

FOURTH EDITION

Computer Networks

ANDREW S. TANENBAUM



Computer Networks

Fourth Edition

Andrew S. Tanenbaum

*Vrije Universiteit
Amsterdam, The Netherlands*



Prentice Hall PTR
Upper Saddle River, NJ 07458
www.phptr.com

Library of Congress Cataloging-in-Publication Data

Tanenbaum, Andrew S.
Computer networks / Andrew S. Tanenbaum.--4th ed.
p. cm.
Includes bibliographical references.
ISBN 0-13-066102-3
1. Computer networks. I. Title.
TK5105.5 .T36 2002
004.6--dc21 2002029263

Editorial/production supervision: *Patti Guerrieri*
Cover design director: *Jerry Votta*
Cover designer: *Anthony Gemmellaro*
Cover design: *Andrew S. Tanenbaum*
Art director: *Gail Cocker-Bogusz*
Interior Design: *Andrew S. Tanenbaum*
Interior graphics: *Hadel Studio*
Typesetting: *Andrew S. Tanenbaum*
Manufacturing buyer: *Maura Zaldivar*
Executive editor: *Mary Franz*
Editorial assistant: *Noreen Regina*
Marketing manager: *Dan DePasquale*



© 2003 Pearson Education, Inc.
Publishing as Prentice Hall PTR
Upper Saddle River, New Jersey 07458

Prentice Hall books are widely used by corporations and government agencies for training, marketing, and resale.

For information regarding corporate and government bulk discounts please contact:
Corporate and Government Sales (800) 382-3419 or corpsales@pearsontechgroup.com

All products or services mentioned in this book are the trademarks or service marks of their respective companies or organizations.

All rights reserved. No part of this book may be reproduced, in any form or by any means, without permission in writing from the publisher.

Printed in the United States of America

10 9 8 7 6 5

ISBN 0-13-066102-3

Pearson Education LTD.
Pearson Education Australia PTY, Limited
Pearson Education Singapore, Pte. Ltd.
Pearson Education North Asia Ltd.
Pearson Education Canada, Ltd.
Pearson Educación de México, S.A. de C.V.
Pearson Education — Japan
Pearson Education Malaysia, Pte. Ltd.

To Suzanne, Barbara, Marvin, and the memory of Bram and Sweetie π

CONTENTS

PREFACE

1 INTRODUCTION

1

1.1 USES OF COMPUTER NETWORKS 3

- 1.1.1 Business Applications 3
- 1.1.2 Home Applications 6
- 1.1.3 Mobile Users 9
- 1.1.4 Social Issues 12

1.2 NETWORK HARDWARE 14

- 1.2.1 Local Area Networks 16
- 1.2.2 Metropolitan Area Networks 18
- 1.2.3 Wide Area Networks 19
- 1.2.4 Wireless Networks 21
- 1.2.5 Home Networks 23
- 1.2.6 Internetworks 25

1.3 NETWORK SOFTWARE 26

- 1.3.1 Protocol Hierarchies 26
- 1.3.2 Design Issues for the Layers 30
- 1.3.3 Connection-Oriented and Connectionless Services 32
- 1.3.4 Service Primitives 34
- 1.3.5 The Relationship of Services to Protocols 36

1.4 REFERENCE MODELS	37
1.4.1 The OSI Reference Model	37
1.4.2 The TCP/IP Reference Model	41
1.4.3 A Comparison of the OSI and TCP/IP Reference Models	44
1.4.4 A Critique of the OSI Model and Protocols	46
1.4.5 A Critique of the TCP/IP Reference Model	48
1.5 EXAMPLE NETWORKS	49
1.5.1 The Internet	50
1.5.2 Connection-Oriented Networks: X.25, Frame Relay, and ATM	59
1.5.3 Ethernet	65
1.5.4 Wireless LANs: 802.11	68
1.6 NETWORK STANDARDIZATION	71
1.6.1 Who's Who in the Telecommunications World	71
1.6.2 Who's Who in the International Standards World	74
1.6.3 Who's Who in the Internet Standards World	75
1.7 METRIC UNITS	77
1.8 OUTLINE OF THE REST OF THE BOOK	78
1.9 SUMMARY	79

2 THE PHYSICAL LAYER

85

2.1 THE THEORETICAL BASIS FOR DATA COMMUNICATION	85
2.1.1 Fourier Analysis	86
2.1.2 Bandwidth-Limited Signals	86
2.1.3 The Maximum Data Rate of a Channel	89
2.2 GUIDED TRANSMISSION MEDIA	90
2.2.1 Magnetic Media	90
2.2.2 Twisted Pair	91
2.2.3 Coaxial Cable	92
2.2.4 Fiber Optics	93
2.3 WIRELESS TRANSMISSION	100
2.3.1 The Electromagnetic Spectrum	100
2.3.2 Radio Transmission	103

