in order to remove looping and make a faster route convergence in the presence of any link or node failure.

19.3.4 Dynamic Source Routing (DSR) Protocol

The *Dynamic Source Routing* (DSR) protocol is an on-demand, or source-initiated, routing protocol in which a source node finds an unexpired route to a destination to send the packet. DSR quickly adapts to topological changes and is typically used for networks in which mobile nodes move with moderate speed. Overhead is significantly reduced with this protocol, since nodes do not exchange routing table information when there are no changes in the topology. DSR creates multiple paths from a source to a destination, eliminating route discovery when the topology changes. Similar to most ad-hoc networks, DSR has two phases: route discovery and route maintenance.

Route Discovery and Maintenance

Route discovery is initiated when a node wants to send packets to another node and no unexpired route to the destination is in its routing table. In such circumstances, the node first broadcasts a *route-request packet*, including the destination address, source address, and a unique identification number. When it receives a route-request packet, the neighboring node it looks at its table; if any route to the requested destination address is already present in the node's route record, the packet is discarded to avoid the looping issue. Otherwise, the node adds its own address to the preassigned field of the route-request packet and forwards it to its neighboring nodes.

When the route-request packet reaches a destination node or another intermediate node that has an unexpired route to the destination, this node generates a *route-reply packet*, which contains a route record of the sequence of nodes taken from the source to this node. Once the source receives all the route-reply packets, it updates its routing table with the best path to the destination and sends its packets through that selected route.

Example. Figure 19.4 shows an ad-hoc network with eight mobile nodes and a broken link (3-7). Node 1 wants to send a message to the destination, node 8. Node 1

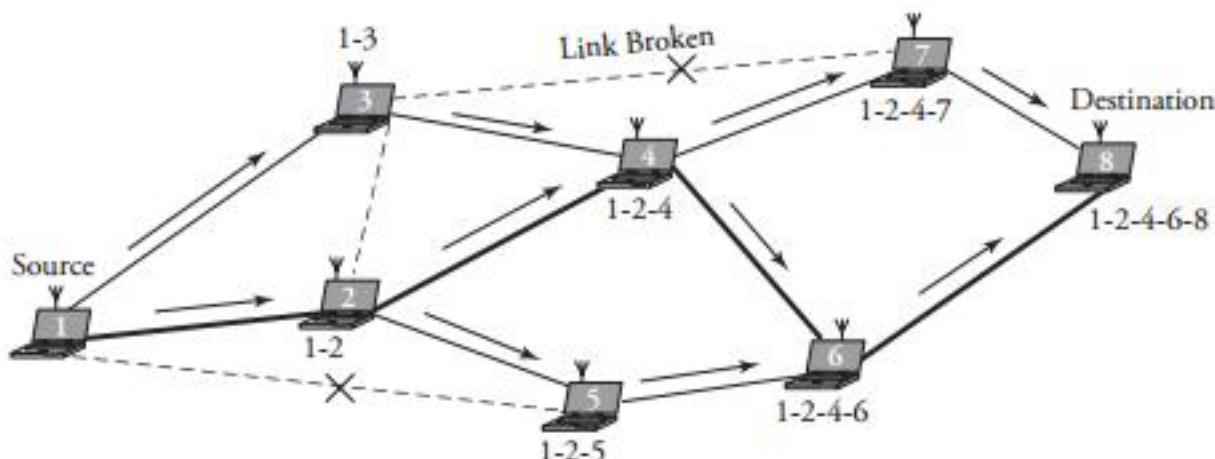


Figure 19.4 Summary of DSR connection setup from node 1 to node 8

looks at its routing table, finds an expired route to node 8, and then propagates route-request packets to nodes 3 and 2. Node 3 finds no route to the destination and so appends the route record 1-3 to the route-request packet and forwards it to node 4. On receiving this packet, node 7 finds a route to the destination and so stops propagating any route-request packet and instead sends a route-reply packet to the source. The same happens when a route-request packet reaches the destination node 8 with a route record 1-2-4-6. When the source, node 1, compares all the route-reply packets, it concludes that the best route is 1-2-4-6-8 and establishes this path.

Route maintenance in this protocol is fast and simple. In case of a fatal error in the data-link layer, a *route-error packet* is generated from a failing node. When the route-error packet is received, the failing node is removed from its route cache, and all routes containing that node are truncated. Another route-maintenance signal is the acknowledgment packets sent to verify the correct operation of the route links.

19.3.5 Temporally Ordered Routing Algorithm (TORA)

The *Temporally Ordered Routing Algorithm* (TORA) is a source-initiated routing algorithm and, thus, creates multiple routes for any source/destination pair. The advantage of multiple routes to a destination is that route discovery is not required for every alteration in the network topology. This feature conserves bandwidth usage and increases adaptability to topological changes by reducing communication overhead.

TORA is based on the following three rules:

1. Route creation/discovery
2. Route maintenance
3. Route erasure

TORA uses three types of packets: *query packets* for route creation, *update packets* for both route creation and maintenance, and *clear packets* for route erasure. With TORA, nodes have to maintain routing information about adjacent nodes. This loop-free protocol is distributed based on the concept of *link reversal*.

Every node maintains a table describing the distance and status of all connected links. When a node has no route to a desired destination, a *route-creation process* starts. A query packet contains the destination ID, and an update packet contains the destination ID and the distance of the node. A receiving node processes a query packet as follows

- If the receiving node realizes that there are no further downstream links, the query packet is again broadcast. Otherwise, the node discards the query packet.
- If the receiving node realizes that there is at least one downstream link, the node updates its routing table to the new value and broadcasts an update packet.

Once it receives the update packet, a node sets its distance greater than the neighbor from which it received the packet and then rebroadcasts it. The update is eventually received by the source. When it realizes that there is no valid route to the destination, the node adjusts its distance and generates an update packet. TORA performs efficient routing in large networks and in mildly congested networks.

19.3.6 Associative-Based Routing (ABR) Protocol

Associative-Based Routing (ABR) is an efficient on-demand, or source-initiated, routing protocol. ABR's is better than WRPs' network-change adaptation feature, to the extent that it is almost free of loops and is free of packet duplications. In ABR, the destination node decides the best route, using node *associativity*. ABR is ideal for small networks, as it provides fast route discovery and creates shortest paths through associativity.

In ABR, the movements of nodes are observed by other nodes in the network. Each node keeps track of associativity information by sending messages periodically, identifying itself and updating the *associativity ticks* for its neighbors. If the associativity ticks exceed a maximum, a node has associativity stability with its neighbors. In other words, a low number of associativity ticks shows the node's high mobility, and high associativity indicates a node's sleep mode. The associativity ticks can be reset when a node or its neighbor moves to a new location.

Each point-to-point routing in ABR is connection oriented, with all nodes participating in setting up and computing a route. In a point-to-point route, the source node or any intermediate node decides the details of routes. If the communication must be of broadcast type, the source broadcasts the packet in a *connectionless routing* fashion.

Route discovery is implemented when a node wants to communicate with a destination with no valid route. Route discovery starts with sending a *query packet* and an *await-reply*. The broadcast query packet has the source ID, destination ID, all intermediate nodes' IDs, sequence number, *CRC*, *LIVE* field and a *TYPE* field to identify the type of message. When it receives a query packet, an intermediate node looks at its routing table to see whether it is the intended destination node; otherwise, it appends its ID to the list of all intermediate IDs and rebroadcasts the packet. When it receives all copies of the query packet, the destination node chooses the best route to the source and then sends a *reply packet* including the best route. This way, all nodes become aware of the best route and thereby make other routes to the destination invalid.

ABR is also able to perform *route reconstruction*. This function is needed for partial route discovery or invalid route erasure.

19.3.7 Ad-Hoc On-Demand Distance Vector (AODV) Protocol

The *Ad-Hoc On-Demand Distance Vector* (AODV) routing protocol is an improvement over DSDV and is a source-initiated routing scheme capable of both unicast and multicast routing. AODV establishes a required route only when it is needed as opposed to maintaining a complete list of routes, with DSDV. AODV uses an improved version of the *distance vector* algorithm, explained in Chapter 7, to provide on-demand routing. AODV offers quick convergence when a network topology changes because of any node movement or link breakage. In such cases, AODV notifies all nodes so that they can invalidate the routes using the lost node or link. This protocol adapts quickly to dynamic link conditions and offers low processing, memory overhead, and network utilization. Loop-free AODV is self-starting and handles large numbers of mobile nodes. It allows mobile nodes to respond to link breakages and changes in network topology in a timely manner. The algorithm's primary features are as follows.

- It broadcasts packets only when required.
- It distinguishes between local connectivity management and general maintenance.
- It disseminates information about changes in local connectivity to neighboring mobile nodes that need this information.
- Nodes that are not part of active paths neither maintain any routing information nor participate in any periodic routing table exchanges.

- A node does not have to find and maintain a route to another node until the two nodes communicate. Routes are maintained on all nodes on an active path. For example, all transmitting nodes maintain the route to the destination.

AODV can also form multicast trees that connect multicast group members. The trees are composed of group members and the nodes needed to connect them. This is similar to multicast protocols, explained in Chapter 15.

Routing Process

A route is *active* as long as data packets periodically travel from a source to a destination along that path. In other words, an active route from a source to a destination has a valid entry in a routing table. Packets can be forwarded only on active routes. Each mobile node maintains a *routing table entry* for a potential destination. A routing table entry contains

- Active neighbors for a requested route
- Next-hop address
- Destination address
- Number of hops to destination
- Sequence number for the destination
- Expiration time for the route table entry (timeout)

Each routing table entry maintains the addresses of the active neighbors so that all active source nodes can be notified when a link along the route to the destination breaks. For each valid route, the node also stores a list of precursors that may transmit packets on this route. These precursors receive notifications from the node when it detects the loss of the *next-hop* link.

Any route in the routing table is tagged with the destination's *sequence number*, an increasing number set by a counter and managed by each originating node to ensure the freshness of the reverse route to a source. The sequence number is incremented whenever the source issues a new route-request message. Each node also records information about its neighbors with bidirectional connectivity. The insertion of a sequence number guarantees that no routing loops form even when packets are delivered out of order and under high node mobility. If a new route becomes available to the requested destination, the node compares the destination sequence number of the new incoming route with the one stored in the routing table for the current route. The existing entry

is updated by replacing the current entry with the incoming one if one of the following conditions exist.

- The current sequence number in the routing table is marked invalid.
- The new incoming sequence number is greater than the one stored in the table.
- The sequence numbers are the same, and the new hop count is smaller than the one for the current entry.

Once the source stops sending data packets on an established connection, the routes time out and are eventually deleted from the intermediate-node routing tables. At the reverse-path entry of any routing table, a *request-expiration timer* purges the reverse-path routing entries from the nodes that do not lie on the route from the source to the destination. When an entry is used to transmit a packet, the timeout for the entry is reset to the current time plus the active-route timeout. The routing table also stores the *routing caching time out*, which is the time after which the route is considered to be invalid.

Route Discovery and Establishment

Suppose that a source node does not have information about a destination node, perhaps because it is unknown to the source node or a previously valid path to the destination expires or is marked invalid. In such cases, the source node initiates a *route-discovery* process to locate the destination. Route discovery is done by broadcasting a *route request* (RREQ) packet to neighbors, which in turn is forwarded to their neighbors until the destination node is reached. If it receives an already processed RREQ, a node discards the RREQ and does not forward it. Neighboring nodes update their information and set up backward pointers to the source node in their route tables. Each neighbor either responds to the RREQ by sending back a route-reply RREP to the source or increases the hop count and broadcasts the RREQ to its own neighbors; in this case, the process continues.

An RREQ packet contains the following information:

- Source address
- A unique RREQ
- Destination address
- Source sequence number
- Destination sequence number
- Hop count
- Lifetime

When an RREQ packet arrives at an intermediate node on a route, the node first updates the route to the previous hop. The node then checks whether the available route is current and completes this check by comparing the destination sequence number in its own route entry to the destination sequence number in the RREQ packet. If the destination sequence number is greater than the one available in the intermediate node's routing table, the intermediate node must not use its recorded route to respond to the RREQ. In this case, the intermediate node rebroadcasts the RREQ to its neighboring node.

On receipt of an RREQ, an intermediate node maintains the address of each neighbor from which the first copy of the packet is received in their route tables and establishes a reverse path. Therefore, if additional copies of the same RREQ are received, they are discarded. When the RREQ reaches the destination node, it responds by sending a *route reply* (RREP) packet back to the neighbor from which the RREQ was first received. As the RREP is routed back, nodes along this reverse path setup forward route entries in their routing tables, which indicates the node from which the RREP came.

Any intermediate node checks whether it has received an RREQ with the same source node IP address and RREQ within at least the last path-discovery time. If such an RREQ has been received, the node drops the newly received RREQ. The reverse-route entries are maintained long enough for the RREQ packet to pass through the network and produce a reply to the sender. For the RREQ that is not dropped, the node increments the hop count and then searches for a reverse route to the source. At this point, a routing timer associated with each route times out and deletes the entry if it has not received any RREP or is not used within a specified time. The protocol uses destination sequence-numbers to ensure that links are free of loops at all times and thus avoids counting to infinity. The destination address field is incremented every time a source issues a new RREQ.

Example. Figure 19.5 shows the process of signals with AODV from node 1 to node 8. To establish a connection, source node 1 searches in its table for a valid route to destination node 8. In the figure, an RREQ reaches the destination for the first time through path 1-2-4-6-8. The destination then issues an RREP packet to the source. After a while, the destination receives another RREQ, this time through path 1-3-7-8. The destination evaluates this path, and finds that path 1-3-7-8 is better, and then issues a new RREP packet, telling the source to discard the other reply.

As an RREP packet is propagated back to the source, involving nodes set up forward pointers to the destination. At this time, the hop-count field is incremented at each node. Thus, when the RREP packet reaches the source, the hop count represents the

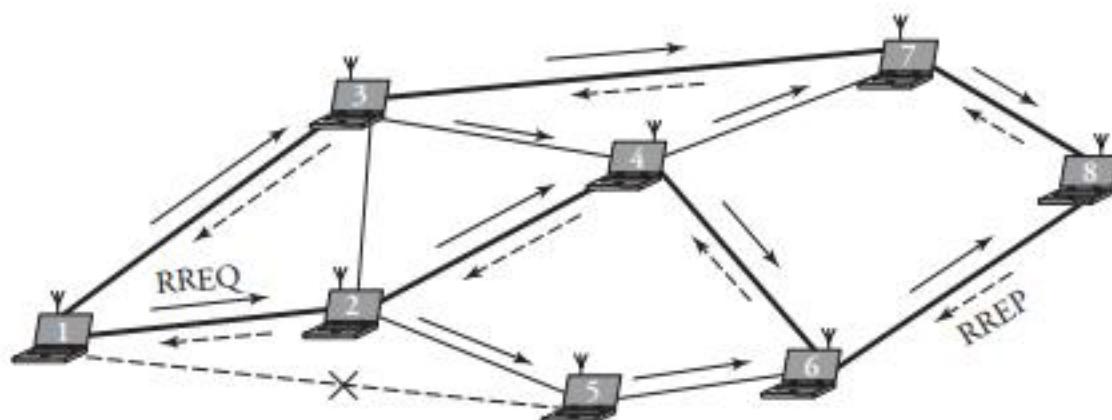


Figure 19.5 AODV communication signaling from node 1 to node 8

distance of the destination from the originator. The source node starts sending the data packets once the first RREP is received. If it receives an RREP representing a better route to the destination, the source node then updates its routing information. This means that, while transmitting, if it receives a better RREP packet containing a greater sequence number or the same sequence number with a smaller hop count, the source may update its routing information for that destination and switch over to the better path. As a result, a node ignores all less-desired RREPs received while transmitting.

At the reverse-path entry of any routing table, a *request-expiration timer* purges the reverse-path routing entries from the nodes that do not lie on the route from the source to the destination. When an entry is used to transmit a packet, the timeout for the entry is reset to the current time plus the active-route timeout. The routing table also stores the *routing caching timeout*, which is the time after which the route is considered invalid. In Figure 19.6, a timeout has occurred. From the source node 1 to the destination node 8, two routes 1-2-4-6-8 and 1-3-7-8, are found. However, the RREP at the intermediate node 7 exceeds the time allowed to be released. In this case, route 1-3-7-8 is purged from the involving routing tables.

Route Maintenance

After it knows how to establish a path, a network must maintain it. In general, each forwarding node should keep track of its continued connectivity to its active next hops. If a source node moves, it can reinitiate *route discovery* to find a fresh route to the destination. When a node along the route moves, its upstream neighbor identifies the move and propagates a link-failure notification message to each of its upstream neighbors. These nodes forward the link-failure notification to their neighbors, and so

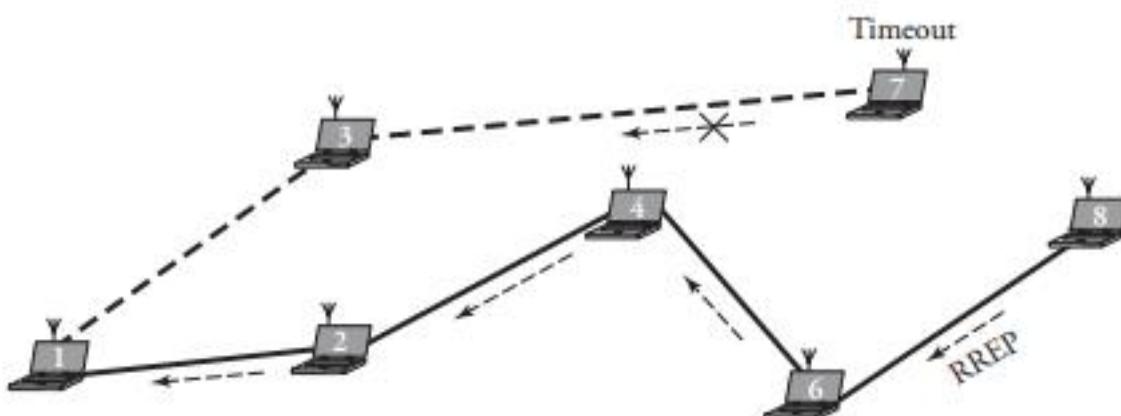


Figure 19.6 Occurrence of a timeout

on, until the source node is reached. The source node may reinitiate *path-discovery* if a route is still desired.

When the local connectivity of a mobile node is required, each mobile node can get information about other nodes in its neighborhood by using local broadcasts known as HELLO messages. A node should use HELLO messages only if it is part of an active route. A node that does not receive a HELLO message from its neighbors along an active route sends a link-failure notification message to its upstream node on that route.

When it moves during an active session, a source node can start the route-discovery process again to find a new route to the destination node. If the destination or the intermediate node moves, a special RREP is sent to the affected source nodes. Periodic HELLO messages can normally be used to detect link failures. If a link failure occurs while the route is active, the upstream node of the breakage propagates a *route-error* (RERR) message. An RERR message can be either broadcast or unicast. For each node, if a link to the next hop is undetectable, the node should assume that the link is lost and take the following steps.

1. Make all related existing routes invalid.
2. List all destination nodes that would potentially be affected.
3. Identify all neighboring nodes that may be affected.
4. Send an RERR message to all such neighbors.

As shown in Figure 19.1, some next-hop nodes in a network might be unreachable. In such cases, the upstream node of the unreachable node propagates an unsolicited RREP with a fresh sequence number, and the hop count is set to infinity to all active upstream neighbors. Other nodes listen and pass on this message to their active

neighbors until all nodes are notified. AODV finally terminates the unreachable node (broken associated links).

Joining a New Node to Network

A new node can join an ad-hoc network by transmitting a HELLO message containing its identity and sequence number. When a node receives a HELLO message from a neighbor, the node makes sure that it has an active route to it or, if necessary, creates one. After this update, the current node can begin using this route to forward data packets. In general, nodes in a network may learn about their neighbors when a node receives a normal broadcast message or HELLO message. If the HELLO message is not received from the next hop along an active path, the active neighbors using that next hop are sent notification of a link break.

A node receiving HELLO messages should be part of an active route in order to use them. In every predetermined interval, active nodes in a network check whether they received HELLO messages from their neighbors. If it does not receive any packets within the hello interval, a node broadcasts a HELLO message to all its neighbors. Neighbors that receive this packet update their local connectivity information. Otherwise, if it does not receive any HELLO messages for more than some predetermined time, a node should assume that the link to this neighbor failed.

19.4 Security of Ad-Hoc Networks

Because of dynamic topological changes, ad-hoc networks are vulnerable at the physical link, as they can easily be manipulated. An intruder can easily attack ad-hoc networks by loading available network resources, such as wireless links and energy (battery) levels of other users, and then disturb all users. Attackers can also disturb the normal operation of routing protocols by modifying packets. The intruder may insert spurious information into routing packets, causing erroneous routing table updates and thus misrouting. Some other security vulnerabilities of ad-hoc networks follow.

- *Limited computational capabilities.* Typically, nodes in ad-hoc networks are modular, independent, and limited in computational capability and therefore may become a source of vulnerability when they handle public-key cryptography during normal operation.
- *Limited power supply.* Since nodes normally use battery as power supply, an intruder can exhaust batteries by creating additional transmissions or excessive computations to be carried out by nodes.

- *Challenging key management.* Dynamic topology and movement of nodes in an ad-hoc network make *key management* difficult if cryptography is used in the routing protocol.

In any network, routing information can give an attacker access to relationships among nodes and their IP addresses. Especially in ad-hoc networks, an attacker may be able to bring the network down.

19.4.1 Types of Attacks

Attacks in ad-hoc networks are either *passive* or *active*. In a passive attack, the normal operation of a routing protocol is not interrupted. Instead, an intruder tries to gather information by listening. Active attacks can sometimes be detectable and thus are less important. In an active attack, an attacker can insert some arbitrary packets of information into the network to disable it or to attract packets destined to other nodes.

Pin Attack

With the *pin*, or *black-hole*, *attack*, a malicious node pretends to have the shortest path to the destination of a packet. Normally, the intruder listens to a path set-up phase and, when learns of a request for a route, sends a reply advertising a shortest route. Then, the intruder can be an official part of the network if the requesting node receives its malicious reply before the reply from a good node, and a forged route is set up. Once it becomes part of the network, the intruder can do anything within the network, such as undertaking a denial-of-service attack.

Location-Disclosure Attack

By learning the locations of intermediate nodes, an intruder can find out the location of a target node. The *location-disclosure attack* is made by an intruder to obtain information about the physical location of nodes in the network or the topology of the network.

Routing Table Overflow

Sometimes, an intruder can create routes whose destinations do not exist. This type of attack, known as the *routing table overflow*, overwhelms the usual flow of traffic, as it creates too many dummy active routes. This attack has a profound impact on *proactive* routing protocols, which discover routing information before it is needed, but minimal impact on *reactive routing protocols*, which create a route only when needed.

Energy-Exhaustion Attack

Battery-powered nodes can conserve their power by transmitting only when needed. But an intruder may try to forward unwanted packets or request repeatedly fake or unwanted destinations to use up the energy of nodes' batteries.

19.4.2 Criteria for a Secure Routing Protocol

In order to prevent ad-hoc networks from attacks and vulnerability, a routing protocol must possess the following properties:

- *Authenticity.* When a routing table is updated, it must check whether the updates were provided by authenticated nodes and users. The most challenging issue in ad-hoc networks is the lack of a centralized authority to issue and validate certificates of authenticity.
- *Integrity of information.* When a routing table is updated, the information carried to the routing updates must be checked for eligibility. A misleading update may alter the flow of packets in the network.
- *In-order updates.* Ad-hoc routing protocols must contain unique sequence numbers to maintain updates in order. Out-of-order updates may result in the propagation of wrong information.
- *Maximum update time.* Updates in routing tables must be done in the shortest possible time to ensure the credibility of the update information. A timestamp or time-out mechanism can normally be a solution.
- *Authorization.* An unforgeable credential along with the certificate authority issued to a node can determine all the privileges that the node can have.
- *Routing encryption.* Encrypting packets can prevent unauthorized nodes from reading them, and only those routers having the decryption key can access messages.
- *Route discovery.* It should always be possible to find any existing route between two points in a network.
- *Protocol immunization.* A routing protocol should be immune to intruding nodes and be able identify them.
- *Node-privacy location.* The routing protocol must protect the network from spreading the location or other unpublic information of individual nodes.
- *Self-stabilization.* If the *self-stabilization* property of ad-hoc network performs efficiently, it must stabilize the network in the presence of damages continually received from malicious nodes.

- *Low computational load.* Typically, an ad hoc node is limited in powers as it uses a battery. As a result, a node should be given the minimal computational load to maintain enough power to avoid any denial-of-service attacks from low available power.

19.5 Summary

A wireless ad-hoc network supports “independent” wireless and mobile communication systems. A mobile user in fact acts as a routing node. Routing protocols can be *centralized versus distributed*, *static versus adaptive*, and *table driven versus source initiated routing*.