2.3.3 Microwave Transmission	104
2.3.4 Infrared and Millimeter Waves	106
2.3.5 Lightwave Transmission	107
2.4 COMMUNICATION SATELLITES	109
2.4.1 Geostationary Satellites	109
2.4.2 Medium-Earth Orbit Satellites	113
2.4.3 Low-Earth Orbit Satellites	114
2.4.4 Satellites versus Fiber	117
2.5 THE PUBLIC SWITCHED TELEPHONE NETWORK	118
2.5.1 Structure of the Telephone System	119
2.5.2 The Politics of Telephones	122
2.5.3 The Local Loop: Modems, ADSL, and Wireless	124
2.5.4 Trunks and Multiplexing	137
2.5.5 Switching	146
2.6 THE MOBILE TELEPHONE SYSTEM	152
2.6.1 First-Generation Mobile Phones: Analog Voice	153
2.6.2 Second-Generation Mobile Phones: Digital Voice	157
2.6.3 Third-Generation Mobile Phones: Digital Voice and Data	166
2.7 CABLE TELEVISION	169
2.7.1 Community Antenna Television	169
2.7.2 Internet over Cable	170
2.7.3 Spectrum Allocation	172
2.7.4 Cable Modems	173
2.7.5 ADSL versus Cable	175
2.8 SUMMARY	177

3 THE DATA LINK LAYER

183

3.1 DATA LINK LAYER DESIGN ISSUES	184
3.1.1 Services Provided to the Network Layer	184
3.1.2 Framing	187
3.1.3 Error Control	191
3.1.4 Flow Control	192

3.2 ERROR DETECTION AND CORRECTION	192
3.2.1 Error-Correcting Codes	193
3.2.2 Error-Detecting Codes	196
3.3 ELEMENTARY DATA LINK PROTOCOLS	200
3.3.1 An Unrestricted Simplex Protocol	204
3.3.2 A Simplex Stop-and-Wait Protocol	206
3.3.3 A Simplex Protocol for a Noisy Channel	208
3.4 SLIDING WINDOW PROTOCOLS	211
3.4.1 A One-Bit Sliding Window Protocol	214
3.4.2 A Protocol Using Go Back N	216
3.4.3 A Protocol Using Selective Repeat	223
3.5 PROTOCOL VERIFICATION	229
3.5.1 Finite State Machine Models	229
3.5.2 Petri Net Models	232
3.6 EXAMPLE DATA LINK PROTOCOLS	234
3.6.1 HDLC—High-Level Data Link Control	234
3.6.2 The Data Link Layer in the Internet	237
3.7 SUMMARY	242

4 THE MEDIUM ACCESS CONTROL SUBLAYER 247

4.1 THE CHANNEL ALLOCATION PROBLEM	248
4.1.1 Static Channel Allocation in LANs and MANs	248
4.1.2 Dynamic Channel Allocation in LANs and MANs	249
4.2 MULTIPLE ACCESS PROTOCOLS	251
4.2.1 ALOHA	251
4.2.2 Carrier Sense Multiple Access Protocols	255
4.2.3 Collision-Free Protocols	259
4.2.4 Limited-Contention Protocols	261
4.2.5 Wavelength Division Multiple Access Protocols	265
4.2.6 Wireless LAN Protocols	267

4.3 ETHERNET	271
4.3.1 Ethernet Cabling	271
4.3.2 Manchester Encoding	274
4.3.3 The Ethernet MAC Sublayer Protocol	275
4.3.4 The Binary Exponential Backoff Algorithm	278
4.3.5 Ethernet Performance	279
4.3.6 Switched Ethernet	281
4.3.7 Fast Ethernet	283
4.3.8 Gigabit Ethernet	286
4.3.9 IEEE 802.2: Logical Link Control	290
4.3.10 Retrospective on Ethernet	291
4.4 WIRELESS LANS	292
4.4.1 The 802.11 Protocol Stack	292
4.4.2 The 802.11 Physical Layer	293
4.4.3 The 802.11 MAC Sublayer Protocol	295
4.4.4 The 802.11 Frame Structure	299
4.4.5 Services	3-1
4.5 BROADBAND WIRELESS	302
4.5.1 Comparison of 802.11 with 802.16	303
4.5.2 The 802.16 Protocol Stack	305
4.5.3 The 802.16 Physical Layer	306
4.5.4 The 802.16 MAC Sublayer Protocol	307
4.5.5 The 802.16 Frame Structure	309
4.6 BLUETOOTH	310
4.6.1 Bluetooth Architecture	311
4.6.2 Bluetooth Applications	312
4.6.3 The Bluetooth Protocol Stack	313
4.6.4 The Bluetooth Radio Layer	315
4.6.5 The Bluetooth Baseband Layer	315
4.6.6 The Bluetooth L2CAP Layer	316
4.6.7 The Bluetooth Frame Structure	316
4.7 DATA LINK LAYER SWITCHING	318
4.7.1 Bridges from 802.x to 802.y	320
4.7.2 Local Internetworking	322
4.7.3 Spanning Tree Bridges	324
4.7.4 Remote Bridges	325
4.7.5 Repeaters, Hubs, Bridges, Switches, Routers, and Gateways	326
4.7.6 Virtual LANs	329
4.8 SUMMARY	337