Table-driven routing protocols find routes to all possible destinations before they are needed. The routes are recorded in nodes’ routing tables and are updated within predefined intervals. Examples of such protocols are DSDV, CGSR, and WRP. CGSR had a better converging capability than others do. Source-initiated routing protocols, such as DSR, ABR, TORA, and AODV, create routes only when they are requested by source nodes. AODV is very popular owing to its ability to stabilize the routing and its better security. Security is a critical issue in ad-hoc networks. Security vulnerability to various types of attacks can be minimized by meeting specific security criteria.

The next chapter explores a special version of ad-hoc network: *sensor network*.

19.6 Exercises

1. Compare table-driven and source-initiated routing algorithms in terms of
 - (a) Speed of routing table updates
 - (b) Point-to-point routing when there is a sudden change in the network topology
2. Suppose that we have the ad-hoc network shown in Figure 19.7. The number on each link indicates the strength of the signal on that link. At time t_1 , the threshold value of 2.5 is applied on all links as the minimum good connection. Use the DSDV protocol to
 - (a) Show the content of the routing table for each node for $t < t_1$
 - (b) Perform the routing algorithm to connect A to F
 - (c) Show the content of the routing table for each node in $t \geq t_1$
3. With the same conditions stated in exercise 2, use the AODV protocol to
 - (a) Show the content of the routing table for each node for $t < t_1$

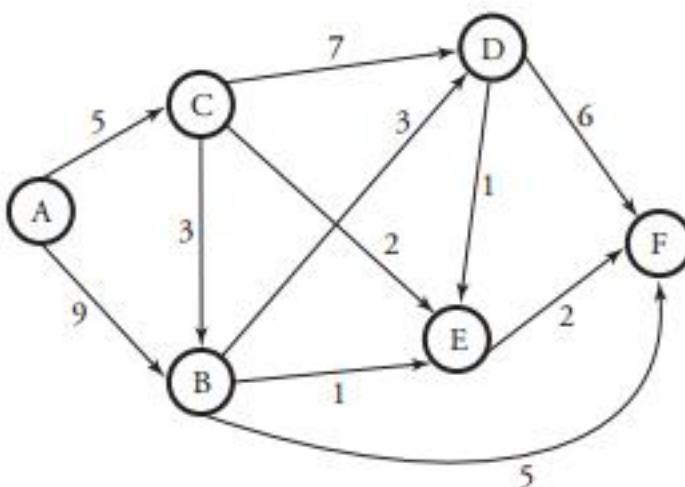


Figure 19.7 Exercises 2 and 3 ad-hoc network

- (b) Show the details of routing to connect A to F, including the communications with neighboring nodes
- (c) Show all the detailed steps of updating the neighboring nodes for each node in $t \geq t_1$
4. Consider an ad-hoc network consisting of users spread randomly in locations with Poisson distribution, where there are n users per square meter. Each user transmits with 40 percent probability and receives data with 60 percent probability in an assigned time slot. Make an assumption that the path loss exponent is β .
 - (a) Find the distribution of interference power received in any randomly selected user.
 - (b) Show that if $\beta > 2$, the average interference power is finite.
5. Consider Figure 19.3. We want to make a more detailed assignment to each node for routing from node A to F, using CGSR. Assume that the energy of cluster heads is normalized to A = 23, B = 18, C = 15, D = 16, E = 25, and F = 14. Find the best path from A to F. Assume that cluster heads have bidirectional links.
6. *Computer simulation project.* Consider the ad-hoc network shown in Figure 19.5. We want to implement AODV on this network. We first apply distance-vector routing to discover routes on an interconnected network. The primary distance vector routing uses the Bellman-Ford algorithm. Consider the network to be subject to topological changes at any time. You assign these times. Changes include a link to fail unexpectedly or a new link to be created or added. Let your program discover these changes, automatically update nodes' routing tables, and propagate these changes to the other nodes.

7. *Computer simulation project.* Now consider the network you analyzed in exercise 6. We want to test the AODV route-discovery procedure and the ability of AODV to handle link and node failures. At the network level, design a least-cost algorithm that gives the shortest path, taking failing nodes (malicious nodes) into account. If one of the nodes in the shortest path is a bad one, the next available shortest path must be found in the algorithm. In the algorithm you develop, the failure of a link should not affect the shortest-path determination. If a malicious node is in the network, that node is being discarded, and fresh routes are determined that have shortest path to the destination requested by the source node in the network. Assign faults deterministically, starting from the best path, and determine the shortest paths available. Determine how the best paths vary when the number of failing nodes is assigned randomly. Study how the network topology changes for a given n nodes with k failing nodes in the network. You will see that the total costs of the shortest paths between the sources and the destinations gradually increase with the number of fault nodes in the network if the fault nodes are assigned starting from the best path.

This page intentionally left blank

CHAPTER 20

Wireless Sensor Networks

Self-organizing sensor networks hold the potential to revolutionize many segments of public security, environmental monitoring, and manufacturing. Sensors can be networked to form a network to enhance sensing capability. Like a computer network, a sensor network has a packet with a header flowing in a network of nodes (sensor nodes). This chapter presents the architectures and protocols of such networks, discussing the following topics:

- *Sensor networks and protocol structures*
- *Communication energy model*
- *Clustering protocols*
- *Routing protocols*
- *Case study: Simulation of a sensor network*
- *Other related technologies*

We begin with an overview of sensor networks and explain some popular applications. We also provide an overview of a protocol stack for sensor networks and explain how the power factor distinguishes the routing protocols of sensor networks from those of computer networks. The protocol stack combines power efficiency and least-cost-path routing. Networking protocols and power efficiency are integrated through the wireless medium, and cooperative efforts of sensor nodes are promoted.

Clustering protocols in sensor networks specify the topology of the hierarchical network partitioned into nonoverlapping *clusters* of sensor nodes. Typically, a robust clustering technique is essential for self-organizing sensor networks. Two clustering protocols are the *Low-Energy Adaptive Clustering Hierarchy* (LEACH) algorithm and the *Decentralized Energy-Efficient Cluster Propagation* (DEEP) protocol. After well-distributed cluster heads and clusters have been established in a network, energy-conscious routing is essential in order to set communication routes among cluster heads.

This last chapter also offers a detailed numerical case-study on the implementation of a clustering protocol, as well as a discussion of *ZigBee technology*, which is based on the IEEE 802.15.4 standard. This technology uses low-power nodes and is a well-known low-power standard.

20.1 Sensor Networks and Protocol Structures

Chemical, biological, or solar sensors can be networked together as a *sensor network* to strengthen the power of sensing. A sensor network is controlled through a software core engine. The network is typically wireless but may also be wired. Sensor networks are designed to be self-configuring such that they can gather information about a large geographical area or about movements of an object for surveillance purposes.

Sensor networks can be used for target tracking, environmental monitoring, system control, and chemical or biological detection. In military applications, sensor networks can enable soldiers to see around corners and to detect chemical and biological weapons long before they get close enough to cause harm. Civilian uses include environmental monitoring, traffic control, and providing health care monitoring for the elderly while allowing them more freedom to move about.

20.1.1 Clustering in Sensor Networks

The region being sensed is normally partitioned into equally loaded *clusters* of sensor nodes, as shown in Figure 20.1. A cluster in a sensor network resembles a domain in a computer network. In other words, nodes are inserted in the vicinity of a certain predefined region, forming a cluster. Different types of sensors can also be deployed in a region. Thus, a sensor network is typically cluster based and has irregular topology. The most effective routing scheme in sensor networks is normally based on the energy (battery level) of nodes. In such routing schemes, the best path has the highest amount of total energy. The network of such sensing nodes is constructed with identical sensor nodes, regardless of the size of the network. In Figure 20.1, three clusters are

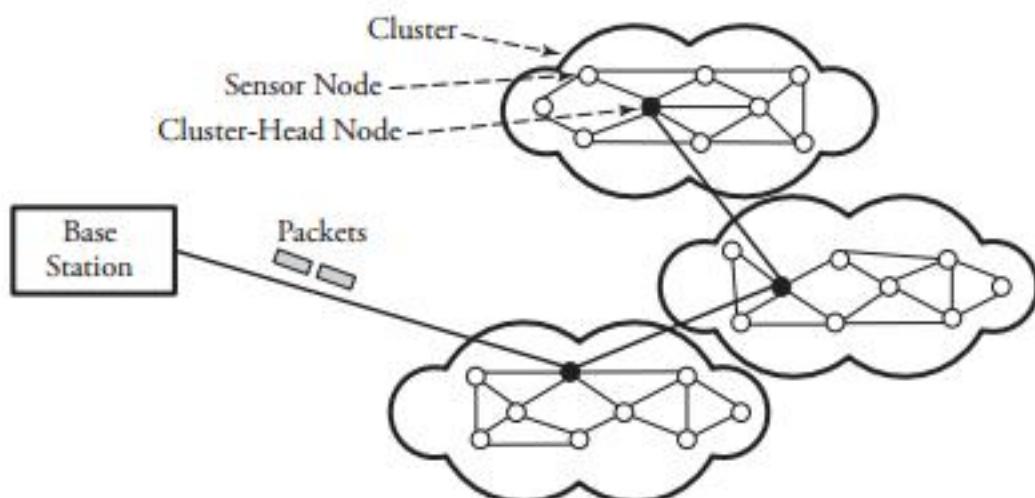


Figure 20.1 A sensor network and its clusters

interconnected to the main base station, each cluster contains a *cluster head* responsible for routing data from its corresponding cluster to a *base station*.

Communicating nodes are normally linked by a wireless medium, such as radio. The wireless sensor node is equipped with a limited power source, such as a battery or even a *solar cell*, where there is enough sunlight exposure on the node. However, a solar cell may not be the best choice as a power supply, owing to its weight, volume, and expense. In some application scenarios, sensor-node lifetime depends on the battery lifetime. Removal of dead nodes can cause significant topological changes and may require packet rerouting. As a result, power management is a key issue in system design, node design, and communication protocol development. In summary, efficient energy-conscious clustering and routing algorithms can potentially prolong the network lifetime.

20.1.2 Protocol Stack

The algorithms developed for wireless ad-hoc networks cannot be used for sensor networks, for several reasons. One is that the number of sensor nodes is typically much more than in a typical ad-hoc network, and sensor nodes, unlike ad-hoc nodes, are prone to permanent failure. In addition, sensor nodes normally use broadcast rather than point-to-point communication with its limited power and memory. Unlike computer networks, sensor nodes do not have global ID, since a typical packet overhead can be too large for them.

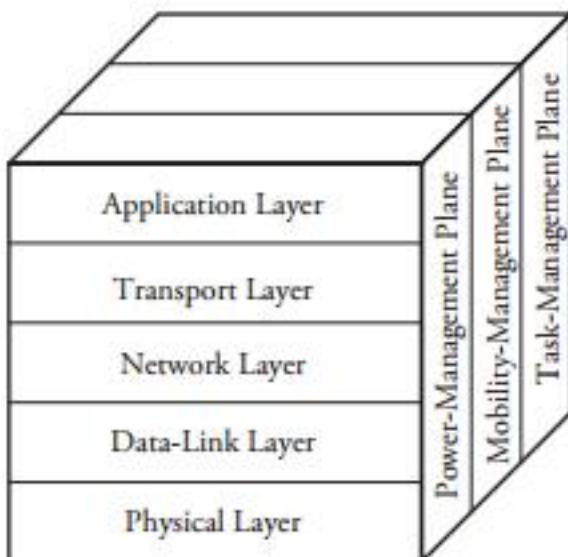


Figure 20.2 Sensor network protocol stack architecture

Figure 20.2 shows a protocol architecture for sensor networks. The protocol stack combines power efficiency and least-cost-path routing. This protocol architecture integrates networking protocols and power through the wireless medium and promotes cooperative efforts of sensor nodes. The protocol stack consists of the physical layer, data-link layer, network layer, transport layer, and application layer, backed by a power-management plane, mobility-management plane, and task-management plane. The physical layer is responsible for robust modulation, transmission, and receiving signals. *Media access control* (MAC) at the data-link layer must minimize packet collision with neighboring nodes, as power is a restricted factor. The network layer routes packets provided by the transport layer. The application layer uses software for preparation of data on an event. The power-management plane monitors the sensor's power level among the sensor nodes and manages the amount of power a sensor node has used.

Most of the sensor network routing techniques and sensing tasks require an accurate knowledge of location. Thus, a sensor node commonly has a location-finding system. A mobilizer may sometimes be needed to move sensor nodes to carry out assigned tasks. Sensor network routing protocols must be capable of self-organizing. For these purposes, a series of energy-aware MAC, routing, and clustering protocols have been developed for wireless sensor networks. Most of the energy-aware MAC protocols aim to either *adjust the transmission power* or *keep transceivers off as long as possible*.

20.1.3 Sensor Node Structure

Figure 20.3 shows a typical sensor node. A node consists mainly of a sensing unit, a processing unit and memory, a self-power unit, and a wireless transceiver component,

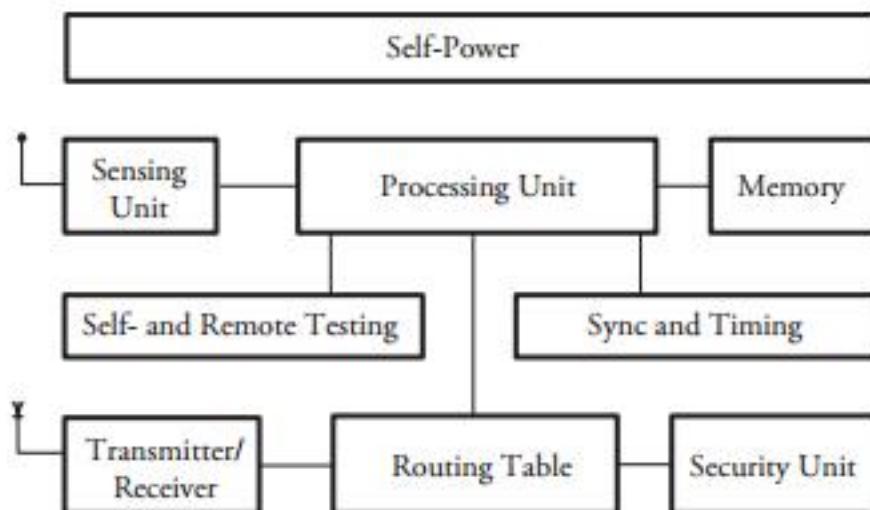


Figure 20.3 A typical wireless sensor node

as well as a self- and remote-testing unit, a synchronizing and timing unit, a routing table, and security units. Since nodes in a network are not physically accessible once they are deployed in the field, they are not worth being brought under test. An option is an on-board remote self-testing unit for the node on a routine basis.

Each node must determine its location. This task is carried out by a location-finding system based on the *global positioning system* (GPS). All the processes within the sensor node are synchronized by a local clocking and synchronizing system. The communication and security protocol units are in fact part of the processing unit. These two units are responsible for computing the best path for networking and security of the data being transmitted. The three main blocks of the sensor node—sensing unit, processing and memory unit, and power unit—are described in more detail in the following subsections.

Sensing Unit

The sensing unit consists of a sensor and an analog-to-digital converter. A smart sensor node consists of a combination of multiple sensors. The analog signals produced by the sensors, based on the observed event, are converted to digital signals by the converter and then fed into the processing unit. The sensing unit collects data externally and interacts with the central processor at the heart of the node.

Processing and Memory Unit

The *processing unit* performs certain computations on the data and, depending on how it is programmed, may send the resulting information out to the network. The processing unit, which is generally associated with memory, manages the procedures

that make the sensor node collaborate with the other nodes to carry out the assigned sensing task. The central processor determines what data needs to be analyzed, stored, or compared with the data stored in memory. The streamed data from the sensor input is processed as it arrives. The database in memory stores an indexed data list to be used as a reference to detect an event. Since sensing nodes are typically tiny and many nodes are engaged in a network, the communication structure makes use of a hierarchically arranged self-routing network through cluster heads.

In smart wireless sensor networks, a tiny processor and a tiny database are used in a node. Thousands of such nodes are spread on fields to power up the sensing task, as in the deployment of numerous small intelligent sensor nodes in a battlefield to monitor enemy movements. By inserting self-organizing capability into a sensor network, a smart node can extract data, compare it with the data stored in its memory database, and process it for relevance before relaying it to its central base station.

Self-Power Unit

A sensor node is supposed to be mounted in a small physical unit, limiting space for the battery. Moreover, the random distribution of sensors makes it impossible to periodically recharge or exchange batteries. In most types of sensor networks, the power unit in a sensor node is the most important unit of the node because the liveliness and existence of a node depend on the energy left in the node, and the routing in the sensor network is based on the algorithm that finds a path with the most energy. Thus, it is essential to use energy-efficient algorithms to prolong the life of sensor networks. The main task of the sensor node is to identify events, to process data, and then to transmit the data. The power of a node is consumed mainly in the transmitter and receiver unit. The sensor node can be supplied by a *self-power unit*, self-power unitbattery, or solar cells.

20.2 Communication Energy Model

IEEE standardsas 802.11a, b, and g provide a wide range of data rates: 54, 48, 36, 24, 18, 12, 9, and 6 Mb/s. This range reflects the trade-off between the transmission range and data rate intrinsic in a wireless communication channel. An accurate energy model is crucial for the development of energy-efficient clustering and routing protocols. The energy consumption, E , for all components of the transceiver in watts is summarized as

$$E = \theta + \eta \omega d^n, \quad (20.1)$$

where θ is the distance-independent term that accounts for the overhead of the radio electronics and digital processing, and $\eta \omega d^n$ is the distance-dependent term, in which

η represents the amplifier inefficiency factor, ω is the free-space path loss, d is the distance, and n is the environmental factor. Based on an environmental condition, n can be a number between 2 and 4, and η specifies the inefficiency of the transmitter when generating maximum power ωd^n at the antenna. Clearly, the distance-dependent portion of total energy consumption depends on the real-world transceiver parameters, θ , η , and the path attenuation ωd^n . If the value of θ overshadows $\eta \omega d^n$, the reduction in the transmission distance through the use of multihop communication is not effective.

In theory, the maximum efficiency of a power amplifier is 48.4 percent. However, practical implementations show that the power-amplifier efficiency is less than 40 percent. Therefore, θ is calculated assuming that $\eta = 1/0.4 = 2.5$. Using Equation (20.1), we can express the energy consumption of a transmitter and a receiver, E_T and E_R , respectively, by

$$E_T = \theta_T + \eta \omega d^n \quad (20.2)$$

and

$$E_R = \theta_R, \quad (20.3)$$

where θ_T and θ_R are the distance-dependent terms for the transmitter and the receiver, respectively. Although maximum output power and total power consumption are provided in the manufacturer's data sheet, θ can be calculated the following formula:

$$\theta = \theta_{TX} + \theta_{RX} = (E_T - \eta \omega d^n) + E_R. \quad (20.4)$$

Example. Table 20.1 shows values of E_T and E_R based on a manufacturer's data sheet and θ and $\eta \omega d^n$ calculated for a selected chipset. Although path-attenuation energy increases exponentially by the transmission distance, the data illustrates that the static power consumption, θ , dominates the path loss. Clearly, this causes total power consumption to remain constant as the transmission distance increases. Standards 802.11a, 802.11b, and 802.11g have multirate capabilities. Although, sensor nodes in general generate data in low rates, they can transmit the information using wireless high-speed modulation and techniques.

Table 20.2 shows the expected data rate for the 802.11g wireless technology. Although exploiting the multirate capabilities of wireless standards has never been proposed for sensor networks, this technique can decrease the transmission energy for smaller distances by switching to higher data rates and keeping the transceiver on for a

Table 20.1 Energy consumption parameters

IEEE Standard	Max. Output Power, ωd^n (dBm)	Total Power Consumption (W)	θ (W)	$\eta \times \omega d^n$ (W)
802.11a	+14	1.85 (E_{TX}) 1.20 (E_{RX})	2.987	0.0625
802.11b	+21	1.75 (E_{TX}) 1.29 (E_{RX})	2.727	0.3125
802.11g	+14	1.82 (E_{TX}) 1.40 (E_{RX})	3.157	0.0625

Table 20.2 Expected data rate of IEEE 802.11g technology

Rate (Mb/s)	Maximum Range	Rate (Mb/s)	Maximum Range
1	100.00 m	18	51.00 m
2	76.50 m	24	41.25 m
6	64.50 m	36	36.00 m
9	57.00 m	48	23.10 m
12	54.00 m	54	18.75 m

shorter period of time. In this case, the energy in terms of Joule/bit reduces discretely as transmission distance shrinks:

$$E = \frac{1}{R} (\theta + \eta \omega d^n), \quad (20.5)$$

where R is the rate in bits/sec. Figure 20.4 shows energy consumption using 802.11g technology at the constant rate of 1 Mb/s and the same technology with the multirate extension. Owing to large values of θ compared to the maximum output power, single-rate communication energy consumption remains constant as the transmission distance increases, whereas the communication energy consumption for multirate transmission decreases for shorter transmission ranges. However, this scenario does not follow the model of ωd^n . Meanwhile, the multirate communication necessitates the presence of a robust rate-selection protocol.

Multi-Hop Communication Efficiency

Considering the impact of real-world radio parameters and multirate communication, we should reevaluate the effectiveness of multihop communications. Since a multirate communication reduces energy consumption for shorter distances by switching to

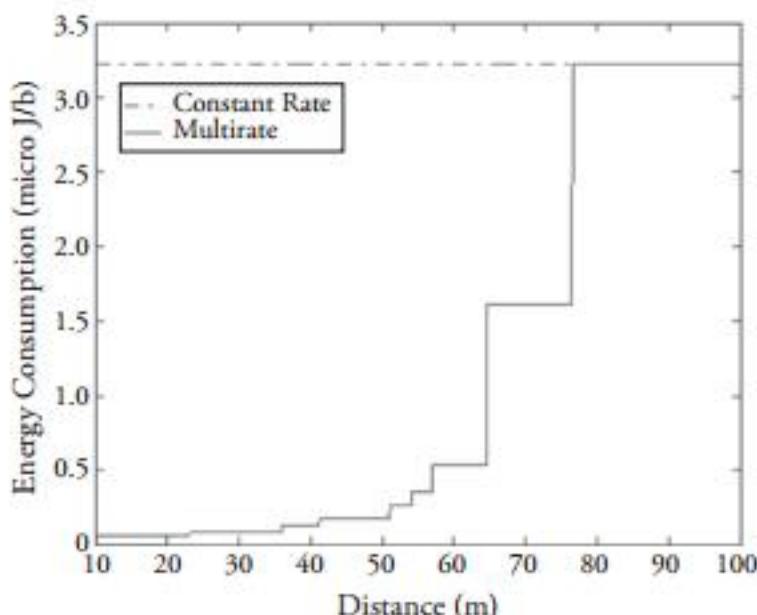


Figure 20.4 Energy consumption versus transmission distance for single-rate and multirate communication using 802.11g technology

higher data rates, multihop communication can conserve energy. The traditional objective of multihop communication is to divide the transmission distance into a number of hops, m , and to relatively conserve energy, considering Equation (20.3), by means of

$$E = m \left(\theta + \omega \left(\frac{d}{m} \right)^n \right). \quad (20.6)$$

However, if the division of transmission distance happens when the maximum range is less than 18.75 m for standard 802.11g, the data rate remains constant, and total energy consumption multiplies by the number of hops. Since sensor networks deal with two- or even three-dimensional spaces, multihop efficiency depends on the network scale and density.

Example. Figure 20.5 shows an organization in which sensor nodes A, B, C, D, and E are placed d meters apart and tend to send their data packets to the *cluster head* (CH). Note that d is an application-dependent parameter and can be chosen based on the sensor's characteristics. Assume that standard 802.11g technology is used in an environment in which sensors are placed on average no more than 10 meters apart. Compare nodes' energy consumptions, using Figure 20.5.

Solution. With the choice of 10 meters for d in the 802.11g charts, if node B tries to use node A as a relay node and then sends data to the cluster head, the total energy

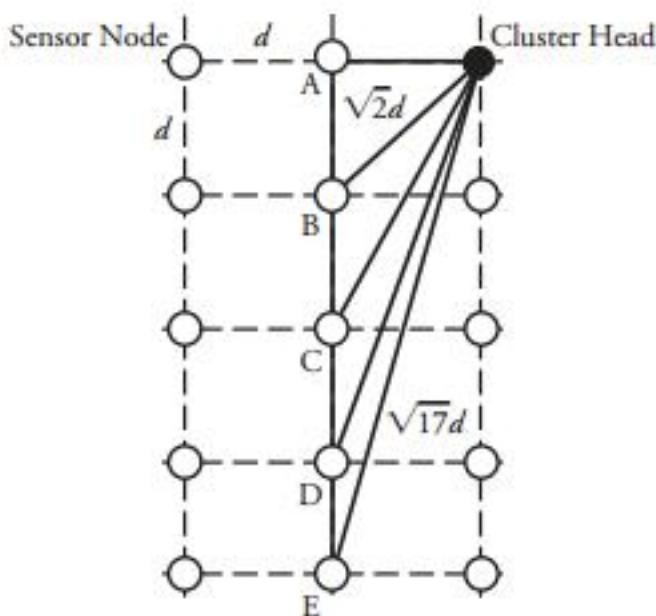


Figure 20.5 Cluster-head distances from sensor nodes A, B, C, D, and E in a two-dimensional model

of the chosen two-hop path is larger than the direct-transmission energy obtained as

$$E(\sqrt{2}d) = 0.0517 < E(d) + E(d) = 0.1034. \quad (20.7)$$

Also, for nodes C and D, there is no multihop path that can lead to better energy consumption than the direct communication path:

$$E(\sqrt{5}d) = 0.0581 < E(\sqrt{2}d) + E(d) = 0.1034 \quad (20.8)$$

and

$$E(\sqrt{10}d) = 0.0775 < E(\sqrt{5}d) + E(d) = 0.1098. \quad (20.9)$$

But if node E first sends the data to the intermediate node D, total energy consumption will be less than the direct communication path:

$$E(\sqrt{17}d) = E(41.23) = 0.1789 > E(\sqrt{10}d) + E(d) = 0.1292 \quad (20.10)$$

Node E is 41.23 meters away from the cluster head. This shows that for nodes more than 41.23 meters apart, direct transmission is no longer the best-possible communication method.

Example. Continuing the previous example using 802.11g technology, set up an environment representing one cluster. The dimension of the field is 50 m × 50 m, and

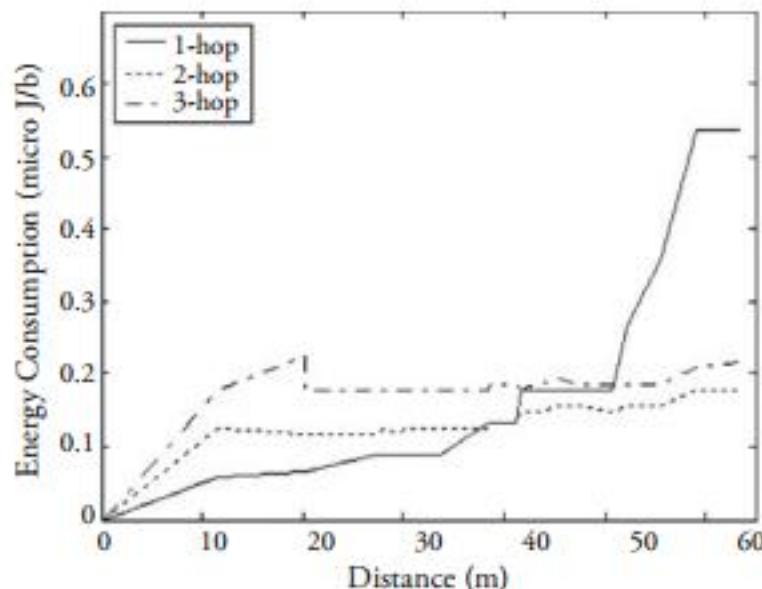


Figure 20.6 Communication energy versus distance from cluster head for 802.11g technology

25 nodes are randomly dispersed in the field. Compare the energy consumption of direct and multihop communication inside the cluster.

Solution. At this point, we assume that the cluster head is chosen randomly among the sensors. (The details of cluster-head selection algorithms explained in Section 20.3.) Figure 20.6 shows the energy consumption of a direct, minimum-energy two-hop and minimum-energy three-hop path, based on the distance between nodes in the cluster and the cluster head. For 802.11g technology, the direct transmission is the optimum choice for ranges less than 37 meters, which is almost the same as the result from analytical calculations (41 m). However, for ranges greater than 37 meters, the minimum-energy two-hop path can lead to significantly lower energy consumption.

20.3 Clustering Protocols

Clustering protocols specify the topology of the hierarchical nonoverlapping *clusters* of sensor nodes. A robust clustering technique is essential for self-organizing sensor networks. An efficient clustering protocol ensures the creation of clusters with almost the same radius and cluster heads that are best positioned in the clusters. Since every node in a clustered network is connected to a cluster head, route discovery among cluster heads is sufficient to establish a feasible route in the network. For a large sensor network, clustering can simplify multihop route discovery and limit the number of transmissions compared to a flat, nonclustered network.

20.3.1 Classification of Clustering Protocols

Clustering techniques can be either *centralized* or *decentralized*. Centralized clustering algorithms require each sensor node to send its individual information, such as energy level and geographical position, to the central base station. Based on a predefined algorithm, a base station calculates the number of clusters, their sizes, and the cluster heads' positions and then provides each node with its newly assigned duty.

Given the assumption that sensor networks might consist of thousands of nodes, it is impractical, if not impossible, for a base station to collect information about every node in the network prior to route setup. Therefore, centralized clustering is not an option for large sensor networks. Since a sensor node begins a clustering process without any knowledge about its location relative to the corresponding base station, a clustering algorithm should be able to form clusters without the help of the base station and knowledge of node positioning. Although location-finder devices can also be deployed to perform this task, they are often either costly or add too much overhead on the network.

Decentralized clustering techniques create clusters without the help of any centralized base station. An energy-efficient and hierarchical clustering algorithm can be such a way whereby each sensor node becomes a cluster head with a probability of p and advertises its candidacy to nodes that are no more than k hops away from the cluster head. Given the limited transmission range of wireless sensor nodes, a hierarchical structure with an arbitrary number of levels has its limitations. As the number of hierarchical levels grows, the distance between upper-level cluster heads may increase to the point that they are no longer able to communicate with one another. The *Low-Energy Adaptive Clustering Hierarchy* (LEACH) algorithm and the *Decentralized Energy-Efficient Cluster Propagation* (DEEP) protocol are two examples of the decentralized clustering protocols and are explained next.

20.3.2 LEACH Clustering Protocol

The *Low-Energy Adaptive Clustering Hierarchy* (LEACH) protocol is a decentralized clustering algorithm that does not offer a complete energy-optimization solution, as it has no strategy for specifying cluster-head positioning and distribution. LEACH is an application-specific protocol architecture that aims to prolong network lifetime by periodic reclustering and change of the network topology.

LEACH is divided into *rounds* consisting of a clustering phase and a steady-state phase for data collection. At the start of each round, a sensor node randomly chooses a number between 0 and 1 and then compares this number to a calculated threshold

called $T(n)$. If $T(n)$ is larger than the chosen number, the node becomes a cluster head for the current round. The value $T(n)$ is calculated using the following formula:

$$T(n) = \begin{cases} \frac{p}{1-p(r \bmod (1/p))} & \text{for } n \in G \\ 0 & \text{otherwise} \end{cases}, \quad (20.11)$$

where p is the ratio of the total number of cluster heads to the total number of nodes, r is the number of rounds, and G is a set of nodes that have not been chosen as cluster heads for the last $1/p$ rounds. For the first round ($r=0$), $T(n)$ is equal to p , and nodes have an equal chance to become cluster head. As r gets closer to $1/p$, $T(n)$ increases, and nodes that have not been selected as cluster head in the last $1/p$ rounds have more chance to become cluster head. After $1/p - 1$ rounds, $T(n)$ is equal to 1, meaning that all the remaining nodes have been selected as cluster head. Thus, after $1/p$ rounds, all the nodes have had a chance to become a cluster head once. Since being the cluster head puts a substantial burden on the sensor nodes, this ensures that the network has no overloaded node that runs out of energy sooner than the others.

After cluster heads are self-selected, they start to advertise their candidacy to the other sensor nodes. When it receives advertisements from more than one cluster-head candidate, a sensor node starts to make a decision about its corresponding cluster head. Each node listens to the advertisement signals and chooses the candidate whose associated signal is received with higher power. This ensures that each sensor node chooses the closest candidate as cluster head. The LEACH algorithm is distributed, as it can be accomplished by local computation and communication at each node, rather than the transmission of all the nodes' energy level and geographical position to a centralized point. However, cluster heads are chosen randomly, and there is no optimization in terms of energy consumption.

20.3.3 DEEP Clustering Protocol

The *Decentralized Energy-Efficient Cluster Propagation* (DEEP) protocol that establishes clusters with uniformly distributed cluster heads. This protocol balances the load among all the cluster heads by keeping the clusters' radii fairly equal. This protocol is completely decentralized, and there is no need for any location-finder device or hardware. The protocol starts with an initial cluster head and forms new cluster-head candidates gradually by controlling the relative distance between a pair of cluster heads and the circular radius of each cluster. Owing to the balanced load among cluster heads, periodic reclustering is not necessary, and operational expenses caused by frequent reclustering are therefore eliminated.

An efficient path-selection algorithm for nodes that are placed more than ℓ meters away from a cluster head is necessary in order to find the optimum two-hop or three-hop path. Although direct transmission to a cluster head can eliminate the overhead created by the route set-up packets, its efficiency is questionable, owing to the limited transmission range. In order to avoid the frequent control signal transmission and extra power consumption associated with that, a cluster head can be placed at the center of the cluster, with sensor nodes positioned closer than ℓ meters around it. In this case, cluster members can send the data packets directly to the cluster head without the need for any route set-up protocol, while efficiency has already been achieved through the choice of cluster shape and cluster size.

In order to explain the details of this algorithm, control signals and protocol parameters need to be introduced:

- Control signals: (1) cluster-head declaration signal or (2) cluster-head exploration signal
- Membership search signal with control parameters: declaration range (d_r), exploration range (d_{r1}, d_{r2}), minimum number of members (m_n), E_{rc1} , and E_{rc2} .

Protocol-control parameters are application-specific choices and can be defined prior to network deployment. DEEP forms clusters by starting with an initial cluster head that can be chosen prior to network deployment. This initial cluster head starts the cluster set-up phase by propagating cluster-head declaration signals within the range of d_r . This means that the cluster-head candidate chooses an appropriate data rate and signal output power so that it can reach nodes that are less than d_r away from the sender.

At this point, sensor nodes that receive the declaration signal accept the corresponding cluster head as a leader. They can then estimate their relative distance to the candidate by looking at the received signal's energy level. Once they know the relative distance to the cluster head, they can conserve energy by adjusting the transmission speed to the appropriate value and switching to sleep mode. Now, the initial cluster-head candidate propagates the cluster-head exploration signal within the range of d_{r2} , as shown in Figure 20.7. All the sensor nodes in this range can listen to the exploration signal, but only nodes that have never played the role of a cluster head and verify the following inequality are chosen as new candidates:

$$E_{rc1} < E_r < E_{rc2}, \quad (20.12)$$

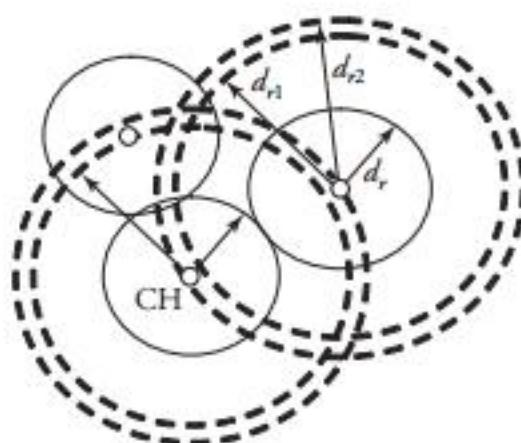


Figure 20.7 Initial cluster head starts the advertisement process. New cluster-head candidates send the exploration signal within the range of d_{r2} to continue the process of cluster establishment.

where E_r is the received signal energy. Note that E_{rc1} and E_{rc2} are fixed protocol parameters that can be precalculated and stored in the sensor-node memory, using the following formula:

$$E_{rc1} = P_{out} - \omega d_{r1}^n \quad (20.13)$$

and

$$E_{rc2} = P_{out} - \omega d_{r2}^n, \quad (20.14)$$

where P_{out} is the constant output power of the cluster-head exploration signal, and ω and n are parameters that can be determined based on the environmental conditions of the deployment area. This way, any of these nodes can consider itself a candidate. This ensures that new cluster-head candidates are positioned between d_{r1} and d_{r2} , away from the initial cluster head.

After a new cluster-head candidate is assigned, it sends a declaration signal within the range of d_r to find new cluster members. If two candidates can hear each other's declaration signal, they are too close to each other to be considered cluster-head candidates. Therefore, one of them is eliminated through a negotiation phase. Whenever it receives a declaration signal, a cluster head informs the sender of the message, using an acknowledgment message. A cluster head that receives the acknowledgment sends a dissolution message and informs all nodes within the range of d_r about its elimination. A node that receives a declaration signal from more than one candidate chooses the candidate whose associated signal is received with a higher power.

At this point, all confirmed cluster heads propagate exploration signals and search for new cluster-head candidates. Nodes that have already been chosen as cluster head or member ignore the cluster-head exploration or declaration signals. Therefore, this advertisement process terminates automatically when all the nodes in the field belong to a cluster. At this point, the algorithm might have produced some clusters with a very small number of members. Therefore, a cluster whose total number of members is smaller than the minimum number of members, m_n , is dissolved, and all its members, including its cluster head, initiate a *membership-search signal*.

After this process is completed, nodes listen to the responses from the local cluster heads and choose the closest cluster head, based on the received signal power. At the end, if the timeout has been reached, a sensor node that has not received any control signal sends a *membership-search signal* and chooses the closest cluster head as leader. The following algorithm summarizes the core segment of the DEEP protocol.

Begin DEEP Clustering Algorithm

1. **Initialize:** Initial cluster head finds cluster members by sending "cluster-head declaration."
2. Initial cluster head finds new cluster-head candidates by sending "cluster-head exploration signal."
3. **Repeat:** Cluster-head candidates that are placed on the (d_{r1}, d_{r2}) ring find cluster members.
4. Nodes that receive more than one cluster-head declaration choose the closest cluster head, based on the received signal energy.
5. Cluster-head candidates that receive a cluster-head declaration signal negotiate with the sender, and one of them gets eliminated.
6. Confirmed cluster heads send "cluster-head exploration" signals to find new cluster-head candidates (Go to step 4).
7. **Finalize:** If the number of members in a cluster is less than m_n , all the members find new clusters by sending the membership-search signal.
8. At the end, a node that has not received any control signal sends the membership-search signal. ■

DEEP has several advantages over other clustering protocols. With DEEP, a sensor node can either select itself as a cluster head by receiving a cluster-head exploration signal or join a cluster by receiving a cluster-head declaration signal. After the execution of the protocol, all the sensor nodes are covered and belong to only one cluster. This clearly shows that this protocol is completely decentralized. In addition, for the execution of

DEEP, there is no need for any location-finder hardware, such as the *global positioning system* (GPS) or a position-estimation protocol that puts extra overhead on sensor nodes.

DEEP can control a cluster-head distribution across the sensor network through protocol-execution methodologies. For example, cluster-head candidates should receive the cluster-head exploration signal with a certain amount of energy; if they can hear the declaration signal of each other, one of the candidates is eliminated. Communication cost is low through proper selection of protocol parameters, such as declaration range, exploration range, and minimum number of members.

With DEEP, *intracluster* communication is controlled by cluster heads, and nodes transmit their data directly to cluster heads. Therefore, no additional control signal is associated with route selection and maintenance inside the cluster. Also, owing to the uniform distribution of cluster heads, communication cost of a direct transmission path between a pair of neighboring cluster heads is almost identical across the sensor field. This is one of the most important protocol characteristics contributing to convenient deployment of an *intercluster* routing protocol.

20.3.4 Reclustering

In order to prevent overutilization of some sensor nodes, clustering technique should ensure that the cluster-head responsibility rotates among all sensor nodes. To achieve this, reclustering is performed periodically in LEACH. However, every round of reclustering requires several control-signal exchanges among self-elected cluster heads and sensor nodes. The reclustering process in DEEP is based on one small shift in the initial cluster head. When the current period of cluster setting is finished, the current initial CH chooses the nearest node that has never acted as an initial cluster head. This newly chosen initial cluster head starts the clustering process and creates a totally different cluster-head constellation.

20.4 Routing Protocols

After clusters with well-distributed cluster heads have been established in a network, energy-conscious routing is essential in order to set communication routes among cluster heads in a two-level hierarchical system. Similar to computer networks, routing protocols in sensor networks can be classified as either *intracluster* or *intercluster*. This section looks at both categories.

The fundamental concept behind them is much the same as the concept behind intradomain and interdomain routings (see Chapter 7). Assuming that every node in a

cluster can act as a relay node, there could be a large number of possible routes from a source to a sink. Because of the limited transmission range associated with low-power wireless technologies cluster-head packets cannot reach the base station unless other cluster heads act as relay nodes. Two major approaches can be used for routing and selecting the best path in a sensor network, as follows:

1. *Centralized routing*, whereby the routing decision is made by a single command center
2. *Distributed routing*, whereby the routing decision is made in a distributed fashion by several entities

Distributed routing algorithms are further classified as *proactive* or *reactive*. With proactive routing algorithms, such as link-state routing and distance-vector routing, nodes keep a routing table that contains next-hop information to every node in the network. Reactive routing protocols set a route to the desirable destination only when it is needed. Note that none of the ad hoc network protocols explained earlier consider energy consumption.

Another group of on-demand reactive routing protocols address the exclusive issues of wireless sensor network. For example, *directed diffusion* introduces a concept of “interest” propagation whenever a node wants to send data or a source needs to ask for it. With this type of protocol, flooding the network with interest signals establishes a path from a sink to every possible source (spanning tree).

20.4.1 Intracluster Routing Protocols

A routing algorithm within a cluster can be either *direct* or *multihop*. In a direct routing algorithm, the cluster head as the destination for all cluster nodes is located in the center of the cluster, so all nodes can communicate with the cluster head directly, as shown in Figure 20.8. Note that in this figure, two nodes cannot reach the destination, as they are located far from it. The number shown in each node indicates the level of energy the corresponding node has.

In a multihop routing algorithm, a node can face multiple hops in order to reach the destination. If a multihop algorithm is used for the centralized clustering procedure, the algorithm aims to choose the appropriate next neighbor for each node, using a central command node. Typically, a central command node collects the information about direct paths’ costs and geographical positions of the nodes and finds the best path.

Figure 20.9 shows a routing implementation. Sensor nodes are usually scattered in the field. A packet from a node is routed to a neighboring node that exhibits the

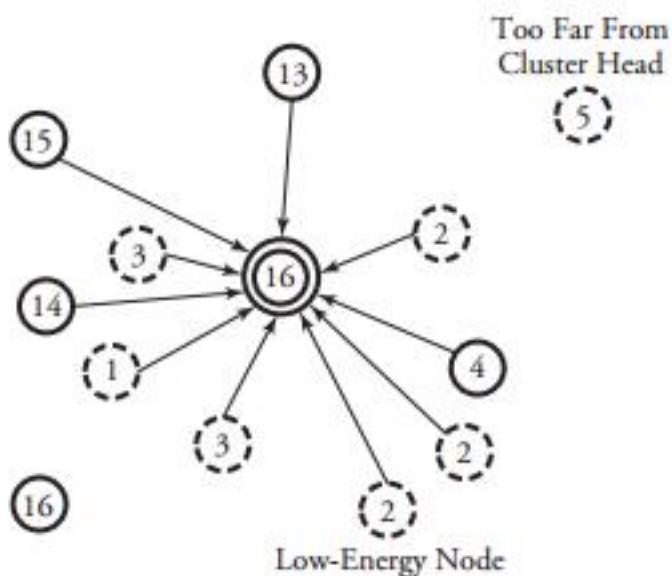


Figure 20.8 Direct routing in a cluster. The number associated with each node indicates a normalized value of the remaining energy in that node.

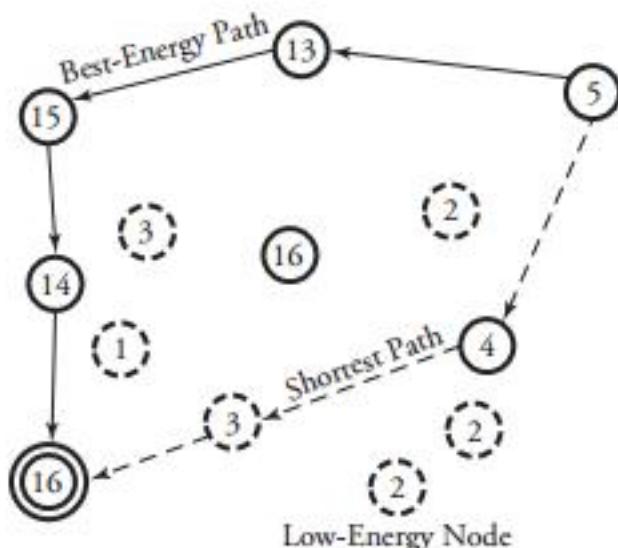


Figure 20.9 Multihop routing in a cluster in which the number associated with each node indicates a normalized value of the remaining energy in that node

highest amount of energy. The energy is an indication of the node's battery level. The number associated with each node indicates a normalized value of the remaining energy in that node. Figure 20.9 shows two paths from a node to a cluster-head node. One path involves the shortest distance in terms of hop counts; the other one uses the highest-energy route. The challenge here is to find the best path that suits the rapid and secure

deployment of data. Data is routed to the cluster head as shown in Figure 20.1, where the cluster head may communicate with the base station via radio frequencies.

The least-cost algorithm and the best-energy path can be modeled and compared for a network to provide a behavioral benchmark. The model can determine all possible paths available between a given source and destination. The energy of each node and hence all possible least-cost paths are computed regularly by cluster heads and propagated to all cluster nodes for database updating. During the phase of path finding to a cluster head, the routing algorithm accepts a failing (low-energy) node and finds the least-cost path, taking the failing node into account. The inputs for route process then include source node, destination node, failing nodes, and all other nodes.

20.4.2 Intercluster Routing Protocols

Intercluster protocols are not typically different from the multihop ones for intradomain cases. Interdomain protocols are available for

- Intercluster energy conscious routing (ICR)
- Energy-aware routing (EAR)
- Direct diffusion

ICR uses interest flooding similar to directed diffusion and EAR to establish routes between the base station and sensor nodes but differs from EAR and directed diffusion in some aspects.

Intercluster Energy-Conscious Routing (ICR)

ICR is a destination-initiated reactive routing protocol. This means that a destination, local base station (LBS), initiates an explicit route-discovery phase, which includes the propagation of an *interest* signal that floods throughout the network and establishes energy-efficient routes. Based on the application, which can be either periodic data collection or event driven, the interest signal can include the *type* and the *period* of the desired data shown in Figure 20.10. For an application requiring information from specific locations, the interest signal also includes the position of the required information.

Type	Period	Sender's Address	Cost Field

Figure 20.10 Interest-signal structure in a packet

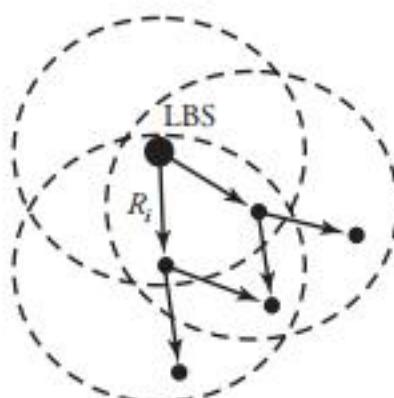


Figure 20.11 LBS starts route discovery by generating interest signals.

If the LBS requires some periodic data collection, it sets the *period* in which nodes send the specific *type* of information. Monitoring and surveillance applications are examples for the data-collection paradigm. If it requires sensor nodes to detect one specific event, an LBS includes the *type* of the event in the interest signal. Following the route-discovery phase, sensor nodes switch to sleep mode and wait for the specific event. In case of event detection, non-cluster-head nodes send the data directly to the associated cluster head, which uses the previously established route to send the information back to the LBS. In short, ICR occurs in two phases: *route discovery* and *data acquisition*.

In *route discovery*, the local base station initiates route discovery by sending an interest signal within the range of R_i . The value of R_i should be both high enough to keep the cluster-head network connected and low enough to prevent unnecessary energy consumption and interest generation. Owing to even distribution of cluster heads achieved by a clustering protocol, R_i can be chosen slightly bigger than the average distance between a pair of adjacent cluster heads. The LBS should adjust its output power and data rate of the interest signal to limit its transmission range to R_i . Also, the cost field is set to zero before interest propagation starts.