5 THE NETWORK LAYER**343**

- 5.1 NETWORK LAYER DESIGN ISSUES 343
 - 5.1.1 Store-and-Forward Packet Switching 344
 - 5.1.2 Services Provided to the Transport Layer 344
 - 5.1.3 Implementation of Connectionless Service 345
 - 5.1.4 Implementation of Connection-Oriented Service 347
 - 5.1.5 Comparison of Virtual-Circuit and Datagram Subnets 348
- 5.2 ROUTING ALGORITHMS 350
 - 5.2.1 The Optimality Principle 352
 - 5.2.2 Shortest Path Routing 353
 - 5.2.3 Flooding 355
 - 5.2.4 Distance Vector Routing 357
 - 5.2.5 Link State Routing 360
 - 5.2.6 Hierarchical Routing 366
 - 5.2.7 Broadcast Routing 368
 - 5.2.8 Multicast Routing 370
 - 5.2.9 Routing for Mobile Hosts 372
 - 5.2.10 Routing in Ad Hoc Networks 373
 - 5.2.11 Node Lookup in Peer-to-Peer Networks 380
- 5.3 CONGESTION CONTROL ALGORITHMS 384
 - 5.3.1 General Principles of Congestion Control 386
 - 5.3.2 Congestion Prevention Policies 388
 - 5.3.3 Congestion Control in Virtual-Circuit Subnets 389
 - 5.3.4 Congestion Control in Datagram Subnets 391
 - 5.3.5 Load Shedding 394
 - 5.3.6 Jitter Control 395
- 5.4 QUALITY OF SERVICE 397
 - 5.4.1 Requirements 397
 - 5.4.2 Techniques for Achieving Good Quality of Service 398
 - 5.4.3 Integrated Services 409
 - 5.4.4 Differentiated Services 412
 - 5.4.5 Label Switching and MPLS 415
- 5.5 INTERNETWORKING 418
 - 5.5.1 How Networks Differ 419
 - 5.5.2 How Networks Can Be Connected 420
 - 5.5.3 Concatenated Virtual Circuits 422
 - 5.5.4 Connectionless Internetworking 423

5.5.5 Tunneling	425
5.5.6 Internetwork Routing	426
5.5.7 Fragmentation	427
5.6 THE NETWORK LAYER IN THE INTERNET	431
5.6.1 The IP Protocol	433
5.6.2 IP Addresses	436
5.6.3 Internet Control Protocols	449
5.6.4 OSPF—The Interior Gateway Routing Protocol	454
5.6.5 BGP—The Exterior Gateway Routing Protocol	459
5.6.6 Internet Multicasting	461
5.6.7 Mobile IP	462
5.6.8 IPv6	464
5.7 SUMMARY	473

6 THE TRANSPORT LAYER 481

6.1 THE TRANSPORT SERVICE	481
6.1.1 Services Provided to the Upper Layers	481
6.1.2 Transport Service Primitives	483
6.1.3 Berkeley Sockets	487
6.1.4 An Example of Socket Programming: An Internet File Server	488
6.2 ELEMENTS OF TRANSPORT PROTOCOLS	492
6.2.1 Addressing	493
6.2.2 Connection Establishment	496
6.2.3 Connection Release	502
6.2.4 Flow Control and Buffering	506
6.2.5 Multiplexing	510
6.2.6 Crash Recovery	511
6.3 A SIMPLE TRANSPORT PROTOCOL	513
6.3.1 The Example Service Primitives	513
6.3.2 The Example Transport Entity	515
6.3.3 The Example as a Finite State Machine	522
6.4 THE INTERNET TRANSPORT PROTOCOLS: UDP	524
6.4.1 Introduction to UDP	525
6.4.2 Remote Procedure Call	526
6.4.3 The Real-Time Transport Protocol	529

6.5 THE INTERNET TRANSPORT PROTOCOLS: TCP	532
6.5.1 Introduction to TCP	532
6.5.2 The TCP Service Model	533
6.5.3 The TCP Protocol	535
6.5.4 The TCP Segment Header	536
6.5.5 TCP Connection Establishment	539
6.5.6 TCP Connection Release	541
6.5.7 Modeling TCP Connection Management	541
6.5.8 TCP Transmission Policy	543
6.5.9 TCP Congestion Control	547
6.5.10 TCP Timer Management	550
6.5.11 Wireless TCP and UDP	553
6.5.12 Transactional TCP	555
6.6 PERFORMANCE ISSUES	557
6.6.1 Performance Problems in Computer Networks	557
6.6.2 Network Performance Measurement	560
6.6.3 System Design for Better Performance	562
6.6.4 Fast TPDU Processing	566
6.6.5 Protocols for Gigabit Networks	569
6.7 SUMMARY	573

7 THE APPLICATION LAYER 579

7.1 DNS—THE DOMAIN NAME SYSTEM	579
7.1.1 The DNS Name Space	580
7.1.2 Resource Records	582
7.1.3 Name Servers	586
7.2 ELECTRONIC MAIL	588
7.2.1 Architecture and Services	590
7.2.2 The User Agent	591
7.2.3 Message Formats	594
7.2.4 Message Transfer	602
7.2.5 Final Delivery	605
7.3 THE WORLD WIDE WEB	611
7.3.1 Architectural Overview	612
7.3.2 Static Web Documents	629

7.3.3 Dynamic Web Documents	643
7.3.4 HTTP—The HyperText Transfer Protocol	651
7.3.5 Performance Enhancements	656
7.3.6 The Wireless Web	662
7.4 MULTIMEDIA	674
7.4.1 Introduction to Digital Audio	674
7.4.2 Audio Compression	676
7.4.3 Streaming Audio	679
7.4.4 Internet Radio	683
7.4.5 Voice over IP	685
7.4.6 Introduction to Video	692
7.4.7 Video Compression	696
7.4.8 Video on Demand	704
7.4.9 The MBone—The Multicast Backbone	711
7.5 SUMMARY	714

8 NETWORK SECURITY

721

8.1 CRYPTOGRAPHY	724
8.1.1 Introduction to Cryptography	725
8.1.2 Substitution Ciphers	727
8.1.3 Transposition Ciphers	729
8.1.4 One-Time Pads	730
8.1.5 Two Fundamental Cryptographic Principles	735
8.2 SYMMETRIC-KEY ALGORITHMS	737
8.2.1 DES—The Data Encryption Standard	738
8.2.2 AES—The Advanced Encryption Standard	741
8.2.3 Cipher Modes	745
8.2.4 Other Ciphers	750
8.2.5 Cryptanalysis	750
8.3 PUBLIC-KEY ALGORITHMS	752
8.3.1 RSA	753
8.3.2 Other Public-Key Algorithms	755

8.4 DIGITAL SIGNATURES	755
8.4.1 Symmetric-Key Signatures	756
8.4.2 Public-Key Signatures	757
8.4.3 Message Digests	759
8.4.4 The Birthday Attack	763
8.5 MANAGEMENT OF PUBLIC KEYS	765
8.5.1 Certificates	765
8.5.2 X.509	767
8.5.3 Public Key Infrastructures	768
8.6 COMMUNICATION SECURITY	772
8.6.1 IPsec	772
8.6.2 Firewalls	776
8.6.3 Virtual Private Networks	779
8.6.4 Wireless Security	780
8.7 AUTHENTICATION PROTOCOLS	785
8.7.1 Authentication Based on a Shared Secret Key	786
8.7.2 Establishing a Shared Key; The Diffie-Hellman Key Exchange	791
8.7.3 Authentication Using a Key Distribution Center	793
8.7.4 Authentication Using Kerberos	796
8.7.5 Authentication Using Public-Key Cryptography	798
8.8 E-MAIL SECURITY	799
8.8.1 PGP—Pretty Good Privacy	799
8.8.2 PEM—Privacy Enhanced Mail	803
8.8.3 S/MIME	804
8.9 WEB SECURITY	805
8.9.1 Threats	8-5
8.9.2 Secure Naming	806
8.9.3 SSL—The Secure Sockets Layer	813
8.9.4 Mobile Code Security	816
8.10 SOCIAL ISSUES	819
8.10.1 Privacy	819
8.10.2 Freedom of Speech	922
8.10.3 Copyright	826
8.11 SUMMARY	828

9 READING LIST AND BIBLIOGRAPHY 835**9.1 SUGGESTIONS FOR FURTHER READING 835**

9.1.1 Introduction and General Works 836

9.1.2 The Physical Layer 838

9.1.3 The Data Link Layer 840

9.1.4 The Medium Access Control Sublayer 840

9.1.5 The Network Layer 842

9.1.6 The Transport Layer 844

9.1.7 The Application Layer 844

9.1.8 Network Security 846

9.2 ALPHABETICAL BIBLIOGRAPHY 848**INDEX 869**

PREFACE

This book is now in its fourth edition. Each edition has corresponded to a different phase in the way computer networks were used. When the first edition appeared in 1980, networks were an academic curiosity. When the second edition appeared in 1988, networks were used by universities and large businesses. When the third edition appeared in 1996, computer networks, especially the Internet, had become a daily reality for millions of people. The new item in the fourth edition is the rapid growth of wireless networking in many forms.

The networking picture has changed radically since the third edition. In the mid-1990s, numerous kinds of LANs and WANs existed, along with multiple protocol stacks. By 2003, the only wired LAN in widespread use was Ethernet, and virtually all WANs were on the Internet. Accordingly, a large amount of material about these older networks has been removed.

However, new developments are also plentiful. The most important is the huge increase in wireless networks, including 802.11, wireless local loops, 2G and 3G cellular networks, Bluetooth, WAP, i-mode, and others. Accordingly, a large amount of material has been added on wireless networks. Another newly-important topic is security, so a whole chapter on it has been added.

Although Chap. 1 has the same introductory function as it did in the third edition, the contents have been revised and brought up to date. For example, introductions to the Internet, Ethernet, and wireless LANs are given there, along with some history and background. Home networking is also discussed briefly.

Chapter 2 has been reorganized somewhat. After a brief introduction to the principles of data communication, there are three major sections on transmission (guided media, wireless, and satellite), followed by three more on important examples (the public switched telephone system, the mobile telephone system, and cable television). Among the new topics covered in this chapter are ADSL, broadband wireless, wireless MANs, and Internet access over cable and DOCSIS.

Chapter 3 has always dealt with the fundamental principles of point-to-point protocols. These ideas are essentially timeless and have not changed for decades.

Accordingly, the series of detailed example protocols presented in this chapter is largely unchanged from the third edition.

In contrast, the MAC sublayer has been an area of great activity in recent years, so many changes are present in Chap. 4. The section on Ethernet has been expanded to include gigabit Ethernet. Completely new are major sections on wireless LANs, broadband wireless, Bluetooth, and data link layer switching, including MPLS.

Chapter 5 has also been updated, with the removal of all the ATM material and the addition of additional material on the Internet. Quality of service is now also a major topic, including discussions of integrated services and differentiated services. Wireless networks are also present here, with a discussion of routing in ad hoc networks. Other new topics include NAT and peer-to-peer networks.