Since the distance between a pair of neighboring cluster heads is approximately the same across the network, the communication energy consumption associated with two distinct adjacent cluster heads is also the same. Therefore, the cost, or weight, of a multihop path is defined exclusively by the number of hops. In addition, the remaining energy in the cluster heads along the path affects the route-selection decision. The total-cost function C is defined as

$$C = \alpha h + \beta \sum_i \frac{B_M}{B_{ri}}, \quad (20.15)$$

where h is the hop number and B_{ri} represents the remaining energy in the battery of node i , B_M shows the maximum battery capacity of a sensor node, and α and β are normalization factors. The second part of the cost field favors the paths that include nodes with higher energy. To update the cost field, each intermediate cluster head calculates the inverse of its remaining battery power plus 1 (increment in the number of hops) and adds the outcome to the existing cost value.

Each intermediate cluster head that receives the interest signal saves the interest in its memory, including the address of the nodes that sent the message. Then the node should update the cost field of the outgoing interest signal and send it within the range of R_i . All the cluster heads within this range around the sender can hear the incoming signal. If it receives an interest signal that currently exists in memory but the sender's address is different, a cluster head compares the cost field of the received signal with the cost field of the previously saved message. If the incoming interest signal includes a cost field smaller than the previously saved message, the node replaces the old interest entry, updates the cost field, and propagates the packet, since the new signal represents a shorter, or more energy-efficient, path. If the new interest signal represents a path with a higher number of hops, the node should destroy the packet.

The *data-acquisition phase* occurs after each cluster head collects the requested information from sensor nodes and compresses it into a packet with fixed length, searches for the neighbor's address in memory, and relays the packet to that neighbor. In order to reduce the diffusion of spare data bits in the network, relay nodes can receive the data packets, each of length L , from N nodes and aggregate them into one single packet of length L . This reduces the number of data bits forwarded by the relay node from NL to L . To enable data aggregation during the data-collection period, cluster heads that are closer to the base station—that is, the cost field of the saved interest message includes fewer hops—should wait for their neighbors to send their data packets and then compress the incoming information with their own data and send the packet with the fixed length to the relay neighbor.

Comparison of ICR and EAR

ICR is different from EAR in two aspects. In EAR, sensor nodes save and propagate most of the incoming interest signals and eliminate only the ones with a very high cost field. However, in ICR, every time that the cost field of the incoming interest message is higher than the previously saved one, the packet gets destroyed. This puts a limit on the generation of interest messages.

In EAR, in order to ensure that the optimal path does not get depleted and that the network degrades evenly, multiple paths are found between a source and a destination. Each node has to wait for all the interest signals to come and then calculates the average

cost between itself and the destination. Based on the average cost, each path is assigned a probability of being chosen. Depending on the probability, each time one of the paths is chosen, ICR assumes that data aggregation is executed among cluster heads, and no packet moves along the chosen path independently. This means that during the data-collection period, each cluster head aggregates the data from its N adjacent cluster heads and has to forward only one compressed packet rather than N distinct packets. After the execution of the routing protocol, a spanning tree is established that is rooted in the base station and connects all the cluster heads to the base station. Therefore, only the least-cost, or the optimum, path is a final, established route for each cluster head. This way, the degradation of the optimal path for each packet is prevented.

20.5 Case Study: Simulation of a Sensor Network

This section presents a case study that shows the implementation of DEEP and ICR for a wireless sensor network spread over an area. The network is used for monitoring and protecting the area. The basic objective is to deploy a large number of low-cost and self-powered sensor nodes, each of which acquires and processes data from a hazardous event and alerts a base station to take necessary action. In this scenario, 3,000 sensor nodes are randomly distributed in a field of $550\text{ m} \times 550\text{ m}$. Therefore, the density of sensor nodes is about one per $10\text{ m} \times 10\text{ m}$ area, which is the maximum detection range for the hazard sensors.

MAC assigns a unique channel for every node and prevents possible collisions. With this assumption, we extracted the MAC layer from our simulations, and data packets were sent directly from the network layer of one node to the network layer of the neighbor. We simulated the DEEP algorithm, using parameters d_r , d_{r1} , d_{r2} , and m , and put the initial cluster head at the center of the field.

20.5.1 Cluster-Head Constellation and Distribution of Load

Figure 20.12 shows the result of the simulation with parameters $d_r = 30\text{ m}$, $d_{r2} = 80\text{ m}$, $d_{r1} = 78\text{ m}$, $m = 14$. Based on the results obtained from Section 20.2, the distance of 30 meters is an initial choice for d_r . In order to avoid overlapping between clusters, the value of d_{r1} and d_{r2} should be more than twice the value of d_r . Since the average distance between sensor nodes in this application is 10 m, 80 m is a fair choice for d_{r2} . The width of the (d_{r1}, d_{r2}) ring should be large enough to accommodate new cluster-head candidates and small enough to avoid cluster-head candidates that are too close to each other. We chose an initial value 2 m for the ring width.

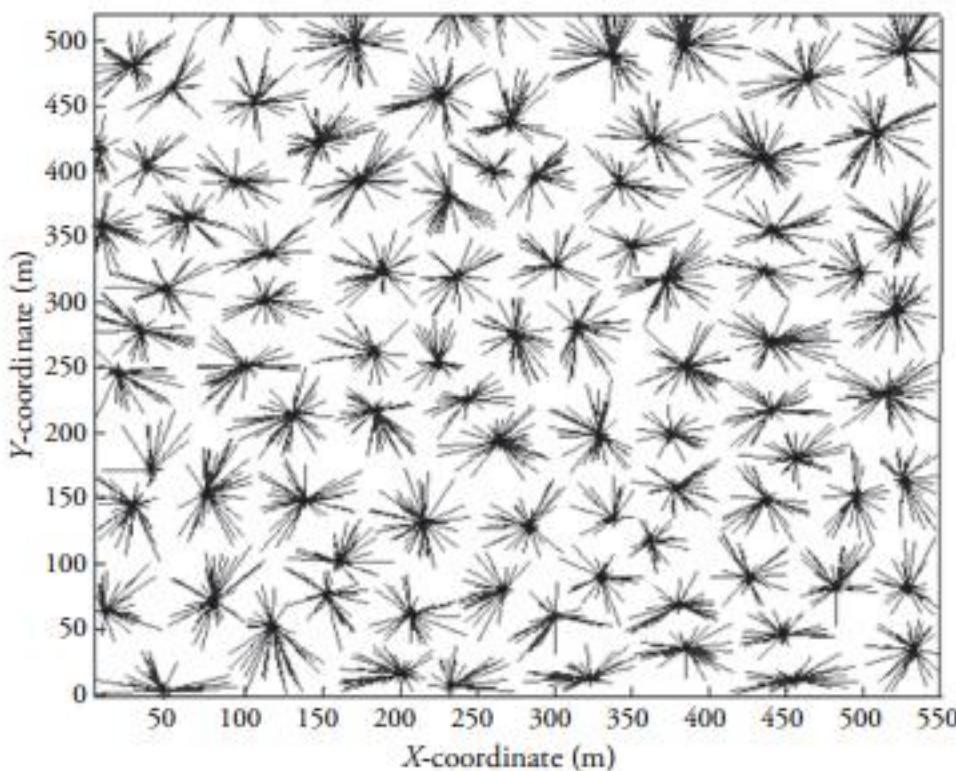


Figure 20.12 Simulation results on distributed clusters whose sensor nodes are directly connected to their associated cluster heads. The initial cluster head is put in the center of the sensor field, the simulation starts by advertising its candidacy, and cluster heads are gradually dispersed across the network.

In order to balance the load among cluster heads, DEEP controls the cluster-head distribution rather than the number of cluster members. Although cluster heads that manage more members should execute more signal processing for the sake of data aggregation, digital processing consumes much less energy than wireless transmission, and no overutilized cluster head is using this protocol.

Figure 20.13 demonstrates the cluster-head distribution achieved using LEACH and DEEP. Because of the random selection of cluster heads in LEACH, some of the cluster heads are too close to each other; others, too far. This type of cluster-head selection causes a lot of burden on some cluster heads and quickly drains their batteries. It can be shown that compared with LEACH, DEEP is capable of minimizing energy consumption associated with reclustering overheads more efficiently by reducing the number of necessary rounds.

20.5.2 Optimum Percentage of Cluster Heads

In order to determine the optimum cluster-head density and compare the performance of the routing protocol on both DEEP and LEACH, we used a 1,600-node network.

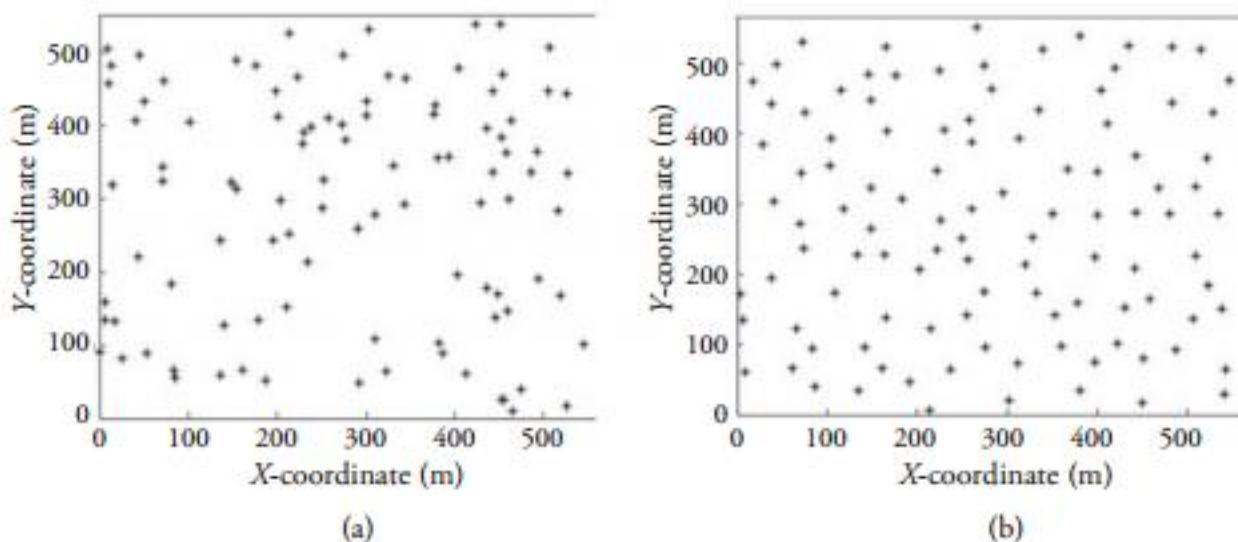


Figure 20.13 Comparison of cluster-head constellation between (a) LEACH and (b) DEEP. DEEP generates well-distributed cluster heads across the network.

Nodes were randomly distributed in a field $400\text{ m} \times 400\text{ m}$. In this scenario, sensor nodes send the information directly to their associated cluster head. Each cluster head compresses the data and waits for neighbor cluster-heads' data packets. Then, the cluster head compresses all the received data packets into a packet with fixed length and sends it to the relay neighbor. The relay neighbor address has been saved in node memory through the propagation of the interest signal. In-network data aggregation performed by cluster heads helps to reduce the amount of data dispersed in the network.

20.6 Other Related Technologies

Other sensor-network based technologies use low-power nodes. The one discussed in this section is the *ZigBee technology*.

20.6.1 Zigbee Technology and IEEE 802.15.4

The *ZigBee technology* is a communication standard that provides a short-range low-cost networking capability that allows low-cost devices to quickly transmit small amounts of data, such as temperature readings for thermostats, on/off requests for light switches, or keystrokes for a wireless keyboard. Other ZigBee applications are in professional installation kits for lighting controls, heating, ventilation, air conditioning, and security. Even the Bluetooth short-range wireless technology found in laptops and cellphones lacks the affordability, power savings, and mesh-networking capabilities of ZigBee.

ZigBee comes from higher-layer enhancements by a multivendor consortium called the Zigbee Alliance. IEEE standard 802.15.4/ZigBee specifies the MAC and physical layers. The 802.15.4 standard specifies 128-bit AES encryption; ZigBee specifies how to handle encryption key exchange. The 802.15.4/ZigBee networks run in the unlicensed frequencies, 900 MHz and 2.4 GHz band, based on a packet radio standard and support many cordless telephones, allowing data to be sent over distances up to 20 meters.

ZigBee devices, typically battery powered, can transmit information much farther than 20 meters, because each device within listening distance passes the message along to any other device within range. Only the intended device acts on the message. By instructing nodes to wake up only for those split-second intervals when they are needed, ZigBee device batteries might last for years. Although this technology is targeting for manufacturing, health care, shipping, and homeland defense, the ZigBee Alliance is initially keeping its focus small.

20.7 Summary

This last chapter focused on wireless sensor networks. Some applications of sensor networks are target tracking, environmental monitoring, system control, and chemical or biological detection.

The protocol stack for sensor networks concerned with power factor. The sensor network protocol stack combines two features: power-efficiency and least-cost-path routing. Thus, the protocol architecture integrates networking protocols and power efficiency through the wireless medium and promotes cooperative efforts among sensor nodes. The protocol stack consists of the physical, data-link, network, transport, and application layers, power-management, mobility-management, and task-management planes. The internal structure of an intelligent sensor node consists of three units for sensing, processing, and storage, respectively, and communications capabilities.

An energy model for a transceiver of a node can be developed. The energy consumption, E , for all components of a transceiver in watts can be modeled by $E = \theta + \eta\omega d^n$, where θ is the distance-independent term that accounts for the overhead of the radio electronics and digital processing, and $\eta\omega d^n$ is the distance-dependent term in which η represents the amplifier inefficiency factor, ω is the free-space path loss, and d is the distance.

Two clustering protocols in sensor networks are the *Low-Energy Adaptive Clustering Hierarchy* (LEACH) algorithm and the *Decentralized Energy-Efficient Cluster Propagation* (DEEP) protocol. DEEP is based on the idea of controlling the geographical dimensions of clusters and the distribution of cluster heads. Because of the balanced

load among cluster heads, there is no need for frequent re-clustering, but after current cluster heads are out of energy, the protocol can rotate the cluster-head position among all the sensor nodes. Also, the identical distance between a pair of neighboring cluster heads leads to the ease of route set-up deployment. After establishing well-distributed cluster heads and clusters in a network, energy-conscious routing is essential in order to set communication routes among cluster heads. *ZigBee technology*, which is based on the IEEE 802.15.4 standard, is a related technology that uses low-power nodes.

20.8 Exercises

1. Assume that the maximum line of sight for each node in the sensor network shown in Figure 20.9 is exactly 1 mile. The normalized maximum energy at each node is 20. Assume that 1 mile is scaled down to 1 inch on this map, measured between two centers of circles.
 - (a) Apply the cost function presented in Equation (20.15) for the best-energy path.
 - (b) Apply the cost function for the shortest path shown in the figure.
2. *Computer simulation project.* Consider a field of 100×100 square meter, on which 50 sensor nodes are randomly distributed. Apply DEEP, and use a mathematical tool, such as Matlab, to simulate the clustering process each at the size of five sensor nodes. Use randomly one of the four available levels of energy to each node.
 - (a) Sketch all clusters and the locations on cluster heads.
 - (b) Show on the same chart how cluster heads are connected to regular nodes.
3. *Computer simulation project.* Continue the project from exercise 2, and now assume that an event is detected by a sensor at the closest cluster to one of the field's corners. Sketch the intercluster routing to a base station situated at the opposite corner from the event cluster.
4. *Computer simulation project.* Continue the project from exercise 3, now using LEACH.

This page intentionally left blank

APPENDIX A

Glossary of Acronyms

AAL	<i>ATM application layer</i>
ABR	<i>associative-based routing</i>
ACC	<i>arriving-cell counter</i>
AE	<i>arbitration element</i>
AES	<i>Advanced Encryption Standard</i>
AODV	<i>Ad-Hoc On-Demand Distance Vector</i>
ARP	<i>Address Resolution Protocol</i>
ARQ	<i>automatic repeat request</i>
ASN.1	<i>abstract syntax notation one</i>
ATM	<i>asynchronous transfer mode</i>
BISDN	<i>broadband integrated services digital network</i>
BBC	<i>buffer-control bit circuit</i>
BCC	<i>buffer control circuit</i>
BCN	<i>broadcast channel numbers</i>
BER	<i>bit error rate</i>
BGMP	<i>Border Gateway Multicast Protocol</i>
BGP	<i>Border Gateway Protocol</i>
BPDU	<i>bridge protocol data unit</i>
BS	<i>bit slice</i>

BTC	<i>broadcast translation circuit</i>
CBR	<i>constant bit rate</i>
CCITT	<i>International Consultative Committee for Telecom and Telegraphy</i>
CD	<i>column decoder</i>
CDMA	<i>code-division multiple access</i>
CDN	<i>content distribution network</i>
CGSR	<i>cluster-head gateway switch routing</i>
CH	<i>cluster head</i>
CIDR	<i>classless interdomain routing</i>
CMOS	<i>complimentary metal-oxide semiconductor</i>
CN	<i>copy network</i>
CRC	<i>cyclical redundancy check</i>
CSMA	<i>carrier sense multiple access</i>
DEEP	<i>decentralized energy-efficient cluster propagation</i>
DQDB	<i>distributed queue dual bus</i>
DNS	<i>domain name system</i>
DRR	<i>deficit round robin</i>
DS	<i>differentiated service</i>
DSDV	<i>destination-sequenced distance vector</i>
DSF	<i>distributed coordination function</i>
DSSS	<i>direct-sequence spread spectrum</i>
DSR	<i>dynamic source routing</i>
DVMRP	<i>distance vector multicast routing protocol</i>
EDF	<i>earliest deadline first</i>
FCFS	<i>first come, first served</i>
FDM	<i>frequency division multiplexing</i>
FDMA	<i>frequency division multiple access</i>
FEC	<i>forward equivalence class</i>
FHSS	<i>frequency-hopping spread spectrum</i>
FIFO	<i>first-in first-out</i>
FOL	<i>fiber optic link</i>
FTP	<i>file transfer protocol</i>
GFC	<i>generic flow control</i>
GPS	<i>global positioning system</i>

GIF	<i>graphics interchange format</i>
GSS	<i>grant signal selector</i>
HEC	<i>header error control</i>
HD	<i>header decoder</i>
HDTV	<i>high-definition television</i>
HTTP	<i>hyper-text transfer protocol</i>
ICMP	<i>Internet control message protocol</i>
IID	<i>independent and identically distributed</i>
IGMP	<i>Internet group management protocol</i>
IPP	<i>input port processor</i>
IPv6	<i>Internet protocol version 6</i>
JPEG	<i>joint photographic experts group</i>
L2TP	<i>layer 2 tunneling protocol</i>
LBC	<i>local buffer controller</i>
LDP	<i>label distribution protocol</i>
LEACH	<i>low-energy adaptive clustering hierarchy</i>
LIB	<i>label information base</i>
LLC	<i>logic link control</i>
LSP	<i>label switch path</i>
LSR	<i>label switch routers</i>
MAC	<i>medium access control</i>
MBGP	<i>multi-protocol extensions to BGP</i>
MBC	<i>multipath buffered crossbar</i>
MIB	<i>management information base</i>
MOSPF	<i>multicast open shortest path first</i>
MPEG	<i>moving pictures expert group</i>
MPLS	<i>multi-protocol label switching</i>
MSDP	<i>multicast source discovery protocol</i>
MTU	<i>maximum transmission unit</i>
MSN	<i>Manhattan street network</i>
NDS	<i>network data slices</i>
NNI	<i>network-node interface</i>
NVT	<i>network virtual terminal</i>
OC	<i>optical carrier</i>

OCC	<i>output control circuit</i>
OD	<i>output port decoder</i>
OFDM	<i>orthogonal frequency division multiplexing</i>
OPP	<i>output port processor</i>
OSI	<i>open systems interconnection</i>
OSPF	<i>open shortest path first</i>
OXC	<i>optical cross connect</i>
PAM	<i>pulse amplitude modulation</i>
PCF	<i>point coordination function</i>
PDF	<i>probability density function</i>
PDU	<i>protocol data units</i>
PFC	<i>priority field circuit</i>
PIM	<i>protocol independent multicast</i>
PHB	<i>per-hop behavior</i>
PMF	<i>probability mass function</i>
P2P	<i>peer to peer</i>
PPP	<i>point-to-point protocol</i>
PPTP	<i>point-to-point tunneling protocol</i>
PPM	<i>pulse position modulation</i>
PWM	<i>pulse width modulation</i>
QAM	<i>quadrature amplitude modulation</i>
QoS	<i>quality of service</i>
QPSK	<i>quadrature phase shift keying</i>
RARP	<i>reverse address resolution protocol</i>
RCB	<i>receive buffer</i>
RCYC	<i>recycling buffer</i>
RED	<i>random early detection</i>
RIP	<i>routing information protocol</i>
RN	<i>routing network</i>
RNG	<i>row number generator</i>
RPC	<i>request priority circuit</i>
RPF	<i>reverse path forwarding</i>
RSA	<i>Rivert, Shamir, and Aldeman</i>
RSQ	<i>resequencing buffer</i>

RSVP	<i>resource reservation protocol</i>
RTCP	<i>real-time control protocol</i>
RTP	<i>real-time protocol</i>
RTT	<i>round-trip time</i>
RVC	<i>request vector circuit</i>
SBC	<i>shared buffer crossbar</i>
SCP	<i>secure copy</i>
SCTP	<i>stream control transmission protocol</i>
SDC	<i>self-driven crosspoint</i>
SIP	<i>session initiation protocol</i>
SHF	<i>super-high frequency</i>
SMI	<i>structure of management information</i>
SMTP	<i>simple mail transfer protocol</i>
SNMP	<i>simple network management protocol</i>
SNR	<i>signal-to-noise ratio</i>
SONET	<i>synchronous optical nework</i>
SSH	<i>secure shell</i>
TCP	<i>transmission control protocol</i>
TDM	<i>time-division multiplexer</i>
TDMA	<i>time division multiple access</i>
TOS	<i>type-of-service</i>
TORA	<i>temporally ordered routing algorithm</i>
TP	<i>transport protocol</i>
TRIP	<i>telephone routing over IP</i>
UDP	<i>user datagram protocol</i>
UGC	<i>upstream grant circuit</i>
UHF	<i>ultra-high frequency</i>
UNI	<i>user-network interface</i>
URL	<i>uniform resource locator</i>
VBR	<i>variable bit-rate</i>
VCI	<i>virtual circuit identifier</i>
VLSI	<i>very large scale integration</i>
VPI	<i>virtual path identifier</i>
VPN	<i>virtual private network</i>

VXT	<i>virtual circuit translation table</i>
WDM	<i>wavelength division multiplexing</i>
WEP	<i>wired equivalent privacy</i>
WFQ	<i>weighted fair queuing</i>
WiMAX	<i>world-wide interoperability for microwave access</i>
WIS	<i>telephone switching office</i>
WMN	<i>wireless mesh networks</i>
WRP	<i>wireless routing protocol</i>
WTG	<i>waiting time generator</i>
WWW	<i>World Wide Web</i>
XMB	<i>transmit buffer</i>

APPENDIX B

RFCs

Requests for Comments (RFCs) are an informal series of reports about the computer communication protocols, including TCP/IP and other Internet architectures, wireless and mobile networks, and their history. RFCs are loosely coordinated sets of notes but are rich in information. RFCs are generally available online for public access.