Chap. 6 is still about the transport layer, but here, too, some changes have occurred. Among these is an example of socket programming. A one-page client and a one-page server are given in C and discussed. These programs, available on the book's Web site, can be compiled and run. Together they provide a primitive remote file or Web server available for experimentation. Other new topics include remote procedure call, RTP, and transaction/TCP.

Chap. 7, on the application layer, has been more sharply focused. After a short introduction to DNS, the rest of the chapter deals with just three topics: e-mail, the Web, and multimedia. But each topic is treated in great detail. The discussion of how the Web works is now over 60 pages, covering a vast array of topics, including static and dynamic Web pages, HTTP, CGI scripts, content delivery networks, cookies, and Web caching. Material is also present on how modern Web pages are written, including brief introductions to XML, XSL, XHTML, PHP, and more, all with examples that can be tested. The wireless Web is also discussed, focusing on i-mode and WAP. The multimedia material now includes MP3, streaming audio, Internet radio, and voice over IP.

Security has become so important that it has now been expanded to a complete chapter of over 100 pages. It covers both the principles of security (symmetric- and public-key algorithms, digital signatures, and X.509 certificates) and the applications of these principles (authentication, e-mail security, and Web security). The chapter is both broad (ranging from quantum cryptography to government censorship) and deep (e.g., how SHA-1 works in detail).

Chapter 9 contains an all-new list of suggested readings and a comprehensive bibliography of over 350 citations to the current literature. Over 200 of these are to papers and books written in 2000 or later.

Computer books are full of acronyms. This one is no exception. By the time you are finished reading this one, the following should ring a bell: ADSL, AES, AMPS, AODV, ARP, ATM, BGP, CDMA, CDN, CGI, CIDR, DCF, DES, DHCP, DMCA, FDM, FHSS, GPRS, GSM, HDLC, HFC, HTML, HTTP, ICMP, IMAP, ISP, ITU, LAN, LMDS, MAC, MACA, MIME, MPEG, MPLS, MTU, NAP, NAT, NSA, NTSC, OFDM, OSPF, PCF, PCM, PGP, PHP, PKI, POTS,

PPP, PSTN, QAM, QPSK, RED, RFC, RPC, RSA, RSVP, RTP, SSL, TCP, TDM, UDP, URL, UTP, VLAN, VPN, VSAT, WAN, WAP, WDMA, WEP, WWW, and XML. But don't worry. Each will be carefully defined before it is used.

To help instructors using this book as a text for a course, the author has prepared various teaching aids, including

- A problem solutions manual.
- Files containing the figures in multiple formats.
- PowerPoint sheets for a course using the book.
- A simulator (written in C) for the example protocols of Chap. 3.
- A Web page with links to many tutorials, organizations, FAQs, etc.

The solutions manual is available directly from Prentice Hall (but **only** to instructors, not to students). All the other material is on the book's Web site:

<http://www.prenhall.com/tanenbaum>

From there, click on the book's cover.

Many people helped me during the course of the fourth edition. I would especially like to thank the following people: Ross Anderson, Elizabeth Belding-Royer, Steve Bellovin, Chatschik Bisdikian, Kees Bot, Scott Bradner, Jennifer Bray, Pat Cain, Ed Felten, Warwick Ford, Kevin Fu, Ron Fulle, Jim Geier, Mario Gerla, Natalie Giroux, Steve Hanna, Jeff Hayes, Amir Herzberg, Philip Homburg, Philipp Hoschka, David Green, Bart Jacobs, Frans Kaashoek, Steve Kent, Roger Kermode, Robert Kinicki, Shay Kutten, Rob Lanphier, Marcus Leech, Tom Maufer, Brent Miller, Shivakant Mishra, Thomas Nadeau, Shlomo Ovadia, Kaveh Pahlavan, Radia Perlman, Guillaume Pierre, Wayne Pleasant, Patrick Powell, Thomas Robertazzi, Medy Sanadidi, Christian Schmutzer, Henning Schulzrinne, Paul Sevinc, Mihail Sichitiu, Bernard Sklar, Ed Skoudis, Bob Strader, George Swallow, George Thiruvathukal, Peter Tomsu, Patrick Verkaik, Dave Vittali, Spyros Voulgaris, Jan-Mark Wams, Ruediger Weis, Bert Wijnen, Joseph Wilkes, Leendert van Doorn, and Maarten van Steen.

Special thanks go to Trudy Levine for proving that grandmothers can do a fine job of reviewing technical material. Shivakant Mishra thought of many challenging end-of-chapter problems. Andy Dornan suggested additional readings for Chap. 9. Jan Looyen provided essential hardware at a critical moment. Dr. F. de Nies did an expert cut-and-paste job right when it was needed. My editor at Prentice Hall, Mary Franz, provided me with more reading material than I had consumed in the previous 7 years and was helpful in numerous other ways as well.

Finally, we come to the most important people: Suzanne, Barbara, and Marvin. To Suzanne for her love, patience, and picnic lunches. To Barbara and Marvin for being fun and cheery all the time (except when complaining about awful college textbooks, thus keeping me on my toes). Thank you.

ANDREW S. TANENBAUM

Some internet protocols take this method even further and consider the entire transmission on a virtual circuit to be one giant packet, so that each fragment contains the absolute byte number of the first byte within the fragment.