Protocol	RFC
AODV	3561
ARP	826, 903, 925, 1027, 1293, 1329, 1433, 1868, 1931, 2390
BGP	1092, 1105, 1163, 1265, 1266, 1267, 1364, 1392, 1403, 1565, 1654, 1655, 1665, 1771, 1772, 1745, 1774, 2283
BOOTP and DHCP	951, 1048, 1084, 1395, 1497, 1531, 1532, 1533, 1534, 1541, 1542, 2131, 2132
CIDR	1322, 1478, 1479, 1517, 1817
DNS	799, 811, 819, 830, 881, 882, 883, 897, 920, 921, 1034, 1035, 1386, 1480, 1535, 1536, 1537, 1591, 1637, 1664, 1706, 1712, 1713, 1982, 2065, 2137, 2317, 2535, 2671

FTP	114, 133, 141, 163, 171, 172, 238, 242, 250, 256, 264, 269, 281, 291, 354, 385, 412, 414, 418, 430, 438, 448, 463, 468, 478, 486, 505, 506, 542, 553, 624, 630, 640, 691, 765, 913, 959, 1635, 1785, 2228, 2577
HTML	1866
HTTP	2068, 2109
ICMP	777, 792, 1016, 1018, 1256, 1788, 2521
IGMP	966, 988, 1054, 1112, 1301, 1458, 1469, 1768, 2236, 2357, 2365, 2502, 2588
IP	760, 781, 791, 815, 1025, 1063, 1071, 1141, 1190, 1191, 1624, 2113
IPv6	1365, 1550, 1678, 1680, 1682, 1683, 1686, 1688, 1726, 1752, 1826, 1883, 1884, 1886, 1887, 1955, 2080, 2373, 2452, 2463, 2465, 2466, 2472, 2492, 2545, 2590
MIB, MIME, IMAP	196, 221, 224, 278, 524, 539, 753, 772, 780, 806, 821, 934, 974, 1047, 1081, 1082, 1225, 1460, 1496, 1426, 1427, 1652, 1653, 1711, 1725, 1734, 1740, 1741, 1767, 1869, 1870, 2045, 2046, 2047, 2048, 2177, 2180, 2192, 2193, 2221, 2342, 2359, 2449, 2683, 2503
Multicast	1584, 1585, 2117, 2362
NAT	1361, 2663, 2694
OSPF	1131, 1245, 1246, 1247, 1370, 1583, 1584, 1585, 1586, 1587, 2178, 2328, 2329, 2370
RARP	826, 903.925, 1027, 1293, 1329, 1433, 1868, 1931, 2390
RIP	1131, 1245, 1246, 1247, 1370, 1583, 1584, 1585, 1586, 1587, 1722, 1723, 2082, 2453
SCTP	2960, 3257, 3284, 3285, 3286, 3309, 3436, 3554, 3708
SMTP, POP, IMAP	196, 221, 224, 278, 524, 539, 753, 772, 780, 806, 821, 934, 974, 1047, 1081, 1082, 1225, 1460, 1496, 1426, 1427, 1652, 1653, 1711, 1725, 1734, 1740, 1741, 1767, 1869, 1870, 2045, 2046, 2047, 2048, 2177, 2180, 2192, 2193, 2221, 2342, 2359, 2449, 2683, 2503

SNMP,	1065, 1067, 1098, 1155, 1157, 1212, 1213, 1229, 1231,
MIB,	1243, 1284, 1351, 1352, 1354, 1389, 1398, 1414,
SMI	1441, 1442, 1443, 1444, 1445, 1446, 1447, 1448,
	1449, 1450, 1451, 1452, 1461, 1472, 1474, 1537,
	1623, 1643, 1650, 1657, 1665, 1666, 1696, 1697,
	1724, 1742, 1743, 1748, 1749
TCP	675, 700, 721, 761, 793, 879, 896, 1078, 1106, 1110,
	1144, 1145, 1146, 1263, 1323, 1337, 1379, 1644,
	1693, 1901, 1905, 2001, 2018, 2488, 2580
TELNET	137, 340, 393, 426, 435, 452, 466, 495, 513, 529, 562,
	595, 596, 599, 669, 679, 701, 702, 703, 728, 764, 782,
	818, 854, 855, 1184, 1205, 2355
TFTP	1350, 1782, 1783, 1784
UDP	768
VPN	2547, 2637, 2685
WWW	1614, 1630, 1737, 1738

This page intentionally left blank

APPENDIX C

Probabilities and Stochastic Processes

Communication systems, particularly computer networks, often encounter random arrivals of tasks as packets. Such systems require analysis using the theory of probabilities, as seen in various chapters. This appendix reviews principles of probability theory, random variables, and random processes.

C.1 Probability Theory

Let's first consider a random experiment, such as producing random logic 0s and 1s. The *sample space* of the experiment, usually denoted by the symbol S , consists of the set of all possible *outcomes*, indicated by w . In this case, for integers 0 and 1, the sample space is $S = \{0, 1\}$. We define an *event*, A , to be a subset of sample space, which may consist of any number of sample points. Thus, if define event $A = \{1\}$, the event consists of only one point.

The *union* of two events, A and B , covers all outcomes belonging to both events and is shown by $A \cup B$. The *intersection* of events A and B refers to all outcomes shared between the two events and is shown by $A \cap B$. The *complement* of event A is an event that covers all events but the ones belonging to A and is shown by \bar{A} .

The probability of an event A is shown by $P[A]$ and always $0 \leq P[A] \leq 1$. Also, if $A \cap B = \emptyset$, then:

$$P[A \cup B] = P[A] + P[B]. \quad (\text{C.1})$$

In most engineering cases, especially in analyzing random signals and systems, we are often interested in the conditional probability. The probability that event A occurs given that event B has occurred is defined as

$$P[A|B] = \frac{P[A \cap B]}{P[B]}. \quad (\text{C.2})$$

Two events A and B are independent of each other if

$$P[A \cap B] = P[A]P[B]. \quad (\text{C.3})$$

C.1.1 Bernulli and Binomial Sequential Laws

Two fundamental laws of *sequential experiments* are *Bernulli* and *binomial* trials. A Bernulli trial is a sequence of repeated independent random experiments whose outcomes are either a success with probability p or a failure with probability $1 - p$. A binomial trial measures k successes, each with probability p in n independent Bernulli trials. As a result, the probability of having k successes in n trials is

$$P_n(k) = \binom{n}{k} p^k (1-p)^{n-k}, \quad (\text{C.4})$$

where $\binom{n}{k} = \frac{n!}{k!(n-k)!}$.

C.1.2 Counting and Sampling Methods

One can simply observe that the probability of an event A can also be computed through a counting method by:

$$P[A] = \frac{n_i}{n}, \quad (\text{C.5})$$

where n_i is the number of outcomes in event A, and n is the total number of outcomes in global sample space. We note that the larger n_i and n are, the more accurate the probability of A becomes.

C.2 Random Variables

Sometimes, the behavior of outcomes on a sample space whose values are real numbers must be considered. A *random variable*, $X(w)$, or simply X , is a function that maps each outcome of the sample space to real numbers. In the previous example, 0 and 1 are

real numbers, but they could have been interpreted as “success” and “fail” as outcomes. In such a case, the random variable maps them into 0 and 1, respectively.

C.2.1 Basic Functions

The probability that a random variable, X , takes a value that does not exceed a given number, x , is called the *cumulative distribution function* (CDF) of X . This is a function of x and is usually denoted by $F_X(x)$:

$$F_X(x) = P[X \leq x]. \quad (\text{C.6})$$

Random variables are either *discrete* or *continuous*. For a discrete random variable, we define, the *probability mass function* (PMF) to be the probability function of the random variable at any given number x . PMF is denoted by

$$P_X(x) = P[X = x]. \quad (\text{C.7})$$

It is obvious that for a discrete random variable X , CDF at any given point x is computed by adding all PMF values up to point x :

$$F_X(x) = \sum_{\substack{i = \text{all values up to } x}} P_X(i). \quad (\text{C.8})$$

Similarly, we can define the *probability density function* (PDF) for a continuous random variable to be the probability function of X . PDF is denoted by $f_X(x)$. Similarly, CDF and PDF are associated with each other through

$$F_X(x) = \int_{-\infty}^x f_X(x) dx. \quad (\text{C.9})$$

C.2.2 Conditional Functions

We can define the conditional CDF of a random variable X as a CDF given that an event A has already occurred. This CDF is obtained naturally by

$$F_X(x|A) = \frac{P[(X \leq x) \cap A]}{P[A]}. \quad (\text{C.10})$$

Similarly, the conditional PDF can be defined as

$$f_X(x|A) = \frac{d}{dx} F_X(x|A), \quad (\text{C.11})$$

and the conditional PMF can be defined as

$$P_X(x|A) = \frac{[P(X=x) \cap A]}{P[A]}. \quad (\text{C.12})$$

C.2.3 Popular Random Variables

Three popular discrete random variables are *Bernulli*, *binomial*, and *Poisson* random variables. Three popular continuous random variables are *uniform*, *Gaussian*, and *exponential*. All are briefly reviewed.

Bernulli Random Variable

Bernulli random variable X is discrete and is defined over a sample space $S_X = \{0, 1\}$, where 0 and 1 represent *failure* with probability p and *success* with probability $1 - p$, respectively. Therefore, PMF of this random variable is defined by

$$P_X(x) = \begin{cases} p & x = 0 \\ 1 - p & x = 1 \end{cases}. \quad (\text{C.13})$$

Binomial Random Variable

A *binomial random variable* X is discrete and is defined over a sample space $S_X = \{0, 1, \dots, n\}$. This random variable is basically n Bernulli random variables, and its PMF is obtained from

$$P_X(x) = \binom{n}{x} p^x (1-p)^{n-x}. \quad (\text{C.14})$$

Geometric Random Variable

A *geometric random variable* X is discrete and is defined over a sample space $S_X = \{1, 2, \dots, x\}$. This random variable is defined for counting x Bernulli random trials in a binomial random variable with only one success at the last trial. Consequently, its PMF is obtained from

$$P_X(x) = p(1-p)^{x-1}. \quad (\text{C.15})$$

Poisson Random Variable

A *Poisson random variable* X is discrete and is defined over a sample space $S_X = \{1, 2, \dots\}$. This random variable is approximated for a binomial random variable when

n large and p is small. With these conditions, its PMF is obtained from Equation (C.14), taking into account the mentioned approximations:

$$P_X(x) = \frac{\alpha^x e^{-\alpha}}{x!}. \quad (\text{C.16})$$

Uniform Random Variable

A *uniform random variable* X is continuous and is defined over a sample space $S_X = [a, b]$, where a and b are two constant numbers. PDF of a uniform random variable is obtained from

$$f_X(x) = \frac{1}{b - a}. \quad (\text{C.17})$$

Exponential Random Variable

An *exponential random variable* X is continuous and is defined over a sample space $S_X = [0, \infty)$. PDF of an exponential random variable is expressed by

$$f_X(x) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases}. \quad (\text{C.18})$$

Gaussian (Normal) Random Variable

A *Gaussian (normal) random variable* X is continuous and is defined over a sample space $S_X = [-\infty, +\infty)$. PDF of an exponential random variable is expressed by

$$f_X(x) = \frac{e^{-(x-E[X])^2/2V[X]}}{\sqrt{2\pi V[X]^2}}, \quad (\text{C.19})$$

where $E[X]$ and $V[X]$ are the expected value and the variance of the random variable, respectively.

C.2.4 Expected Value and Variance

For a random variable X , the *expected value*, or mean, $E[X]$, is defined as the statistical average of all possible values of the random variable. Thus, for a discrete random variable X taking on values from a total of N possible values, the expected value is

$$E[X] = \sum_{\text{all values of } x} x P_X(x). \quad (\text{C.20})$$

The concept is identical for a continuous random variable having infinite points:

$$E[X] = \int_{-\infty}^{\infty} x f_X(x) dx. \quad (\text{C.21})$$

We can also define the *variance* of a random variable that gives a measure of how values differ:

$$V[X] = E[(X - E[X])^2]. \quad (\text{C.22})$$

C.2.5 A Function of Random Variable

If $g(X)$ is a function of X , the expected value of $g(X)$ for a discrete random variable can also be defined as follows:

$$E[g(X)] = \sum_{\text{all values of } x} g(x) P_X(x) \quad (\text{C.23})$$

and for a continuous random variable:

$$E[g(X)] = \int_{-\infty}^{\infty} g(x) f_X(x) dx. \quad (\text{C.24})$$

The expected value of a random variable by itself may not be useful for the numerical evaluation of the random variable. The reason is that two random variables with an identical expected value can take on values, each from a totally different range of numbers. The variance can be calculated by using either Equation C.23 or C.24, depending on the type of the random variable.

C.3 Multiple Random Variables

We often encounter several random variables that are related somehow. For example, a random signal as noise enters several circuits, and the outputs of these circuits can form *multiple random variables*. Multiple random variables are denoted by a vector $\mathbf{X} = \{X_1, X_2, \dots, X_n\}$.

C.3.1 Basic Functions of Two Random Variables

For two random variables X and Y , the *joint cumulative distribution function* denoted by $F_{X,Y}(x, y)$, the *joint probability mass function* denoted by $P_{X,Y}(x, y)$, and the *joint probability density function*, $f_{X,Y}(x, y)$ are, respectively, derived from:

$$F_{X,Y}(x, y) = P[X \leq x, Y \leq y], \quad (\text{C.25})$$

$$P_{X,Y}(x, y) = P[X = x, Y = y], \quad (\text{C.26})$$

and

$$f_{X,Y}(x, y) = \frac{\partial^2 F_{X,Y}(x, y)}{\partial x \partial y}. \quad (\text{C.27})$$

We can define the *marginal* CDF of the two random variables as

$$F_X(x) = F_{X,Y}(x, \infty).$$

Similarly, the *marginal* PMF of the two discrete random variables is

$$P_X(x) = \sum_{\text{all } y \text{s}} P_{X,Y}(x, y),$$

and the *marginal* PDF of the two continuous random variables is

$$f_X(x) = \int_{-\infty}^{\infty} f_{X,Y}(x, y) dy.$$

C.3.2 Two Independent Random Variables

Two random variables are considered independent of each other if one of the following corresponding conditions is met:

$$P_{X,Y}(x, y) = P_X(x)P_Y(y), \quad (\text{C.28})$$

$$f_{X,Y}(x, y) = f_X(x)f_Y(y), \quad (\text{C.29})$$

or

$$F_{X,Y}(x, y) = F_X(x)F_Y(y). \quad (\text{C.30})$$

C.4 Stochastic (Random) Processes

A *stochastic (random) process* is a special version of a random variable that is a function of time. When time is countable, a stochastic process is called *discrete time* and denoted by $X(n, w)$, or $X(n)$, or simply X_n , where n is time. Otherwise, a stochastic process is *continuous time* and is denoted by $X(t)$, where t is time.

C.4.1 IID Random Process

As an example of discrete-time random processes, consider the *independent and identically distributed (IID) random process*. This discrete-time random process is denoted by X_n , in which n “independent” discrete random variables have “identical” CDFs. Therefore, the n random variables X_1 , occurring at time 1, all the way to X_n , occurring at time n , make up the random process $X_n = \{X_1, X_2, \dots, X_n\}$.

C.4.2 Brownian Motion Random Process

The *Brownian motion random process*, also called *Wiener random process*, is an example of continuous-time random processes. A Brownian motion process, $X(t)$, begins at the origin, has zero expected value for all time t , but has a variance that increases linearly with time as

$$E[X(t)] = 0 \quad (\text{C.31})$$

and

$$V[X(t)] = \alpha t. \quad (\text{C.32})$$

The PDF of a Brownian motion random process can be approximated by the PDF of a Gaussian (normal) random variable presented in Equation (C.19) as

$$f_X(x) = \frac{e^{-x^2/2\alpha t}}{\sqrt{2\pi\alpha t}}, \quad (\text{C.33})$$

where αt is the variance. For any time increment δ , the increment of a Brownian motion process, $X(t + \delta) - X(t)$, has a distribution defined by

$$P[X(t + \delta) - X(t) \leq x] = \frac{1}{\sqrt{2\pi\delta}} \int_{-\infty}^x e^{-y^2/2\delta} dy, \quad (\text{C.34})$$

using a variance equal $\alpha t = \delta$. This process is used to capture the nature of batch arrival and bursty traffic (see Chapter 18).

C.5 Theory of Markov Chains

A *Markov process*, X_n , is a stochastic process in which the past state has no impact on the future state if the present state is specified. In other words, in a Markov process, any subsequent behavior of a state is independent of its past activity. We use a state machine called a *Markov chain* to express a Markov process. Therefore, a Markov chain depicts

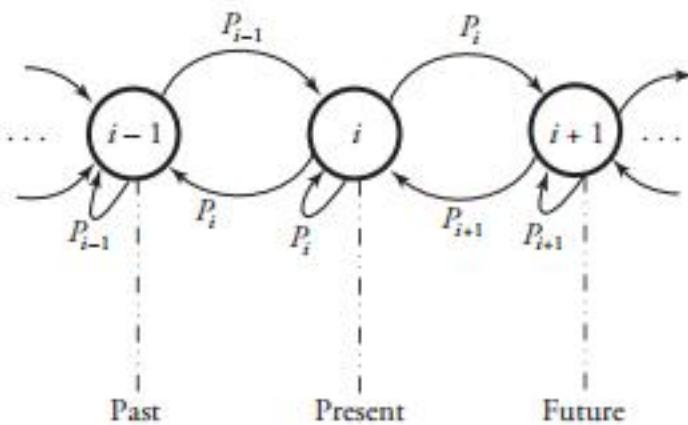


Figure C.1 A simple Markov chain

the activities of a Markov process, state by state, which is a convenient way to grasp the essence of the process. Figure C.1 shows a simple Markov chain.

A chain can start from a state 0 and move toward an ending state, if there is one. The chain in this figure shows three sample states on a Markov chain: $i - 1$, a past state; i , the present state; and $i + 1$, a future state. These three states are connected through their associated probabilities, shown in the figure. Markov chains are also classified as either *discrete time* or *continuous time*.

C.5.1 Continuous-Time Markov Chains

In a continuous-time Markov chain based on a random process, $X(t)$, the transition probabilities occur in a very short period of time, δ . Assuming $\alpha_{i,i}$ to be the rate at which the process leaves state i , the probability that the process remains in state i during δ is estimated as

$$\begin{aligned}
 P_{i,i} &= P[T_i > \delta] = e^{\alpha_{i,i}\delta} \\
 &= 1 - \frac{\alpha_{i,i}\delta}{1!} + \frac{(\alpha_{i,i}\delta)^2}{2!} - \dots \\
 &\approx 1 - \alpha_{i,i}\delta.
 \end{aligned} \tag{C.35}$$

When it leaves state i , the process enters state j with probability $\pi_{i,j}$. Thus, the probability that the process remains in state j during δ is

$$P_{i,j} = (1 - P_{i,i})\pi_{i,j}. \tag{C.36}$$

Combining Equations (C.35) and (C.36), we derive

$$P_{i,j} = \alpha_{i,i} \delta \pi_{i,j} = \alpha_{i,j} \delta, \quad (\text{C.37})$$

where $\alpha_{i,j} = \alpha_{i,i} \pi_{i,j}$ is the rate at which process $X(t)$ moves from state i to state j . If we divide Equations (C.35) and (C.37) by δ and take a limit, we get

$$\begin{cases} \lim_{\delta \rightarrow 0} \left(\frac{P_{i,i} - 1}{\delta} \right) = \alpha_{i,i} \\ \lim_{\delta \rightarrow 0} \left(\frac{P_{i,j}}{\delta} \right) = \alpha_{i,j} \end{cases}. \quad (\text{C.38})$$

To find a state j probability of the process at a given time t denoted by $P_j(t) = P[X(t) = j]$, we can elaborate further:

$$\begin{aligned} P_j(t + \delta) &= P[X(t + \delta) = j] = \sum_i P[X(t + \delta) = j | X(t) = i] P[X(t) = i] \\ &= \sum_i P_{i,j} P_i(t). \end{aligned} \quad (\text{C.39})$$

By subtracting $P_j(t)$ from both sides of this equation, dividing them by δ , taking a limit of $\delta \rightarrow 0$, and applying the Equations in (C.38), we can derive

$$P'_j(t) = \sum_{i \neq j} \alpha_{i,j} P_i(t). \quad (\text{C.40})$$

This is an important result, known as the *Chapman-Kolmogorov equation* on continuous-time Markov chains. This equation clearly deals with the differential operation of a state probability with respect to time t .

Index

- AAL (ATM adaptation layer), 34
Abort chunks, 502
ABR (Associative-Based Routing) Protocol, 521–522
Abstract syntax notation one (ASN.1) language, 242
Access points, 131
Accounting, network management for, 241
Accuracy of routing algorithms, 173
ACK frames, 140
Acknowledgment (ACK) field, 211, 213
Acknowledgment number field, 210
Acronyms, 563–568
Ad-hoc networks
 MANETs. *See* Mobile ad-hoc networks (MANETs)
 WMN support for, 165
Ad-Hoc On-Demand Distance Vector (AODV) protocol
 new nodes for, 528
 route discovery and establishment in, 524–526
 route maintenance in, 526–528
 routing process in, 523–524
Adaptability of routing algorithms, 173
Adapters for ARP, 105
Adaptive modulation, 79–80
Adaptive protocols, 514
Adaptive reservation schemes, 339
Add cyclic prefixes, 81
Additional information field, 231–232
Additive increase, multiplicative decrease congestion control, 217–219
Address autoconfiguration procedure, 161–162
Address field, 139
Address family identifier field, 185
Address mapping in DNS, 230–231
Address Resolution Protocol (ARP), 105–106
Addresses
 ATM, 38
 IP. *See* IP addresses
 in mobile IP, 156–158
 in network-layer routing, 173–174
 routing tables. *See* Routing tables
Admission control, 324
ADSL (asymmetric DSL), 54
ADUs (application data units), 491
Advanced Encryption Standard (AES) protocol, 259
Agent address field, 245
Agents
 H.323, 488–489

- Agents (*continued*)
mobile IP, 156–159
network management, 241
SIP, 484–485
- Aggregate technique in CIDR, 28
- All-optical networks, 392
- All-optical switches, 391, 395–398
- Aloha-based protocols, 83–84
- Alphabets in lossy compression, 463
- Amplifiers in optical networks, 382
- Amplitude shift keying (ASK), 52–53
- Answer field, 231
- Antennas
in cellular networks, 143–144
WiMAX, 164
for wireless links, 74–76
- Anycast addresses, 32
- AODV. *See* Ad-Hoc On-Demand Distance Vector (AODV) protocol
- Application data units (ADUs), 491
- Application layer, 225–226
DNS, 227–232
email, 235–236
exercises, 246–247
FTP, 237
network management, 239–245
overview, 226–227
remote login protocols, 232–235
summary, 245–246
in TCP/IP protocol model, 21
WWW, 237–239
- Application-specific type RTCP packets, 494
- Arbitration elements, 64
- Area ID field, 189
- ARP (Address Resolution Protocol), 105–106
- ARQ (automatic repeat request)
in channel coding, 79
in TCP, 209
- Arrival rate in multipath buffered crossbars, 374
- ASK (amplitude shift keying), 52–53
- ASN.1 (abstract syntax notation one)
language, 242
- Associative-Based Routing (ABR) Protocol, 521–522
- Associativity in ABR, 521
- Associativity ticks, 521
- Assured forwarding PHBs, 336
- Asymmetric DSL (ADSL), 54
- Asymmetric encryption, 260
- Asynchronous MAC protocols, 106–107
- Asynchronous Transfer Mode (ATM)
technology, 33–34
cell structure, 36, 38–39
M/D/1 model for, 298
protocol structure, 34–37
resource allocation in, 340–344
- ATM layer, 34
- ATM adaptation layer (AAL), 34
- Attacks
on ad-hoc networks, 529–530
categories, 251–255
- Attenuation, bit, 89
- Audio, streaming, 462–463
- Authentication, 263–264
categories, 256–257
in cellular networks, 150
digital signatures for, 265
SHA, 263–265
- Authentication field, 190
- Authentication data field, 266
- Authentication type field, 190
- Authenticity
in ad-hoc networks, 530
in security, 250
- Authoritative servers, 230
- Authority field, 231–232
- Authorization in ad-hoc networks, 530
- Automatic discovery of services, 513

- Automatic repeat request (ARQ)
in channel coding, 79
in TCP, 209
- Automatic routing table updates, 120–121
- Autonomous systems, 10
- Average queue length in RED, 199
- Await-reply packets, 522
- B (bidirectional) frames, 462
- Back-off mode, 109
- Back-off time, 135
- Back-pressure signaling, 196–197
- Backbones
Internet, 480
MBone, 413–414
- Balance equations in queueing, 289–292
- Bandwidth
ADSL, 55
TCP, 217
- Bandwidth brokers, 336
- Banyan networks, 355
- Base stations
cellular networks, 142–143
ICR, 554
sensor networks, 537
- Basic encoding rules in H.323, 486
- Batch arrival models
queuing, 298–299
self-similarity with, 503–506
- Batches of traffic units, 503
- Bellman-Ford algorithm, 178–180, 184
- Benes networks, 356–357
- Bernoulli random variables, 576
- Bernoulli trials, 574
- Best-effort models, 316, 338
- BGMP (Border Gateway Multicast Protocol), 417
- BGP (border gateway protocol), 190–191
details, 192
MBGP, 414–415
in MPLS, 438
- packets, 192–194
path-vector routing in, 191–192
- Bidirectional (B) frames, 462
- Bidirectional congestion control, 197–198
- Billing, network management for, 241
- Binary exponential backoff, 113
- Binomial coefficients, 371
- Binomial random variables, 576
- Binomial trials, 574
- Bipoint connections, 372
- Birth-and-death process
in M/M/1/b queueing systems, 289
overview, 278–279
- Bit-sliced organization, 368
- Black-and-white images, 456
- Black-hole attacks, 529
- Blocking
cellular calls, 145
in Clos networks, 360–361
link, 200–202
TDM, 47–49
- Blocking switches, 350–351, 353
Banyan networks, 355
Benes networks, 356–357
Delta networks, 355–356
Omega networks, 353–355
in optical networks, 386
- Blue-tooth communications, 72
- Bluetooth LANs, 134
- Boolean splitting multicast algorithm, 420–422
- Border Gateway Multicast Protocol (BGMP), 417
- Border gateway protocol (BGP), 190–191
details, 192
MBGP, 414–415
in MPLS, 438
packets, 192–194
path-vector routing in, 191–192
- Border routers, 9–10

- Bridges
in LANs, 58–59, 116–124
in wireless networks, 134–135
in WMNs, 164
- Broadband networks, 4
- Broadcast-and-prune algorithm, 405–406
- Broadcast translated circuits (BTCs), 420
- Broadcasts
in optical networks, 388
in star topologies, 103
in switching networks, 351
- Brownian random process, 504, 580
- Browsers, 238
- BTCs (broadcast translated circuits), 420
- Bucket depth in traffic shaping, 323
- Buffered switching networks, 351, 366–367
- Buffers
in optical networks, 382, 385
router, 60, 64–66
- Burke's theorem, 299–304
- Bursts
in ATM, 33, 342
in multimedia networking, 503–506
- Bus topology, 102
- Busy states in queuing, 280
- Busy time of CSMA channels, 112
- BYE packets, 494
- Bypass state in token-ring, 115
- Cable modems, 56–57
- Caching, Web, 238–239
- Calls in cellular networks. *See* Cellular networks
- CANCEL messages, 486
- Cantor networks, 367–368
- Capacity
in cellular networks, 146–147
in wireless channels, 79
- Carrier protocols, 435
- Carrier Sense Multiple Access (CSMA) protocol, 108–113
- Cascaded nodes, Burke's theorem for, 299–304
- Case studies
all-optical switches, 395–398
FAX compression, 470–472
multipath buffered crossbar, 368–375
sensor network simulation, 557–559
- CBR (constant bit rate), 35, 340–341
- CBT (core-based trees) protocol, 413
- CCK (complementary code keying), 137
- CDF (cumulative distribution function), 112, 575, 579
- CDMA (code-division multiple access), 85–87
- CDMA-based mobile wireless networks, 154–155
- CDNs (content distribution networks), 497–499
- Cell-delay variation (CDV), 341–342
- Cell-loss priority (CLP), 38, 343
- Cell-loss ratio, 342
- Cell sectoring, 148
- Cell splitting, 148
- Cell-transfer delay, 342
- Cells, ATM, 33, 36, 38–39, 342–344
- Cellular networks, 72, 142–143
CDMA-based, 154–155
connectivity in, 143–145
frequency reuse in, 146–149
generations of, 154
handoffs in, 149–150
mobility management in, 150–154
- Centralized access, 137–138
- Centralized clustering, 546
- Centralized protocols
for ad-hoc networks, 514
MAC, 106
- Centralized routing
in network-layer routing, 176–177
in sensor networks, 552
- CFE (contention-free end) frames, 140

CGSR (Cluster-Head Gateway Switch Routing) protocol, 517
Channel access, 82–83
 CDMA, 85–87
 FDMA, 83
 SDMA, 87
 TDMA, 83–85
Channel-usage bandwidth, 154
Channels
 ADSL, 55
 cable modem, 56–57
 in CDMA-based wireless networks, 154–155
 in cellular networks, 145–149
 coding, 79
 FDM, 44
 TDM, 47–50
 WDM, 44–45
 for wireless links, 76–79
Chapman-Kolmogorov equation, 151, 582
Checksums
 Internet, 89
 in IP packets, 24
 in routers, 67
 in TCP segments, 211
 in UDP segments, 213–214
Chip rate, 154
Choke packets, 196–197
Chunks in SCTP, 501–502
CIDR (Classless Interdomain Routing), 27–28
Ciphertext, 257
Circuit-switched networks, 4–5
Cladding, optical cable, 74
Class-based weighted fair queuing, 482
Classes, IP address, 24–25
Classifiers, packet, 325
Classless Interdomain Routing (CIDR), 27–28
Clear packets, 521
Clear to Send (CTS) frames, 140

Client/server model, 226–227
Clipping in TDM, 49–50
Clos Networks, 357–360
 blocking probability estimates for, 360–361
 five-stage, 361–362
CLP (cell-loss priority), 38, 343
Cluster-Head Gateway Switch Routing (CGSR) protocol, 517
Cluster heads, 517, 557–559
Cluster-member tables, 517
Clustering, 545
 classification of, 546
DEEP, 547–551
LEACH, 546–547
 in sensor networks, 536–537
Coaxial cable, 73
Cochannel cells, 146–147
Code-division multiple access (CDMA), 85–87
Code efficiency in compression, 467
Coding for flat fading, 79–80
Collisions
 MAC protocols for, 107–114
 in wireless environments, 84
Collocated foreign addresses, 159
Combined switching networks, 367
Command field, 185–186
Common SCTP headers, 501
Communication energy model, 540–545
Complementary code keying (CCK), 137
Complements of events, 573
Complexity of switching networks, 351
Compression, 449–450
 digital voice, 451–455
 exercises, 472–477
 FAX, 470–472
 lossless, 467–470
 lossy, 463–467
 moving images, 461–463
 overview, 450–451

- Compression (*continued*)
 still images, 455–461
 summary, 472
- Compression ratio, 467
- Concatenation of networks, 352
- Concentration switches, 361–364
- Conditional functions in probability, 575
- Conferencing, ad-hoc networks for, 513
- Confidentiality in security, 250
- Configuration, network management for, 240
- Congestion
 at network layer, 194–196
 bidirectional, 197–198
 link blocking, 200–202
 RED, 198–200
 unidirectional, 196–197
 in routers, 63–65
- TCP. *See* Transmission Control Protocol (TCP)
 in VoIP, 482
- Congestion threshold, 219
- Congestion window, 217–218
- Connection accept packets, 13
- Connection admission control, 342–343
- Connection-oriented networks, 10–11, 13–14
- Connection release packet, 13
- Connection reliability, 156
- Connection request packets, 13
- Connection rules in link blocking, 200–202
- Connection setup for TCP, 212–213
- Connectionless networks, 10–13
- Connectionless routing, 521
- Connectivity in cellular networks, 143–145
- Constant bit rate (CBR), 35, 340–341
- Content distribution networks (CDNs), 497–499
- Content providers, 498
- Contention-access MAC protocols, 107–108
- CSMA, 108–113
- IEEE 802.3 standard, 113–114
- Contention-free end (CFE) frames, 140
- Contention resolution
 in optical networks, 385–386
 in routers, 64–65
- Continuous probability functions, 575–576
- Continuous-time Markov chains, 581–582
- Continuous time stochastic processes, 579
- Contributing source count field, 493
- Contributing source identifier field, 493
- Control channels, 145
- Control chunks, 502
- Control frames, 140
- Control plane, 35–36
- Control signals, 548
- Controlled GFC, 38
- Controlled-load service classes, 316
- Controllers
 in cellular networks, 143–144
 in routers, 64–65
- Convergence
 in ATM, 34–35
 in RIP, 187
 in routing algorithms, 173
- Cookie acknowledgment chunks, 502
- Core, optical cable, 74
- Core-based trees (CBT) protocol, 413
- Core LSRs, 439–440
- Core points
 in CBT, 413
 in sparse-mode algorithm, 406
- Costs, routing, 176, 182
- Customer edge routers, 443
- Count field, 494
- Counting in probability, 574
- Couplers in optical networks, 384, 388
- CRC field
 in IEEE 802.11, 140

- in SSH Packets, 235
- CRCs. *See* Cyclic redundancy checks (CRCs)
- Cross talk, 88, 386
- Crossbar switching, 352–353
 - multipath buffered crossbars, 368–375
 - in optical networks, 387–388
- Crossovers in optical networks, 386
- Crosspoints
 - in crossbar switch fabrics, 352–353
 - in multipath buffered crossbars, 368–369
- Cryptographic techniques, 255–256. *See also* Encryption
- Cryptography, 255
- CSMA (Carrier Sense Multiple Access) protocol, 108–113
- CSMA/CA method, 135
- CTS (Clear to Send) frames, 140
- Cumulative distribution function (CDF), 112, 575, 579
- Cumulative number of packets lost field, 496
- Custom queuing, 482
- Cut-through switching, 14
- Cyclic redundancy checks (CRCs), 89–90
 - effectiveness of, 93–94
 - implementation of, 94
 - at receivers, 90–92
 - in routers, 67
 - at transmitters, 90
- Data-acquisition phase in ICR, 556
- Data frames, 140
- Data-carrying frames, 140
- Data/CFE-ACK frames, 141
- Data/CFE ACK/CFE-Poll frames, 141
- Data/CFE-Poll frames, 141
- Data Encryption Standard (DES) protocol, 257–259
- Data links, 71
 - channel access on, 82–87
- error detection and correction in, 87–94
- exercises, 99–100
- flow control, 94–98
- overview, 72–73
- summary, 98–99
- wired, 73–74
- wireless. *See* Wireless links
- Data transfer in RTP, 491–492
- Database description packets, 190
- Datagram networks, 10–13
- DCF (distributed coordination function) algorithm, 138–139
- DCT (discrete cosine transform) process, 458–459
- Decentralized clustering, 546
- Decentralized Energy-Efficient Cluster Propagation (DEEP) protocol, 547–551
- Decision-feedback equalizers (DFEs), 80–81
- Decryption
 - in AES, 259
 - in RSA, 261–262
- DEEP (Decentralized Energy-Efficient Cluster Propagation) protocol, 547–551
- Deep fading, 77–78
- Deficit round-robin (DRR) scheduler, 333–334
- Definitions in switching networks, 351–352
- Deflected packets, 351
- Deflection routing, 181, 385
- Delay. *See also* Packet queues and delay analysis
 - in connectionless networks, 12–13
 - in CSMA, 111–113
 - in multipath buffered crossbars, 374–375
 - in optical networks, 382
 - packet size in, 14–16
 - in priority queues, 330
 - queueing systems for, 292–293

- Delay (*continued*)
in VoIP, 481
Delay since last SR field, 496
Delta networks, 355–356
Delta routing, 160
Demapping in OFDM, 81
Denial-of-service attacks, 254–255
Dense-mode algorithm, 405–406
Dense-mode PIM, 410
Dependency
wavelength allocation with, 394–395
wavelength allocation without, 393–394
DES (Data Encryption Standard) protocol, 257–259
Destination address field
Ethernet LAN frames, 114
IP packets, 24
IPv6 packets, 31
Destination port field
TCP segments, 210
UDP segments, 213–214
Destination port number field, 501
Destination-Sequenced Distance Vector (DSDV) protocol, 515–517
DFEs (decision-feedback equalizers), 80–81
DHCP (Dynamic Host Configuration Protocol), 174
Differentiated services code points (DSCPs), 336
Differentiated services (DS) QoS, 335–337
Diffie-Hellman key-exchange protocol, 262
DiffServ field, 336
Diffused configuration, 132
DIFS (distributed IFS coordination function), 139
Digital modulation techniques, 52–54
Digital signatures, 265
Digital subscriber line (DSL) modems, 54–55
Digital voice compression
quantization and distortion in, 452–455
signal sampling in, 451–452
Dijkstra's algorithm, 177–178, 409–410
Direct paths for wireless links, 76
Direct routing
in mobile IP, 160
in sensor networks, 552–553
Direct-sequence spread spectrum (DSSS)
in CDMA, 86, 155
in physical layer, 136
Directed beam configuration, 132
Directed diffusion, 552
Directional antennas, 76
Directional couplers, 384
Directional transmission links, 72–73
Discarded packets, 351
Discovery
mobile IP agents, 157–158
route. *See Route discovery*
Discrete cosine transform (DCT) process, 458–459
Discrete probability functions, 575
Discrete-time Markov chains, 581
Discrete time stochastic processes, 579
Distance tables, 518
Distance vector algorithm, 522
Distance Vector Multicast Routing Protocol (DVMRP), 407
Distance vector routing, 183–185
Distortion in voice compression, 452–455
Distributed access, 137–138
Distributed coordination function (DCF)
algorithm, 138–139
Distributed denial-of-service attacks, 254
Distributed IFS coordination function (DIFS), 139
Distributed MAC protocols, 106
Distributed multimedia networking, 497–500
Distributed protocols, 514
Distributed routing
algorithms for, 176–177

- in sensor networks, 552
- Distribution networks, 364–365
- Distribution of sensor network loads, 557–558
- Diversity
 - in CDMA frequency, 86
 - for flat fading, 79–80
- DNS. *See* Domain Name System (DNS) and servers
- Domain name space, 228–229
- Domain Name System (DNS) and servers, 227–228
 - CDN interactions with, 498–499
 - domain name space in, 228–229
 - in H.323, 487
 - hacking attacks on, 251–252
 - message format, 231–232
 - name/address mapping in, 230–231
 - in SIP, 484–485
- Domains
 - highjacking attacks on, 252
 - in packet-switched networks, 10
- Doppler frequency shift, 78
- Dot-decimal notation, 25
- Downstream bandwidth, 55
- Downstream on demand schemes, 442
- Droppers in DiffServ, 335
- Dropping cellular network calls, 145
- DRR (deficit round-robin) scheduler, 333–334
- DS (differentiated services) QoS, 335–337
- DSCPs (differentiated services code points), 336
- DSDV (Destination-Sequenced Distance Vector) protocol, 515–517
- DSL (digital subscriber line) modems, 54–55
- DSR (Dynamic Source Routing) protocol, 519–520
- DSSS (direct-sequence spread spectrum)
 - in CDMA, 86, 155
 - in physical layer, 136
- Duration/connection ID (D/I) field, 139
- DVMRP (Distance Vector Multicast Routing Protocol), 407
- Dynamic Host Configuration Protocol (DHCP), 174
- Dynamic routing, 177
- Dynamic Source Routing (DSR) protocol, 519–520
- EAP (Extensible Authentication Protocol), 268
- EAR (energy-aware routing), 556–557
- Earliest deadline first (EDF) scheduler, 335
- Echo, 88, 482
- Echo cancelers, 88
- EDF (earliest deadline first) scheduler, 335
- EDPAS (erbium-doped fiber amplifiers), 382
- Egress LSRs, 439–441
- Electro-optic switches, 384
- Electronic mail (email), 235–236
- Electrooptical switches, 384
- Email (electronic mail), 235–236
- Encapsulation
 - in routers, 63
 - in VPNs, 434–435
- Encoding in compression, 460–461
- Encryption
 - in ad-hoc networks, 530
 - cryptographic techniques, 255–256
 - public-key, 260–262
 - secret-key, 257–259
 - in SSH, 234
- End-to-end encryption, 255
- End-to-end protocols. *See* Transport and end-to-end protocols
- Energy-aware routing (EAR), 556–557
- Energy-exhaustion attacks, 530
- Enterprise field, 245
- Entropy in lossy compression, 464–465

- Equal-sized packets model. *See* Asynchronous Transfer Mode (ATM) technology
- Erbium-doped fiber amplifiers (EDPAS), 382
- Erlang-B blocking probability, 293
- Erlang-C formula, 290
- Error detection and correction, 87–89
 CRC, 89–94
 in IPv6 packets, 31
 in routers, 67
- Error index field, 244
- Error status field, 244
- Ethernet LANs, 113–114
- Events in probability, 573–574
- Exclude mode in IGMP, 408
- Exp field, 439
- Expansion switches, 364–365
- Expected value, 577–578
- Expedited forwarding PHBs, 336
- Explicit routing, 441–442
- Exponential random variables, 577
- Express multicast, 417
- Extended highest sequence number received field, 496
- Extensible Authentication Protocol (EAP), 268
- Extension (X) field, 492
- Extension headers in IPv6, 32
- Extra data in TCP, 222
- Extranet VPNs, 433
- Fading, 77–80
- Failures, network management for, 240
- Fair-queueing (FQ) scheduler, 331–332
- Fairness index, 340
- Fast Fourier transform (FFT), 81–82
- Fast retransmit method, 216, 220–221
- FAX compression, 470–472
- FCFS (first come, first served) queuing systems, 280
- FDM (frequency-division multiplexing), 44
- FDMA (frequency-division multiple access), 83
- FEC (forward equivalence class), 439, 441–442
- FEC (forward error correction), 79
- Feedback models, 304–305
- FFT (fast Fourier transform), 81–82
- FHSS (frequency-hopping spread spectrum), 136
- Fiber-optic communications, 72
- FIFO (first-in, first-out) queuing systems.
 See Markovian FIFO queueing systems
- FIFO (first-in, first-out) schedulers, 326–327
- File Transfer Protocol (FTP), 237
- Filters
 in optical networks, 382–383, 388
 packet, 270
- Finished (FIN) field, 211, 213
- Firewalls
 operation of, 269–270
 with VPNs, 436–437
- First come, first served (FCFS) queuing systems, 280
- First-in, first-out (FIFO) queuing systems.
 See Markovian FIFO queueing systems
- First-in, first-out (FIFO) schedulers, 326–327
- 5-layer TCP/IP protocol model, 20–22
- Five-stage Clos networks, 361–362
- Fixed reservation schemes, 339
- Fixed routing, 120
- Fixed-size switch elements, 395–396
- Flag field, 501
- Flags field
 DNS, 231
 IP packets, 24, 29

- Flags/code field, 159
Flat fading, 77–80
Flood attacks, 254–255
Flood routing, 180–181
Flow-based switches, 59
Flow control, 94–95
 - sliding-window, 96–98
 - stop-and-wait, 95–96
 - in switching networks, 351
Flow label field, 31
Foreign addresses in mobile IP, 157, 159
Foreign agents in mobile IP, 157
Foreign networks in mobile IP, 156–157
Forward equivalence class (FEC), 439, 441–442
Forward error correction (FEC), 79
Forward links, 154
FQ (fair-queueing) scheduler, 331–332
Fraction loss field, 495
Fragment offset field, 24
Fragmentation
 - packet, 6, 28–29, 33
 - in routers, 60
Frame body field, 140
Frame check sequence, 67
Frame check sequence field, 114
Frame control (FC) field, 139
Frame delay analysis, 110
Frame-switch mode, 103
Frames
 - LAN, 102
 - MAC, 139–141
 - MPEG, 461–462
 - packet-switched networks, 5–6
Free-memory lists, 62
Frequency borrowing, 148
Frequency-division multiple access (FDMA), 83
Frequency-division multiplexing (FDM), 44
Frequency hopping, 85

Frequency-hopping spread spectrum (FHSS), 136
Frequency ranges, 72
Frequency reuse, 146–149
Frequency shift, 78
Frequency shift keying (FSK), 52–53
FTP (File Transfer Protocol), 237
Full-duplex links, 73
Functions
 - probability, 575–576
 - random variable, 578–579
Gatekeepers, 487–489
Gateway/bridges
 - wireless routers with, 141
 - in WMNs, 164
Gateways
 - H.323, 487
 - LAN, 116
Gaussian noise, 88
Gaussian (normal) random variables, 577
General distributions, 295–296
Generating polynomials for CRC, 92
Generator values for CRC, 92–93
Generators, checking, 90–91
Generic flow control (GFC) field, 36, 38
Generic routing encapsulation (GRE), 434
Geometric distributions, 285
Geometric random variables, 576
Geosynchronous orbit satellite systems, 131
Get PDUs, 244
GFC (generic flow control) field, 36, 38
GFR (guaranteed frame rate) service, 343
GIF (graphics interchange format) file
 - compression, 457
Global packet resequencers, 66
Global positioning systems (GPSs), 539, 551
Glossary of acronyms, 563–568
Goodbye packets, 494

- GPSs (global positioning systems), 539, 551
Grant flows, 370
Graphics interchange format (GIF) file compression, 457
GRE (generic routing encapsulation), 434
Guaranteed frame rate (GFR) service, 343
Guaranteed service classes, 316
Guard space, 200
Guided missiles, 72
Guided transmission links, 72–73
- H.323 protocols, 486–490
Hacking attacks, 251–252
Half-duplex links, 73
Handoffs in cellular networks, 145, 149–150
Hardware firewalls, 269
Hardware security, 163
Hash functions, 263
HDSL (high-bit-rate digital subscriber line), 55
HDTV (high-definition television), 461
Header decoders, 64
Header error control (HEC) field, 38–39
Header length (HL) field
 IP packets, 23
 TCP segments, 210
Headers
 ATM, 33, 36, 38
 IP, 23
 IPsec, 266
 IPv6, 31–32
 LAN, 102
 MPLS, 438–439
 OSPF, 189
 packet-switched networks, 5–6
 RIP, 185–187
 RTCP, 494–496
 RTP, 492–493
 SCTP, 501–502
- Heartbeat acknowledgment chunks, 502
Heartbeat request chunks, 502
Heavy-tailed distributions, 503, 505–506
HEC (header error control) field, 38–39
HELLO messages, 527–528
Hello packets, 190
HFC (hybrid fiber-coaxial) networks, 56
Hidden-terminal problem, 84
Hierarchical wireless networks, 131
High-bit-rate digital subscriber line (HDSL), 55
High-definition television (HDTV), 461
Higher layers in ATM, 34
Highjacking attacks, 252
Hold state in RIP, 187
Home addresses in mobile IP, 156–157
Home agent field, 159
Home agents, 156–157, 159
Home networks, 513
Home RF LANs, 134
Hop limit field, 31
Host based resource allocation, 339
Host IDs
 in IP addresses, 24
 in subnet addressing, 26
Hosts
 address assignments for, 173–174
 in intradomain routing protocols, 182
 ISP, 8
Hot-potato routing, 181
Hotspots in WiFi, 142
HTTP (Hyper Text Transfer Protocol), 238–239
Hubs
 in LANs, 116–124
 purpose, 57–58
 in star topologies, 103
Huffman encoding, 468–469
Hurst parameter, 504
Hybrid fiber-coaxial (HFC) networks, 56
Hybrid multiple-access techniques, 87

- Hyper Text Transfer Protocol (HTTP), 238–239
- I (interactive) frames, 462
- I-TCP (Indirect Transmission Control Protocol), 215–216
- ICANN (Internet Corporation for Assigned Names and Numbers), 174
- ICMP (Internet Control Message Protocol), 29–30
- ICR (Intercluster Energy-Conscious Routing), 554–557
- Identification field
 DNS, 231
 IP packets, 24
 mobile IP, 159
- Idle states in queuing, 280
- IEEE 802.3 standard, 113–114
- IEEE 802.5 standard, 116
- IEEE 802.11 standard, 134–136
 MAC layer, 137–141
 physical layer, 136–137
 security for, 267–268
 for WiFi technology, 141–142
- IEEE 802.15.4 standard, 559–560
- IEEE 802.16 standard, 163–164
- IETF (Internet Engineering Task Force), 266
- IFS (interframe space) technique, 138–139
- IGMP (Internet Group Management Protocol), 407–409
- IID (independent and identically distributed) processes, 280, 580
- In-phase QAM components, 54
- Include mode in IGMP, 408
- Incoming label map tables, 440
- Independent and identically distributed (IID) processes, 280, 580
- Independent events, 574
- Independent random variables, 579
- Indirect Transmission Control Protocol (I-TCP), 215–216
- Information-compression process, 451
- Information leakage attacks, 252
- Information-level attacks, 252
- Information-source process, 450–451
- Information theory for lossy compression, 463–464
- Infrared frequency spectrum, 72
- Infrared LANs, 132–133
- Ingress LSRs, 439–440
- Initial sequence numbers (ISNs), 210
- Initiation chunks, 502
- Initiation acknowledgment chunks, 502
- Input port buffers, 370
- Input port processors (IPPs), 60–63, 423
- Insertion loss, 383, 386
- Integrated layer processing, 491
- Integrated services QoS, 316–317
 admission control in, 324
 packet scheduling in, 325–335
 RSVP, 324–325
 traffic shaping in, 317–324
- Integrity
 in ad-hoc networks, 530
 in security, 250
- Interactive (I) frames, 462
- Interarrival jitter field, 496
- Intercluster Energy-Conscious Routing (ICR), 554–557
- Intercluster routing protocols, 551, 554–557
- Interdomain routing protocols, 190
 BGP, 190–194
 multicast, 414
 BGMP, 417
 MBGP, 414–415
 MSDP, 415–417
- Interest signals, 554
- Interference, 36–38, 78–79
- Interframe space (IFS) technique, 138–139