5.6 THE NETWORK LAYER IN THE INTERNET

Before getting into the specifics of the network layer in the Internet, it is worth taking a look at the principles that drove its design in the past and made it the success that it is today. All too often, nowadays, people seem to have forgotten them. These principles are enumerated and discussed in RFC 1958, which is well worth reading (and should be mandatory for all protocol designers—with a final exam at the end). This RFC draws heavily on ideas found in (Clark, 1988; and Saltzer et al., 1984). We will now summarize what we consider to be the top 10 principles (from most important to least important).

1. **Make sure it works.** Do not finalize the design or standard until multiple prototypes have successfully communicated with each other. All too often designers first write a 1000-page standard, get it approved, then discover it is deeply flawed and does not work. Then they write version 1.1 of the standard. This is not the way to go.
2. **Keep it simple.** When in doubt, use the simplest solution. William of Occam stated this principle (Occam's razor) in the 14th century. Put in modern terms: fight features. If a feature is not absolutely essential, leave it out, especially if the same effect can be achieved by combining other features.
3. **Make clear choices.** If there are several ways of doing the same thing, choose one. Having two or more ways to do the same thing is looking for trouble. Standards often have multiple options or modes or parameters because several powerful parties insist that their way is best. Designers should strongly resist this tendency. Just say no.
4. **Exploit modularity.** This principle leads directly to the idea of having protocol stacks, each of whose layers is independent of all the other ones. In this way, if circumstances that require one module or layer to be changed, the other ones will not be affected.
5. **Expect heterogeneity.** Different types of hardware, transmission facilities, and applications will occur on any large network. To handle them, the network design must be simple, general, and flexible.
6. **Avoid static options and parameters.** If parameters are unavoidable (e.g., maximum packet size), it is best to have the sender and receiver negotiate a value than defining fixed choices.

7. **Look for a good design; it need not be perfect.** Often the designers have a good design but it cannot handle some weird special case. Rather than messing up the design, the designers should go with the good design and put the burden of working around it on the people with the strange requirements.
8. **Be strict when sending and tolerant when receiving.** In other words, only send packets that rigorously comply with the standards, but expect incoming packets that may not be fully conformant and try to deal with them.
9. **Think about scalability.** If the system is to handle millions of hosts and billions of users effectively, no centralized databases of any kind are tolerable and load must be spread as evenly as possible over the available resources.
10. **Consider performance and cost.** If a network has poor performance or outrageous costs, nobody will use it.

Let us now leave the general principles and start looking at the details of the Internet's network layer. At the network layer, the Internet can be viewed as a collection of subnetworks or **Autonomous Systems (ASes)** that are interconnected. There is no real structure, but several major backbones exist. These are constructed from high-bandwidth lines and fast routers. Attached to the backbones are regional (midlevel) networks, and attached to these regional networks are the LANs at many universities, companies, and Internet service providers. A sketch of this quasi-hierarchical organization is given in Fig. 5-52.

The glue that holds the whole Internet together is the network layer protocol, **IP (Internet Protocol)**. Unlike most older network layer protocols, it was designed from the beginning with internetworking in mind. A good way to think of the network layer is this. Its job is to provide a best-efforts (i.e., not guaranteed) way to transport datagrams from source to destination, without regard to whether these machines are on the same network or whether there are other networks in between them.

Communication in the Internet works as follows. The transport layer takes data streams and breaks them up into datagrams. In theory, datagrams can be up to 64 Kbytes each, but in practice they are usually not more than 1500 bytes (so they fit in one Ethernet frame). Each datagram is transmitted through the Internet, possibly being fragmented into smaller units as it goes. When all the pieces finally get to the destination machine, they are reassembled by the network layer into the original datagram. This datagram is then handed to the transport layer, which inserts it into the receiving process' input stream. As can be seen from Fig. 5-52, a packet originating at host 1 has to traverse six networks to get to host 2. In practice, it is often much more than six.