- Interleaving, 79–80
Internet, 6–9
Internet access devices, 50–57
Internet backbones, 480
Internet checksums, 89
Internet Control Message Protocol (ICMP), 29–30
Internet Corporation for Assigned Names and Numbers (ICANN), 174
Internet Engineering Task Force (IETF), 266
Internet Group Management Protocol (IGMP), 407–409
Internet Protocol (IP) layer, 23
addressing in. *See* IP addresses
CIDR, 27–28
ICMP, 29–30
IPv6, 30–33
mobile. *See* Mobile IP
packet fragmentation and reassembly, 28–29
packets, 23–24
security, 266–267
subnet addressing and masking, 25–27
telephony. *See* Voice over IP (VoIP)
Internet service providers (ISPs), 7–9
Internetwork components, 9
Interoperability in mobile IP, 156
Interruption attacks, 253
Intersection of events, 573
Intersymbol interference (ISI), 80–81
Intracluster communication, 551
Intracluster routing protocols, 552–554
Intradomain protocols, 182–183
multicast, 406–407
CBT, 413
DVMRP, 407
IGMP, 407–409
MBone, 413–414
MOSPF, 409–410
PIM, 410–413
OSPF, 187–190
RIP, 183–187
Intranet VPNs, 433
INVITE messages, 486
IP. *See* Internet Protocol (IP) layer
IP addresses, 23–24
LAN, 104–106
multicast, 403–404
NAT for, 174–176
RIP, 185
RTP, 491
IP Security (IPsec) protocol, 267–268
IP telephone administrative domains (ITADs), 486
IP version 6 (IPv6), 30–31
addressing format in, 32
extension headers in, 32
mobile IP routing with, 161–163
packet fragmentation in, 33
IPPs (input port processors), 60–63, 423
IPsec (IP Security) protocol, 267–268
IS-95 reverse links, 155
ISI (intersymbol interference), 80–81
ISNs (initial sequence numbers), 210
Isotropic antennas, 75, 87
ISPs (Internet service providers), 7–9
ITADs (IP telephone administrative domains), 486
Iterative mapping, 230–231
Jackets, optical cable, 74
Jackson's theorem, 304–308
Jitter
description, 88–89
in RTCP, 496–497
in VoIP, 481–482
Join messages, 411
Joint cumulative density function, 578
Joint entropy, 465
Joint Photographic Experts Group (JPEG) compression, 455–461

- Joint probability functions, 578
- Keep-alive messages, 192
- Keep-alive packets, 193–194
- Kendal's notations, 279–281
- Keys
- in ad-hoc networks, 529
 - public-key encryption, 256, 260–262
 - secret-key encryption, 256–259
- Knockout switching networks, 363–364
- L2F (Layer 2 Forwarding) protocol, 433
- L2TP (Layer 2 Tunneling Protocol), 433–434
- Label Distribution Protocol (LDP), 440, 442
- Label switch paths (LSPs), 439
- Label switch routers (LSRs), 438–441
- Label value field, 439
- Labels
- for domain names, 228–229
 - in MPLS, 438
- LANs. *See* Local area networks (LANs)
- Large-scale optical switches, 386–388
- Lasers, 72, 382
- Last come, first served (LCFS) queuing systems, 280
- Last SR timestamp field, 496
- Latency. *See* Delay
- Law enforcement, ad-hoc networks for, 513
- Layer 2 and 3 switches, 124–125
- Layer 2 Forwarding (L2F) protocol, 433
- Layer 2 Tunneling Protocol (L2TP), 433–434
- LBSs (local base stations), 554
- LCFS (last come, first served) queuing systems, 280
- LDP (Label Distribution Protocol), 440, 442
- LEACH (Low-Energy Adaptive Clustering Hierarchy) protocol, 546–547
- Leaky-bucket traffic shaping, 318–324
- Least-cost-path algorithms
- Bellman-Ford, 178–180
 - Dijkstra's, 177–178
- Least-cost paths, 176
- Leave group messages, 408
- Lee's method, 200–201
- Lempel-Ziv encoding, 469–470
- Length field
- RTCP packets, 494
 - SCTP packets, 501
 - SSH Packets, 235
- Length/Type field, 114
- Lifetime field, 159
- Light frequency spectrum, 72
- Light networks. *See* Optical networks
- Lightpaths, 380
- Line cards, 60
- Line coding methods, 50–52
- Link-cost tables, 518
- Link costs, 176
- Link layer, 20
- Link reversal, 521
- Link-state acknowledgment packets, 190
- Link-state multicast, 409
- Link-state request packets, 190
- Link-state routing, 188
- Link-state update packets, 190
- Link utilization, 14–15
- Links
- attacks on, 252–253
 - blocking, 200–202
 - data. *See* Data links
 - encrypting, 256
 - ISP, 8
 - in mobile wireless networks, 154–155
 - in optical networks, 380
 - wireless. *See* Wireless links
- Listen state in token-ring, 115
- Little's theorem, 276–278
- LLC (logical-link layer), 103–104

- LLC data field, 114
Load balancing, 173
Load distribution, 557–558
Local area networks (LANs), 9, 101–102
 exercises, 126–128
 IP addresses for, 104–106
 MAC protocol for, 106
 classification, 106–107
 contention-access, 107–114
 round-robin-access, 114–116
 networks of, 116–125
 protocol design for, 103–104
 repeaters, hubs, and bridges for, 116–124
 summary, 125–126
 switches for, 124–125
 topologies, 102–103
 wireless, 131–134
Local base stations (LBSs), 554
Local handoffs, 149–150
Local Internet service providers, 8–9
Location-disclosure attacks, 529
Location management in cellular networks, 150
Location servers in SIP, 484–485
Logical congestion, 194
Logical-link layer (LLC), 103–104
Logical links, 208
Logins, remote, 232–235
Lossless compression, 450, 467–468
 Huffman encoding, 468–469
 Lempel-Ziv encoding, 469–470
 run-length encoding, 468
Lossy compression, 450, 463
 compression ratio and code efficiency in, 467
 entropy in, 464–465
 information theory for, 463–464
 Shannon's coding theorem in, 465–467
Low-earth orbit satellite systems, 130
Low-Energy Adaptive Clustering Hierarchy (LEACH) protocol, 546–547
Low-frequency spectrum, 72
LSPs (label switch paths), 439
LSRs (label switch routers), 438–441
M/D/1 queueing systems, 298
M/G/1 queueing systems, 295–298
M/M/1 queueing systems, 281
 mean delay and queue length in, 285
 number of packets in, 283–285
 packet arrival and service model in, 281–283
M/M/1/b queueing systems, 286
 balance equations in, 289–292
M/M/a, 287–288
 mean number of packets in, 287
M/M/8 queueing systems, 293–295
M/M/a queueing systems, 287–288
M/M/a/a queueing systems, 292–293
MAC (medium access control) protocols
 classification of, 106–107
 contention-access, 107–108
 CSMA, 108–113
 IEEE 802.3 standard, 113–114
 for LANs, 104
 round-robin-access, 114–116
 in sensor networks, 538
MAC addresses, 104–106
MAC frames, 139–141
MAC layers
 IEEE 802.11, 137–141
 in WMNs, 167
Main router buffers, 66
Manageability in multicasting, 402
Managed devices, 241
Management frames, 140
Management information base (MIB), 242–243
Management plane, 35–36
Managers, network management, 241
Manchester encoding method, 52

- MANETs. *See* Mobile ad-hoc networks (MANETs)
- Mapping
name/address, 230–231
in OFDM, 81
- Marginal CDF, 579
- Marginal PDF, 579
- Marginal PMF, 579
- Marker field, 493
- Markers in DiffServ, 335
- Markov chains
in birth-and-death process, 278–279
for multipath buffered crossbars, 371–374
in probability, 580–582
- Markovian FIFO queueing systems, 281
M/M/1, 281–285
M/M/1/b, 286–292
M/M/8, 293–295
M/M/a/a, 292–293
- MASC (Multicast Address-Set Claim) protocol, 417
- Masking, 25–27
- Masquerading attacks, 252
- Max response time field, 408
- Maximum segment size (MSS) option, 211
- Maximum transmission units (MTUs)
in IP, 28–29
in IPv6, 33
- Maximum window size, 217–218
- MBCs (multipath buffered crossbars), 368–369
Markov chain model for, 371–374
queueing model for, 369–370
throughput and delay in, 374–375
- MBGP (multiprotocol BGP), 414–415
- MBone (multicast backbone), 407, 413–414
- MCUs (multipoint control units), 487
- MD5 hash algorithm, 263
- Mean delay
in M/M/1 queueing systems, 285
in M/M/1/b queueing systems, 290
- Mean number of packets in queueing systems, 287
- Mechanical optical switches, 384
- Media sessions in SIP, 486
- Medium access control. *See* MAC (medium access control) protocols
- Medium orbit satellite systems, 130
- Membership-search signals, 548, 550
- Memory control in switch fabrics, 366
- Memory units in sensor networks, 539–540
- Mesh networks, wireless, 163
applications, 164–166
physical and MAC layers in, 167
WiMAX technology for, 163–164
- Mesh users, 164
- Message authentication, 256–257, 263
- Message transmission-list tables, 518
- Messages in packet-switched networks, 5–6
- Meters in DiffServ, 335
- Metric field, 185–186
- MF (more-fragment) bit, 29
- MIB (management information base), 242–243
- Microcells, 148
- Microwave frequency spectrum, 72
- Microwave systems, 72
- Middle-man attacks, 252
- Military, ad-hoc networks for, 513
- MIMO (multiple-input multiple-output) systems, 167
- Minimum number of hops, 176
- Mirror of networks, 352
- Misrouting attacks, 254
- Mixer relays, 492
- Mobile ad-hoc networks (MANETs), 511
exercises, 531–534
overview, 512
protocols for, 515

- Mobile ad-hoc networks (MANETs) (*continued*)
protocols for (*continued*)
 ABR, 521–522
 AODV, 522–528
 CGSR, 517
 DSDV, 515–517
 DSR, 519–520
 TORA, 520–521
 WRP, 517–519
routing in, 513–515
security of, 528–531
summary, 531
- Mobile IP, 155–156
 addresses and agents in, 156–158
 DHCP for, 174
 registration in, 158–159
 routing for, 159–163
 security in, 163
- Mobile switching centers (MSCs)
 in cellular networks, 143–144, 150
 in I-TCP, 215
 in mobile IP, 156–157
- Mobile transport protocols
 TCP, 215–216
 UDP, 216–217
- Mobile unit setup, 144
- Mobility management, 150–154
- Modems, 50
 cable, 56–57
 digital modulation techniques, 52–54
 DSL, 54–55
 line coding methods, 50–52
- Modification attacks, 253–254
- Modulation techniques, 52–54
- Modulo-2 arithmetic, 90–91
- Monochrome images, 456
- More-fragment (MF) bit, 29
- MOSPF (multicast OSPF) protocol, 409–410
- Moving images compression, 461–463
- Moving Pictures Expert Group (MPEG)
 compression, 461–463
- MP3 technology, 462–463
- MPLS. *See* Multiprotocol label switching (MPLS)
- MSCs (mobile switching centers)
 in cellular networks, 143–144, 150
 in I-TCP, 215
 in mobile IP, 156–157
- MSDP (Multicast Source Discovery Protocol), 415–417
- MSS (maximum segment size) option, 211
- MTUs (maximum transmission units)
 in IP, 28–29
 in IPv6, 33
- Multi-hop communication efficiency, 542–545
- Multicarrier modulation, 81
- Multicast Address-Set Claim (MASC) protocol, 417
- Multicast addresses, 32
- Multicast backbone (MBone), 413–414
- Multicast OSPF (MOSPF) protocol, 409–410
- Multicast Source Discovery Protocol (MSDP), 415–417
- Multicast tables, 423
- Multicasting techniques and protocols, 401–402
 definitions and techniques, 402–403
 exercises, 427–430
 interdomain, 414–417
 intradomain, 406–407
 CBT, 413
 DVMRP, 407
 IGMP, 407–409
 MBone, 413–414
 MOSPF, 409–410
 PIM, 410–413
 IP addresses, 403–404
 node-level

- Boolean splitting, 420–422
packet recirculation, 423–424
three-dimensional switches, 424–426
tree-based, 418–420
in routers, 63
summary, 426–427
switching networks, 351
tree algorithms, 404–406
- Multichannel multitransceiver MAC, 167
- Multichannel single-transceiver MAC, 167
- Multiphop routing, 552–553
- Multimedia networking
compression in. *See* Compression
distributed, 497–500
exercises, 507–509
real-time media transport protocols,
490–497
SCTP, 500–502
self-similarity and non-Markovian
streaming analysis in, 503–506
summary, 506–507
Voice over IP. *See* Voice over IP (VoIP)
- Multimedia terminals, 487
- Multipath buffered crossbars (MBCs),
368–369
Markov chain model for, 371–374
queueing model for, 369–370
throughput and delay in, 374–375
- Multipath effects, 86
- Multipath switching networks, 350
- Multiple access, 82–83
CDMA, 85–87
FDMA, 83
hybrid techniques, 87
TDMA, 83–85
- Multiple-input multiple-output (MIMO)
systems, 167
- Multiple random variables, 578–579
- Multiplexers, 43–44
FDM, 44
in packet-switched networks, 9
- TDM, 45–50
WDM, 44–45
- Multipoint control units (MCUs), 487
- Multiprotocol BGP (MBGP), 414–415
- Multiprotocol label switching (MPLS),
437–438
operation of, 438–439
routing in, 439–441
traffic engineering in, 442–443
tunneling in, 441–442
for VPNs, 443–444
- N-stage planar architecture, 388
- Name/address mapping, 230–231
- NAPs (network access points), 7
- Narrowband RF LANs, 133
- NAT (network address translation),
174–176
- NAT translation tables, 175
- National Internet service providers, 7–8
- Natural nonreturn-to-zero (NRZ) line
coding, 51–52
- Network access points (NAPs), 7
- Network-access servers, 433
- Network address translation (NAT),
174–176
- Network communication protocols, 19
ATM, 33–34
cell structure, 36, 38–39
protocol structure, 34–37
exercises, 39
Internet. *See* Internet Protocol (IP) layer
OSI, 22
summary, 39
TCP/IP, 20–22
- Network IDs, 24
- Network interface cards (NICs), 164
- Network latency, 481
- Network layer
congestion control at, 194–196
bidirectional, 197–198

- Network layer (*continued*)
congestion control at (*continued*)
link blocking, 200–202
RED, 198–200
unidirectional, 196–197
routing at, 172–173
address assignments in, 173–174
classification of algorithms for,
176–177
NAT for, 174–176
route costs in, 176
in TCP/IP protocol model, 20, 23
transport layer interaction with, 209
- Network-layer readability field, 194
- Network management, 239–241
elements of, 241
MIB, 242–243
SMI, 241
SNMP, 243–245
- Network-node interface (NNI), 36–38
- Network security. *See* Security
- Network-to-network interface (NNI), 381
- Network topology, 6, 102–103
- Network virtual terminals (NVTs), 233
- Networking devices, 43
exercises, 67–69
modems, 50–57
multiplexers, 43–50
routers, 60–67
summary, 67
switching and routing, 57–59
- Networks
cellular. *See* Cellular networks
LANs. *See* Local area networks (LANs)
mobile ad-hoc. *See* Mobile ad-hoc
networks (MANETs)
optical. *See* Optical networks
packet-switched. *See* Packet-switched
networks
of queues, 299–308
sensor. *See* Sensor networks
- switching. *See* Switch fabrics
wireless. *See* Wireless networks
- Next header field
in IPsec, 266
in IPv6 packets, 31
- NICs (network interface cards), 164
- NNI (network-node interface), 36–38
- NNI (network-to-network interface), 381
- Node-level multicast algorithms
Boolean splitting, 420–422
packet recirculation, 423–424
three-dimensional switches, 424–426
tree-based, 418–420
- Nodes
ad-hoc networks, 528–530
cascaded, 299–304
optical networks, 380, 388–391
packet-switched networks, 5
sensor networks, 538–540
- Noise, 88
- Non-least-cost-path routing, 176–177,
180–181
- Non-Markovian models
arrival, 298
queuing, 295–298
streaming analysis, 503–506
- Non-Poisson models, 298
arrival, 298
queuing, 295
- Non-preemptive priority queues, 327–329
- Non-real-time packets, 326
- Nonblocking switch fabrics, 350–351,
357–360
blocking probability estimates for,
360–361
five-stage Clos networks, 361–362
- Non-electro-optical switches, 384
- Nonpersistent CSMA, 108
- Nonreturn-to-zero (NRZ) line coding,
51–52
- Normal (Gaussian) random variables, 577

- Normalized method, 222
Notification packets, 193–194
NRZ (natural nonreturn-to-zero) line coding, 51–52
NRZ-inverted line coding, 51–52
NTP timestamp field, 495
Number of additional records field, 231
Number of answers field, 231
Number of authoritative records field, 231
Number of questions field, 231
NVTs (network virtual terminals), 233
Nyquist theorem, 451
- OFDM (orthogonal frequency division multiplexing), 81–82, 136–137
OLSR (Optimized Link State Routing) protocol, 142
Omega networks, 353–355
Omnidirectional configuration, 132
1-persistent CSMA, 108, 113
Ongoing calls, 145
Opaque optical switches, 391
Open networks, Jackson's theorem on, 305–308
Open packets in BGP, 192–194
Open Shortest Path First (OSPF) protocol MOSPF, 409–410 operation of, 187–190
Open systems interconnection (OSI) model, 22
Operation error chunks, 502
OPPs (output port processors), 65–67, 423
Optical cross-connect (OXC) switches, 384
Optical fiber, 73
Optical networks, 379–380 amplifiers in, 382 contention resolution in, 385–386 delay elements in, 382 exercises, 399–400 filters in, 382–383, 388 overview, 380
protocol models and standards for, 380–382 routers in, 388–391 summary, 398–399 switches in, 384–388, 395–398 tunable lasers for, 382 wavelength allocation in, 391–395 WDMs in, 383
Optimal quantizers, 455
Optimality packet size, 14–16 routing algorithm, 173
Optimized Link State Routing (OLSR) protocol, 142
Options field IP packets, 23–24 TCP segments, 211–212
OPTIONS messages, 486
OQPSK (orthogonal QPSK), 155
Originated calls, 144
Orthogonal frequency division multiplexing (OFDM), 81–82, 136–137
Orthogonal QPSK (OQPSK), 155
OSI (Open systems interconnection) model, 22
OSPF (Open Shortest Path First) protocol MOSPF, 409–410 operation of, 187–190
Outcomes in probability, 573
Output port buffers, 370
Output port processors (OPPs), 65–67, 423
Overlay models, 381
Overlay networks, 444–446
OXC (optical cross-connect) switches, 384
P (predictive) frames, 462
P-NNI (private NNI), 36
P-persistent CSMA, 108
P2P (peer-to-peer) protocols, 445–446
Packet-by-packet switches, 59

- Packet-drop probability, 198
Packet length field, 189
Packet loss, 482
Packet-mis-treatment attacks, 253–254
Packet queues and delay analysis, 275–276
 birth-and-death process, 278–279
 in connectionless networks, 12
 exercises, 309–314
Little's theorem, 276–278
Markovian FIFO queueing systems, 281
 M/M/1, 281–285
 M/M/1/b, 286–292
 M/M/8, 293–295
 M/M/a/a, 292–293
networks of queues, 299
 Burke's theorem, 299–304
 Jackson's theorem, 304–308
non-Markovian models, 295–298
queueing disciplines, 279–281
self-similarity and batch-arrival models, 298–299
summary, 308
Packet recirculation multicast algorithm, 423–424
Packet-reservation multiple-access (PRMA) scheme, 84–85
Packet scheduling, 325–326
 deficit round-robin, 333–334
 earliest deadline first, 335
 fair queueing, 331–332
 first in, first out, 326–327
 priority queueing, 327–331
 weighted fair queueing, 332–333
Packet-switched networks, 3
 connection-oriented, 13–14
 connectionless, 10–13
 exercises, 17–18
Internet, 6–9
ISPs and internetwork components, 9–10
messages, packets, and frames in, 5–6
packet size in, 14–16
packet switching vs. circuit switching, 4–5
summary, 16
Packet type field, 494
Packets, 4
 BGP, 192–194
 encapsulation, 63
 filtering, 270
 fragmentation and reassembly, 6, 28–29, 33, 60
 IGMP, 408–409
 IP, 23–24
 LANs, 102
 MPLS, 438–439
 OSPF, 189–190
 packet-switched networks, 5–6
 RIP, 185–187
 RTCP, 494–496
 RTP, 492–493
 SCTP, 501–502
 size, 14–16
 SSH, 235
 TCP, 210–212
 TORA, 521
Pad field, 114
Pad length field, 266
Padding field
 IP packets, 24
 IPsec headers, 266
 RTCP packets, 494
 RTP packets, 492
 SSH Packets, 235
Paging channels, 155
Paging in cellular networks, 144
PAM (pulse amplitude modulation), 451
Parallel connections
 in link blocking, 200–202
 in switching networks, 352
Parallel nodes, Burke's theorem on, 302–304
Parallel-plane switching networks, 367–368

- Parallel-to-serial multiplexing, 65
Pareto distributions, 505–506
Parity bits, 89
Parity check methods, 89
Path attributes field, 194
Path-discovery, 527
Path loss
 in optical networks, 386
 in wireless links, 76–77
Path-vector routing, 191–192
Payload data chunks, 502
Payload data field, 266
Payload length field, 31
Payload type field
 ATM cells, 38
 RTP packets, 493
Payloads in ATM cells, 36, 38
PCF (point-coordination function), 139
PCM (pulse code modulation), 480
PCR (peak cell rate), 342
PDF (probability density function), 112, 575, 579
PDUs (protocol data units), 243–245, 501
Peak cell rate (PCR), 342
Peer-to-peer networks
 optical, 381
 wireless, 131
 WMNs for, 165
Peer-to-peer (P2P) protocols, 445–446
Per hop behaviors (PHBs), 336–337
Performance
 network management for, 240
 switch fabrics, 366–368
Period data in ICR, 554–555
Permutations
 in AES, 259
 in switching networks, 352
Phase shift keying (PSK), 52–54
PHBs (per hop behaviors), 336–337
Photographs, 456
Photonic switches, 391
Physical congestion, 194
Physical layer
 ATM, 34
 IEEE 802.11, 136–137
 TCP/IP protocol model, 20
 WMNs, 167
Physical medium in ATM, 34
Pictures, 456
PIFS (point IFS coordination function), 139
Pilot channels, 154
PIM (protocol-independent multicast), 410–413, 416
Pin attacks, 529
Ping of death, 254
Pixels, 470
Plug-and-play protocols, 174
PMF (probability mass function), 575, 579
Point-coordination function (PCF), 139
Point IFS coordination function (PIFS), 139
Point-to-point connections, 372
Point-to-Point Protocol (PPP), 434
Point-to-Point Tunneling Protocol (PPTP), 433, 435
Poisoned-reverse rule, 185
Poisoning attacks, 252–253
Poisson distribution, 111–113
Poisson random variables, 576–577
Polar NRZ line coding, 51
Policing
 in cell scheduling, 343
 traffic, 196–197, 318
Pollaczek-Khinchin formula, 295–298
Polling feature, 139
Polynomial CRC interpretation, 92
Port forwarding, 234–235
Port numbers
 NAT, 175
 RTP, 491
Power ratio, 340

- Power save-poll (PS-Poll) frames, 140
Power supplies for ad-hoc networks, 528
PPM (pulse position modulation), 136, 451
PPP (Point-to-Point Protocol), 434
PPTP (Point-to-Point Tunneling Protocol), 433, 435
PQ (priority queueing) schedulers, 327
Preamble field, 113
Predictive (P) frames, 462
Preemption queuing systems, 280
Preemptive priority queues, 329–331
Prefixes in CIDR, 27
Presentation layer, 22
Priority queueing, 280
non-preemptive, 327–329
preemptive, 329–331
in VoIP, 482
Priority queueing (PQ) schedulers, 327
Privacy in CDMA, 86
Private NNI (P-NNI), 36
PRMA (packet-reservation multiple-access) scheme, 84–85
Proactive distributed routing, 552
Probability, 573–574
blocking, 293, 360–361
expected value and variance in, 577–578
Markov chains in, 580–582
random variables in, 574–577
stochastic processes, 579–580
TDM blocking, 47–48
TDM clipping, 49–50
Probability density function (PDF), 112, 575, 579
Probability mass function (PMF), 575, 579
Processing units, 539–540
Protocol converters, 116
Protocol data units (PDUs), 243–245, 501
Protocol field, 24
Protocol immunization, 530
Protocol-independent multicast (PIM), 410–413, 416
Protocols
ad-hoc networks, 515–528
LANs, 103–104
network management, 241
optical networks, 380–382
sensor networks, 537–538
VoIP, 482–490
Providing routers, 443
Proxy servers
SIP, 484–485
Web, 238–239
Prune messages, 411
Pseudonoise signals, 85
PSK (phase shift keying), 52–54
Public-key encryption, 256
protocols, 260–262
in SSH, 234
Pulse amplitude modulation (PAM), 451
Pulse code modulation (PCM), 480
Pulse position modulation (PPM), 136, 451
Pulse stuffing, 46
Pulse-type plots, 220
Pulse width modulation (PWM), 451
Push (PSH) field, 211
QAM (quadrature amplitude modulation), 52–55, 81
QoS. *See* Quality of service (QoS)
QPSK (quadrature phase shift keying), 56, 81
Quadrature amplitude modulation (QAM), 52–55, 81
Quadrature-phase components, 54
Quadrature phase shift keying (QPSK), 56, 81
Quality of service (QoS), 315–316
differentiated services, 335–337
exercises, 344–348