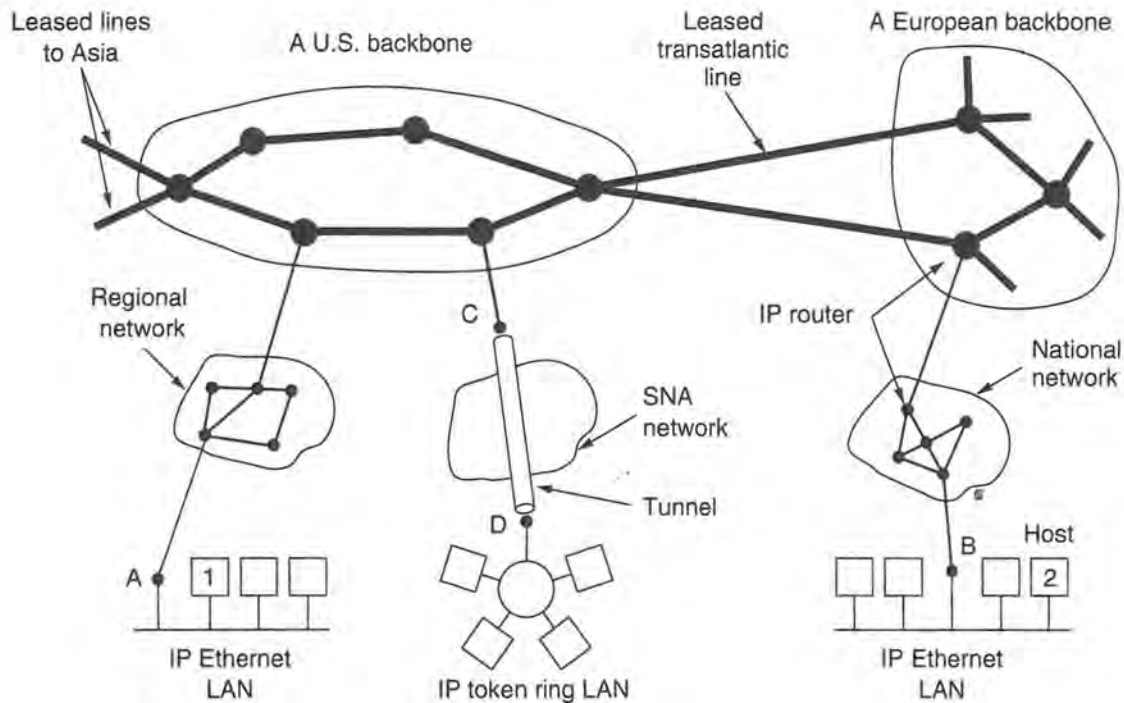


Figure 5-52. The Internet is an interconnected collection of many networks.

5.6.1 The IP Protocol

An appropriate place to start our study of the network layer in the Internet is the format of the IP datagrams themselves. An IP datagram consists of a header part and a text part. The header has a 20-byte fixed part and a variable length optional part. The header format is shown in Fig. 5-53. It is transmitted in big-endian order: from left to right, with the high-order bit of the *Version* field going first. (The SPARC is big endian; the Pentium is little-endian.) On little endian machines, software conversion is required on both transmission and reception.

The *Version* field keeps track of which version of the protocol the datagram belongs to. By including the version in each datagram, it becomes possible to have the transition between versions take years, with some machines running the old version and others running the new one. Currently a transition between IPv4 and IPv6 is going on, has already taken years, and is by no means close to being finished (Durand, 2001; Wiljakka, 2002; and Waddington and Chang, 2002). Some people even think it will never happen (Weiser, 2001). As an aside on numbering, IPv5 was an experimental real-time stream protocol that was never widely used.

Since the header length is not constant, a field in the header, *IHL*, is provided to tell how long the header is, in 32-bit words. The minimum value is 5, which applies when no options are present. The maximum value of this 4-bit field is 15,

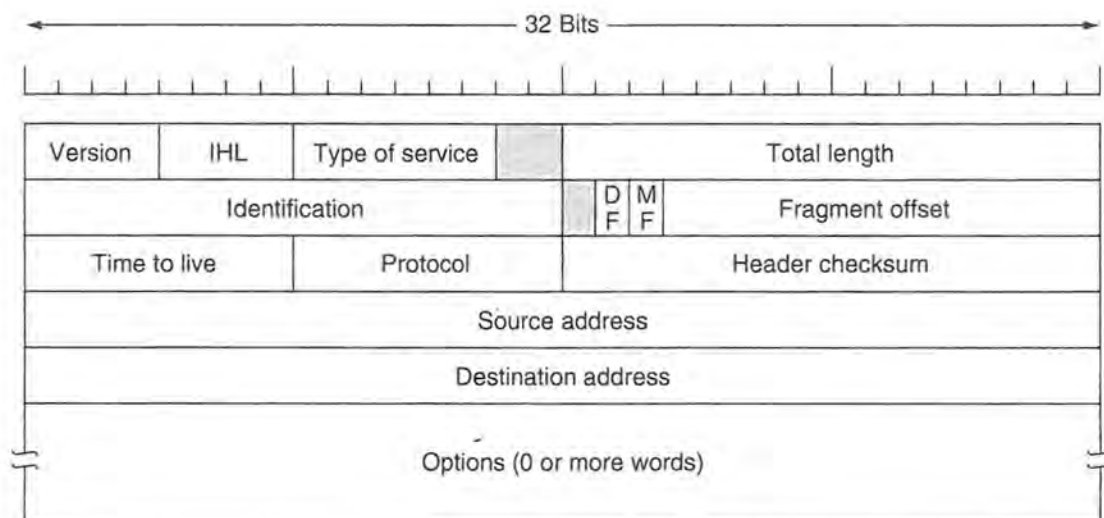


Figure 5-53. The IPv4 (Internet Protocol) header.

which limits the header to 60 bytes, and thus the *Options* field to 40 bytes. For some options, such as one that records the route a packet has taken, 40 bytes is far too small, making that option useless.

The *Type of service* field is one of the few fields that has changed its meaning (slightly) over the years. It was and is still intended to distinguish between different classes of service. Various combinations of reliability and speed are possible. For digitized voice, fast delivery beats accurate delivery. For file transfer, error-free transmission is more important than fast transmission.

Originally, the 6-bit field contained (from left to right), a three-bit *Precedence* field and three flags, *D*, *T*, and *R*. The *Precedence* field was a priority, from 0 (normal) to 7 (network control packet). The three flag bits allowed the host to specify what it cared most about from the set {Delay, Throughput, Reliability}. In theory, these fields allow routers to make choices between, for example, a satellite link with high throughput and high delay or a leased line with low throughput and low delay. In practice, current routers often ignore the *Type of service* field altogether.

Eventually, IETF threw in the towel and changed the field slightly to accommodate differentiated services. Six of the bits are used to indicate which of the service classes discussed earlier each packet belongs to. These classes include the four queueing priorities, three discard probabilities, and the historical classes.