- integrated services, 316–317
 - admission control in, 324
 - packet scheduling in, 325–335
 - RSVP, 324–325
 - traffic shaping in, 317–324
- network management for, 240
- overview, 316
- resource allocation, 337–338
 - ATM networks, 340–344
 - classification of, 338–339
 - fairness in, 340
 - management in, 338
- for streaming, 499–500
- summary, 344
- VoIP, 481–482
- WMNs, 166
- Quantization in compression
 - images, 459–460
 - voice, 452–455
- Query messages, 231
- Query packets
 - ABR, 522
 - TORA, 521
- Question field, 231–232
- Queue sets, 370
- Queues
 - delay-sensitive traffic, 292–293
 - multipath buffered crossbars, 369–370, 374
- packet. *See* Packet queues and delay analysis
- priority, 327–331
- RED, 199
- routers, 62
- VoIP, 482
- Radar, 72
- Radio frequency (RF) spectrum, 72
- Radio link security, 163
- Radio systems, 72
- RADIUS protocol, 268
- Rake receivers, 85–87
- RAMA (root-addressed multicast architecture), 417
- Random-access TDMA techniques, 83–85
- Random early detection (RED) technique, 198–200
- Random padding field, 235
- Random processes
 - in lossy compression, 463
 - in probability, 579–580
- Random service queuing systems, 280
- Random variables
 - functions of, 578–579
 - in probability, 574–577
- Randomizing traffic, 367
- RARP (Reverse Address Resolution Protocol), 106
- Rate-based resource allocation, 339
- Raw-image sampling, 456–459
- Reactive distributed routing, 552
- Reactive routing protocols, 529
- Real-Time Control Protocol (RTCP), 493–496
- Real-time media transport protocols, 490–491
 - RTCP, 493–496
 - RTP, 480–481, 491–493
- Real-time packets, 326
- Real-time sessions, 491–492
- Real-time transport protocol (RTP), 480–481, 491–493
- Rearrangeably nonblocking, 350, 386
- Reassembly, packet, 28–29, 66
- Receiver report (RR) packets, 494–496
- Receivers, CRC generation at, 90–92
- Recirculation
 - in packet recirculation multicast algorithm, 423–424
 - in switching networks, 367
- Reclustering, 551
- Recursive mapping, 230

- Recycling in packet recirculation multicast algorithm, 423
- RED (random early detection) technique, 198–200
- Redirect messages in ICMP, 30
- Redirect servers, 485
- Reflection paths, 76
- Regional handoffs, 149–150
- Regional Internet service providers, 7–9
- REGISTER messages, 486
- Registrar servers, 485
- Registrars in DNS, 232
- Registration in mobile IP, 156, 158–159
- Relays in RTP, 492
- Reliability in mobile IP, 156
- Reliable data delivery, 137
- Remote-access VPN, 433
- Remote controls, 72
- Remote login protocols, 232–233
 SSH, 234–235
 Telnet, 233–234
- Rendezvous points, 406
- Repeaters, 57–58
 LANs, 116–124
 ring topology, 102
 token-ring access protocol, 115
- Replication attacks, 254
- Reply messages, 231
- Reply packets, 522
- Request-expiration timers, 524, 526
- Request ID field, 244
- Request-to-send/clear-to-send (RTS/CTS) scheme, 137
- Request to Send (RTS) frames, 140
- Requests, mobile IP registration, 158
- Requests for comments (RFCs), 569–571
- RERR (route-error) messages, 520, 527
- Rescue operations, ad-hoc networks for, 512
- Resequencers, 66
- Reservation-access MAC protocols, 107
- Reservation-based protocols, 83–85
- Reservations in RSVP, 324–325
- Reset (RST) field, 211, 213
- Resource allocation, 337–338
 ATM networks, 340–344
 classification of, 338–339
 fairness in, 340
 management in, 338
- Resource oriented traffic engineering, 442
- Resource Reservation Protocol (RSVP), 324–325
- Responses, mobile IP registration, 158
- Reuse cluster of cells, 146
- Reverse Address Resolution Protocol (RARP), 106
- Reverse links, 154–155
- RF (radio frequency) spectrum, 72
- RFCs (requests for comments), 569–571
- Ring topology, 102
- RINGING signals, 486
- RIP (routing information protocol), 183–187, 515
- Rivert, Shamir, and Aldeman (RSA) algorithm, 260–262
- Root-addressed multicast architecture (RAMA), 417
- Root points, 406
- Root servers, 229
- Round-robin-access MAC protocols, 114–116
- Round robin queuing systems, 280
- Round-trip times (RTTs), 218, 221
- Rounds in LEACH, 546
- Route costs, 176, 182
- Route-creation process, 521
- Route discovery
 ABR, 522
 in ad-hoc networks, 530
- AODV, 524–526
- DSP, 519–520
- ICR, 555

- Route-error (RERR) messages, 520, 527
Route maintenance, 526–528
Route reconstruction, 522
Route reply (RREP) packets, 519, 525–526
Route request (RREQ) packets, 519, 524–525
Router based resource allocation, 339
Router ID field, 189
Router line cards, 60
Routers
 address assignments for, 173–174
 attacks on, 253
 description, 59
 input port processors for, 60–63
 for ISPs, 7
 in LANs, 116, 124–125
 in MPLS, 443
 in optical networks, 388–391
 output port processors for, 65–67
 in packet-switched networks, 9–10
 in RED, 198–200
 switch controllers in, 64–65
 switch fabric of, 63–64
 wireless, 141
 in WMNs, 164
Routing and internetworking, 7, 171–172
 in ad-hoc networks, 513–515
 algorithm characteristics, 172–173
 in AODV, 523–524
 in cellular networks, 150
 congestion control. *See* Congestion
 devices, 59
 exercises, 203–206
 interdomain routing protocols, 190–194
 intradomain routing protocols, 182–183
 OSPF, 187–190
 RIP, 183–187
 least-cost-path algorithms, 177–180
 for mobile IP, 159–163
 in MPLS domains, 439–441
 network-layer, 172–177
non-least-cost-path algorithms, 180–181
in packet-switched networks, 10
in sensor networks, 551–557
summary, 202–203
Routing caching timeouts, 524, 526
Routing information protocol (RIP), 183–187, 515
Routing tables
 AODV, 523
 automatic updates for, 120–121
 CGSR, 517
 DSDV, 515–516
 overflow attacks on, 529
 in packet-switched networks, 10
 poisoning attacks on, 252–253
 RIP, 185
 routers, 61–62
 WRP, 518
RR (receiver report) packets, 494–496
RREP (route reply) packets, 519, 525–526
RREQ (route request) packets, 519, 524–525
RSA (Rivert, Shamir, and Alderman) algorithm, 260–262
RSVP (Resource Reservation Protocol), 324–325
RTCP (Real-Time Control Protocol), 493–496
RTP (real-time transport protocol), 480–481, 491–493
RTP timestamp field, 495
RTS/CTS (request-to-send/clear-to-send) scheme, 137
RTTs (round-trip times), 218, 221
Run-length encoding, 468

S field, 439
S/P (serial-to-parallel) converters, 81
Sample space, 573
Sampling
 in compression, 451–452, 456–459

- Sampling (*continued*)
in probability, 574
- Satellite systems, 72, 130–131
- Scalability
CDMA, 86
multicasting, 402
VPNs, 434
WMNs, 165
- Scattering paths, 76
- Schedules
cell, 343–344
packet, 325–326
deficit round-robin, 333–334
earliest deadline first, 335
fair queueing, 331–332
first in, first out, 326–327
priority queueing, 327–331
weighted fair queueing, 332–333
- Scheduling discipline, 280
- SCP (Secure Copy Protocol), 237
- SCR (sustainable cell rate), 342
- SCTP (Stream Control Transmission Protocol), 500–502
- SDES (source descriptor) packets, 494
- SDMA (space-division multiple access), 87
- SDP (Session Description Protocol), 484
- SDSL (symmetric digital subscriber line), 55
- Secret-key encryption, 256–259
- Secure Copy Protocol (SCP), 237
- Secure Hash Algorithm (SHA), 263–265
- Secure Shell (SSH) Protocol, 234–235
- Security, 249–250
in ad-hoc networks, 528–531
authentication techniques, 256–257, 263–265
cryptographic techniques, 255–256
elements of, 250–251
exercises, 271–272
firewalls, 269–270
IP, 266–267
- mobile IP, 156, 163
network management for, 240–241
public-key encryption, 260–262
secret-key encryption, 257–259
summary, 270–271
threats to, 251–255
in VPNs, 436–437
in wireless networks, 267–268
- Security parameters index field, 266
- Segment field, 213–214
- Segments
TCP, 210–212, 218
in transport-layer packets, 209
UDP, 213–214
- Selective acknowledgment chunks, 502
- Self-organized overlay networks, 445
- Self-organizing sensor networks, 535
- Self-power units, 540
- Self-routing, 397
- Self-similarity
in multimedia, 503–506
in queuing models, 298–299
- Self-similarity parameter, 504
- Self-stabilization, 513, 530
- Semi-optical routers, 391
- Sender report (SR) packets, 494–496
- Sender's byte count field, 495
- Sender's packet count field, 495
- Sensing units, 539
- Sensor networks, 535–536
clustering in, 536–537, 545–551
communication energy model, 540–545
exercises, 561
node structure for, 538–540
protocol stack for, 537–538
related technologies, 559–560
routing protocols in, 551–552
intercluster, 554–557
intracluster, 552–554
simulation of, 557–559
summary, 560–561

- Sequence control (SC) field, 140
Sequence number field
 IPsec, 266
 RTP packets, 493
 TCP segments, 210
Sequences and sequence numbers
 AODV, 523
 in Shannon's coding theorem, 465–467
 TCP, 210
Sequential experiments, 574
Serial connections, 200–202, 352
Serial-to-parallel (S/P) converters, 81
Serial-to-parallel multiplexing units, 63
Servers
 authoritative, 230
 DNS. *See* Domain Name System (DNS)
 and servers
 in packet-switched networks, 10
 SIP, 484–485
 Web, 238–239
Service-level agreements (SLAs), 336
Service models in queueing systems, 281–283
Service-set identification (SSIDs), 134
Service sharing queuing systems, 280
Session Description Protocol (SDP), 484
Session Initiation Protocol (SIP), 483–486
Session layer, 22
Sessions
 H.323, 487–490
 RTCP, 494
 RTP, 491–492
 SIP, 485–486
Set and Trap PDUs, 244
7-layer OSI model, 22
Shadow fading, 77–78
Shannon's coding theorem, 465–467
Shapers, 335
Shared data buses, 368–369
Shared-memory switch fabrics, 365–366
Shared-tree technique, 406
Short IFS (SIFS) interval, 139
Shortest paths in SSN, 397
Shuffling property, 354
Shutdown chunks, 502
Shutdown acknowledgment chunks, 502
Shutdown complete chunks, 502
SIFS (short IFS) interval, 139
Signal regeneration, 57–58
Signal sampling, 451–452
Signal-to-noise ratio (SNR), 76, 79
Signaling protocols in VoIP, 480, 482–483
 H.323, 486–490
 SIP, 483–486
Signaling servers, 480
Signatures, digital, 265
Simple Mail Transfer Protocol (SMTP), 235–236
Simple multicast, 417
Simple Network Management Protocol (SNMP), 243–245
Simplicity of routing algorithms, 173
Single-key encryption protocols, 257
Single-path switching networks, 350
Single-source denial-of-service attacks, 254
SIP (Session Initiation Protocol), 483–486
Site-to-site VPNs, 433–434
Size, packet, 14–16
SLAs (service-level agreements), 336
Sliding-window flow control, 96–98
Slow start congestion control method, 219–220
SMI (structure of management information), 241
SMTP (Simple Mail Transfer Protocol), 235–236
SNMP (Simple Network Management Protocol), 243–245
SNR (signal-to-noise ratio), 76, 79
Software firewall programs, 269
Solar cells, 537

- SONET (synchronous optical network)
standard, 381
- Sorted deadline lists, 335
- Source address field
Ethernet LAN frames, 114
IGMP packets, 409
IP packets, 24
IPv6 packets, 31
- Source-based congestion avoidance, 221–222
- Source coding systems, 450–451
- Source descriptor (SDES) packets, 494
- Source encoder units, 463
- Source identification in RTCP, 494
- Source-initiated protocols, 514–515
- Source port field
SCTP packets, 501
TCP segments, 210
UDP segments, 213–214
- Source routing, 177
- Space-division multiple access (SDMA), 87
- Spanke-Benes switching networks, 387–388
- Spanning-tree algorithm, 123–124
- Sparse-mode algorithm, 406
- Sparse-mode PIM, 410–412, 416
- Spatial frequencies, 458
- Speed-up factor in switching networks, 367
- Spherical switching networks (SSNs), 395–398
- Spike noise, 88
- Split-horizon rule, 185
- Spread-spectrum techniques
in CDMA, 85–87
in LANs, 133
in physical layer, 136
- Squared-error distortion, 452
- SR (sender report) packets, 494–496
- SSH (Secure Shell) protocol, 234–235
- SSIDs (service-set identification), 134
- SSNs (spherical switching networks), 395–398
- SSRC field, 495
- Stability of routing algorithms, 173
- Stages in switching networks, 350
- Star couplers, 384, 388
- Star topology
for LANs, 102–103
for wireless networks, 131
- Start of frame field, 114
- State cookie error chunks, 502
- Static protocols, 514
- Static routing, 177
- Statistical multiplexing
TDM, 48–50
for virtual paths, 342
- Steady-state phase in LEACH, 546
- Still image compression, 455–461
- Stochastic processes, 579–580
- Stop-and-go model, 151–154
- Stop-and-wait flow control, 95–96
- Store-and-forward operation, 11
- Stream batches, 503
- Stream Control Transmission Protocol (SCTP), 500–502
- Streaming
audio, 462–463
non-Markovian, 503–506
QoS for, 499–500
- Strictly nonblocking networks, 351
- Structure of management information (SMI), 241
- Subnet addressing, 25–27
- Subnet IDs, 26
- Superframe intervals, 139
- Supernetting, 27
- Superposition, 306
- Sustainable cell rate (SCR), 342
- Switch controllers, 64–65
- Switch fabrics, 349
blocking, 350–351, 353–357
characteristics, 350
complexity of, 351

- concentration and expansion switches, 361–365
crossbar, 352–353
definitions and symbols in, 351–352
exercises, 376–378
features of, 351
multipath buffered crossbars, 368–369
 Markov chain model for, 371–374
 queueing model for, 369–370
 throughput and delay in, 374–375
nonblocking, 350–351, 357–362
performance of, 366–368
router, 63–64
shared-memory, 365–366
summary, 375–376
- Switches, 57–59
 in LANs, 116, 124–125, 131
 in optical networks, 384–388, 395–398
 three-dimensional, 424–426
- Switching nodes, 9
- Symbols
 in switching networks, 351–352
 in wireless networks, 155
- Symmetric digital subscriber line (SDSL), 55
- Symmetric encryption, 257
- Synchronization channels, 154
- Synchronization source identifier field
 RTCP packets, 494
 RTP packets, 493
- Synchronize (SYN) field, 211–212
- Synchronous MAC protocols, 106–107
- Synchronous optical network (SONET) standard, 381
- Synchronous TDM, 45–48
- System capacity in cellular networks, 146–147
- Table-driven routing protocols, 514
- Tagging in cell scheduling, 343
- Tail-drop policy, 198
- TCAs (traffic-conditioning agreements), 336
- TCP. *See* Transmission Control Protocol (TCP)
- TCP/IP (Transmission Control Protocol/Internet Protocol)
model, 20–22
- TCP normalized method, 222
- TDM (time-division multiplexing), 45
 statistical, 48–50
 synchronous, 45–48
- TDMA (time-division multiple access), 83–85
- Telephone Routing Over IP (TRIP) protocol, 486
- Telephone systems, 72. *See also* Cellular networks; Voice over IP (VoIP)
- Television systems, 72
- Telnet protocol, 233–234
- Temporally Ordered Routing Algorithm (TORA), 520–521
- Temporary address field, 159
- Thermo-optic switches, 384
- Threats to security
 ad-hoc networks, 529–530
 categories, 251–255
- Three-dimensional switches, 424–426
- Throughput
 in CSMA, 111–113
 in multipath buffered crossbars, 374–375
 in TCP, 217, 222
- Throughput to delay ratio, 340
- Time-division multiple access (TDMA), 83–85
- Time-division multiplexing (TDM), 45
 statistical, 48–50
 synchronous, 45–48
- Time to live field
 IP packets, 24
 MPLS packets, 439

- Timestamp field
 - PDUs, 245
 - RTP packets, 493
- Token arrival rate, 323
- Token-bucket traffic shaping, 322–324
- Token-ring access protocol, 114–116
- Tokens
 - in token-bucket traffic shaping, 323
 - in token-ring access protocol, 114
- Topologies for LANs, 6, 102–103
- TORA (Temporally Ordered Routing Algorithm), 520–521
- Torus-shaped topology, 396
- TOS (type of service) field
 - differentiated services QoS, 336
 - IP packets, 24
 - OSPF, 189
- Total length field, 24
- Total path attributes length field, 194
- Total system delay in queues, 330
- Traffic channels
 - in cellular networks, 145
 - in wireless networks, 155
- Traffic class field, 31
- Traffic classifiers, 336
- Traffic conditioners, 335–336
- Traffic-conditioning agreements (TCAs), 336
- Traffic-congestion case, 337–338
- Traffic engineering, 437, 442–443
- Traffic oriented traffic engineering, 442
- Traffic policing, 196–197, 318
- Traffic shaping, 317–318
 - leaky-bucket, 318–324
 - token-bucket, 322–324
- Transceivers, 143–144
- Translator relays, 492
- Transmission control blocks, 210
- Transmission Control Protocol (TCP), 209–210
 - applications of, 214
- congestion control, 217
 - additive increase, multiplicative decrease congestion control, 217–219
 - congestion avoidance, 221–222
 - fast retransmit method, 220–221
 - slow start method, 219–220
- connection setup for, 212–213
- for mobility, 215–216
- segments in, 210–212
- Transmission Control Protocol/Internet Protocol (TCP/IP) model, 20–22
- Transmission convergence in ATM, 34–35
- Transmissions
 - data link. *See* Data links
 - SSN, 397–398
- Transmit state in token-ring, 115
- Transmitters, CRC generation at, 90
- Transparent bridges, 121
- Transport and end-to-end protocols, 207–208
 - congestion control, 217–222
 - exercises, 223–224
 - mobile, 215–217
 - summary, 222–223
 - TCP, 209–212
 - transport layer for, 208–209
 - UDP, 213–214
- Transport layer, 21, 208–209
- Transport mode in IPsec, 267
- Tree algorithms, 404–406, 418–420
- Tree-based routing, 161
- TRIP (Telephone Routing Over IP) protocol, 486
- TRYING signals, 486
- Tunable dispersion compensators, 382
- Tunable lasers, 382
- Tunable optical filters, 388, 392
- Tunneling
 - in IPsec, 267
 - in MBone, 413–414

- in MPLS, 441–442
- in VPNs, 434–436
- Turbo codes, 79
- Twisted-pair links, 73
- Two-key encryption, 260
- Two random variable functions, 578
- Type data in ICR, 554–555
- Type field
 - IGMP packets, 408
 - for mobile IP, 159
 - OSPF packets, 189–190
 - SCTP packets, 501
 - SSH Packets, 235
- Type of service (ToS) field
 - differentiated services QoS, 336
 - IP packets, 24
 - OSPF, 189
- UDP (User Datagram Protocol), 213
 - applications, 214
 - for mobility, 216–217
 - segments, 213–214
- UDP checksum field, 213–214
- UDP length field, 213–214
- UID (user information database), 484
- Ultrapure fused silica, 74
- Unbuffered switching networks, 351
- Uncontrolled GFC, 38
- Unguided transmission links, 72
- UNI (user-network interface), 36–38, 381
- Unicast addresses, 32
- Unidirectional congestion control, 196–197
- Uniform random variables, 577
- Uniform resource locators (URLs), 238
- Uniform switching networks, 352
- Union of events, 573
- Update packets, 190
 - BGP, 192–193
 - TORA, 521
- Updates
 - in ad-hoc networks, 530
- automatic, 120–121
- RIP routing tables, 185
- Upstream bandwidth, ADSL, 55
- Upstream grant processors, 65
- Urgent (URG) field, 211
- Urgent pointer field, 211
- URLs (uniform resource locators), 238
- Usage parameter control, 343
- User agents
 - H.323, 488–489
 - SIP, 484–485
- User authentication, 150
- User Datagram Protocol (UDP), 213
 - applications, 214
 - for mobility, 216–217
 - segments, 213–214
- User information database (UID), 484
- User mailboxes, 236
- User-network interface (UNI), 36–38, 381
- User plane, 35–36
- User tracking in cellular networks, 150
- Utilization
 - in CSMA, 110
 - in feedback models, 304
 - link, 14–15
 - in M/M/1 queueing systems, 284
- Variable bit rate (VBR)
 - ATM, 35
 - sources, 342–343
- Variables in probability, 574–577
- Variance in probability, 577–578
- VCCs (virtual channel connections), 341–342
- VCIs (virtual channel identifiers), 34, 38
- VDSL (very high bit-rate digital subscriber line), 55
- Verification tag field, 501
- Version field
 - IP packets, 23
 - IPv6 packets, 31

- Version field (*continued*)
 RTCP packets, 494
 RTP packets, 492
- Version number field
 OSPF packets, 189
 RIP packets, 185–186
- Very high bit-rate digital subscriber line (VDSL), 55
- Virtual agents, 160
- Virtual channel connections (VCCs), 341–342
- Virtual channel identifiers (VCIs), 34, 38
- Virtual-circuit networks, 10–11, 13–14
- Virtual-circuit translation tables (VXTs), 64
- Virtual path connections (VPCs), 341–342
- Virtual path identifiers (VPIs), 34, 38
- Virtual private networks (VPNs), 431–432
 Diffie-Hillman key-exchange protocol
 for, 262
 exercises, 447–448
 MPLS-based, 443–444
 remote-access, 433
 security in, 436–437
 site-to-site, 433–434
 summary, 446–447
 tunneling and PPP in, 434–436
- Virtual registration, 160–161
- Voice compression
 quantization and distortion in, 452–455
 signal sampling in, 451–452
- Voice over IP (VoIP), 479–480
 overview, 480–481
 QoS, 481–482
 signaling protocols, 482–490
- VPCs (virtual path connections), 341–342
- VPIs (virtual path identifiers), 34, 38
- VPNs. *See* Virtual private networks (VPNs)
- VXTs (virtual-circuit translation tables), 64
- Walsh matrix, 155
- WANs (wide area networks), 9–10
- Wavelength-conversion gain, 394
- Wavelength-division multiplexing (WDM), 44–45, 383
- Wavelength in optical networks, 380
 allocation, 391–395
 conversion, 385, 389–391
- Wavelength routers, 388
- Wavelength routing nodes, 389–391
- WDM (wavelength-division multiplexing), 44–45, 383
- Web caching, 238–239
- Weighted-fair queueing (WFQ), 326
 scheduler, 332–333
 in VoIP, 482
- Weighted round-robin (WRR) scheduler, 333
- WEP (wired equivalent privacy) standard, 267–268
- White noise, 88
- Wide area networks (WANs), 9–10
- Wide-sense nonblocking networks, 350, 386
- Wiener random processes, 580
- WiFi (wireless fidelity) technology, 141–142
- WiMAX (worldwide interoperability for microwave access) technology, 163–164
- Window based resource allocation, 339
- Window scale option, 212
- Window size field, 211
- Window size in TCP, 211, 217–218
- Wired equivalent privacy (WEP) standard, 267–268
- Wired links, 72–74
- Wireless fidelity (WiFi) technology, 141–142
- Wireless links, 72–74
 antennas for, 74–76
 channels for, 76–79
 flat fading in, 79–80

- intersymbol interference in, 80–81
- OFDM for, 81–82
- Wireless networks, 129
 - ad-hoc. *See* Mobile ad-hoc networks (MANETs)
 - cellular. *See* Cellular networks
 - exercises, 168–170
 - IEEE 802.11, 134–142
 - infrastructure of, 130–131
 - LANs, 131–134
 - mobile IP. *See* Mobile IP
 - security in, 267–268
 - sensor. *See* Sensor networks
 - summary, 168
- WMNs, 163
 - applications of, 164–166
 - physical and MAC layers in, 167
 - WiMAX technology for, 163–164
- Wireless Routing Protocol (WRP), 517–519
- World Wide Web (WWW), 237–239
- Worldwide interoperability for microwave access (WiMAX) technology, 163–164
- WRP (Wireless Routing Protocol), 517–519
- WRR (weighted round-robin) scheduler, 333
- WWW (World Wide Web), 237–239
- ZigBee technology, 559–560