The *Total length* includes everything in the datagram—both header and data. The maximum length is 65,535 bytes. At present, this upper limit is tolerable, but with future gigabit networks, larger datagrams may be needed.

The *Identification* field is needed to allow the destination host to determine which datagram a newly arrived fragment belongs to. All the fragments of a datagram contain the same *Identification* value.

Next comes an unused bit and then two 1-bit fields. *DF* stands for Don't Fragment. It is an order to the routers not to fragment the datagram because the destination is incapable of putting the pieces back together again. For example, when a computer boots, its ROM might ask for a memory image to be sent to it as a single datagram. By marking the datagram with the *DF* bit, the sender knows it will arrive in one piece, even if this means that the datagram must avoid a small-packet network on the best path and take a suboptimal route. All machines are required to accept fragments of 576 bytes or less.

MF stands for More Fragments. All fragments except the last one have this bit set. It is needed to know when all fragments of a datagram have arrived.

The *Fragment offset* tells where in the current datagram this fragment belongs. All fragments except the last one in a datagram must be a multiple of 8 bytes, the elementary fragment unit. Since 13 bits are provided, there is a maximum of 8192 fragments per datagram, giving a maximum datagram length of 65,536 bytes, one more than the *Total length* field.

The *Time to live* field is a counter used to limit packet lifetimes. It is supposed to count time in seconds, allowing a maximum lifetime of 255 sec. It must be decremented on each hop and is supposed to be decremented multiple times when queued for a long time in a router. In practice, it just counts hops. When it hits zero, the packet is discarded and a warning packet is sent back to the source host. This feature prevents datagrams from wandering around forever, something that otherwise might happen if the routing tables ever become corrupted.

When the network layer has assembled a complete datagram, it needs to know what to do with it. The *Protocol* field tells it which transport process to give it to. TCP is one possibility, but so are UDP and some others. The numbering of protocols is global across the entire Internet. Protocols and other assigned numbers were formerly listed in RFC 1700, but nowadays they are contained in an on-line data base located at www.iana.org.

The *Header checksum* verifies the header only. Such a checksum is useful for detecting errors generated by bad memory words inside a router. The algorithm is to add up all the 16-bit halfwords as they arrive, using one's complement arithmetic and then take the one's complement of the result. For purposes of this algorithm, the *Header checksum* is assumed to be zero upon arrival. This algorithm is more robust than using a normal add. Note that the *Header checksum* must be recomputed at each hop because at least one field always changes (the *Time to live* field), but tricks can be used to speed up the computation.

The *Source address* and *Destination address* indicate the network number and host number. We will discuss Internet addresses in the next section. The *Options* field was designed to provide an escape to allow subsequent versions of the protocol to include information not present in the original design, to permit experimenters to try out new ideas, and to avoid allocating header bits to information that is rarely needed. The options are variable length. Each begins with a 1-byte code identifying the option. Some options are followed by a 1-byte option length

field, and then one or more data bytes. The *Options* field is padded out to a multiple of four bytes. Originally, five options were defined, as listed in Fig. 5-54, but since then some new ones have been added. The current complete list is now maintained on-line at www.iana.org/assignments/ip-parameters.

Option	Description
Security	Specifies how secret the datagram is
Strict source routing	Gives the complete path to be followed
Loose source routing	Gives a list of routers not to be missed
Record route	Makes each router append its IP address
Timestamp	Makes each router append its address and timestamp

Figure 5-54. Some of the IP options.

The *Security* option tells how secret the information is. In theory, a military router might use this field to specify not to route through certain countries the military considers to be “bad guys.” In practice, all routers ignore it, so its only practical function is to help spies find the good stuff more easily.

The *Strict source routing* option gives the complete path from source to destination as a sequence of IP addresses. The datagram is required to follow that exact route. It is most useful for system managers to send emergency packets when the routing tables are corrupted, or for making timing measurements.

The *Loose source routing* option requires the packet to traverse the list of routers specified, and in the order specified, but it is allowed to pass through other routers on the way. Normally, this option would only provide a few routers, to force a particular path. For example, to force a packet from London to Sydney to go west instead of east, this option might specify routers in New York, Los Angeles, and Honolulu. This option is most useful when political or economic considerations dictate passing through or avoiding certain countries.

The *Record route* option tells the routers along the path to append their IP address to the option field. This allows system managers to track down bugs in the routing algorithms (“Why are packets from Houston to Dallas visiting Tokyo first?”). When the ARPANET was first set up, no packet ever passed through more than nine routers, so 40 bytes of option was ample. As mentioned above, now it is too small.

Finally, the *Timestamp* option is like the *Record route* option, except that in addition to recording its 32-bit IP address, each router also records a 32-bit timestamp. This option, too, is mostly for debugging routing algorithms.

5.6.2 IP Addresses

Every host and router on the Internet has an IP address, which encodes its network number and host number. The combination is unique: in principle, no two machines on the Internet have the same IP address. All IP addresses are 32 bits

long and are used in the *Source address* and *Destination address* fields of IP packets. It is important to note that an IP address does not actually refer to a host. It really refers to a network interface, so if a host is on two networks, it must have two IP addresses. However, in practice, most hosts are on one network and thus have one IP address.

For several decades, IP addresses were divided into the five categories listed in Fig. 5-55. This allocation has come to be called **classful addressing**. It is no longer used, but references to it in the literature are still common. We will discuss the replacement of classful addressing shortly.

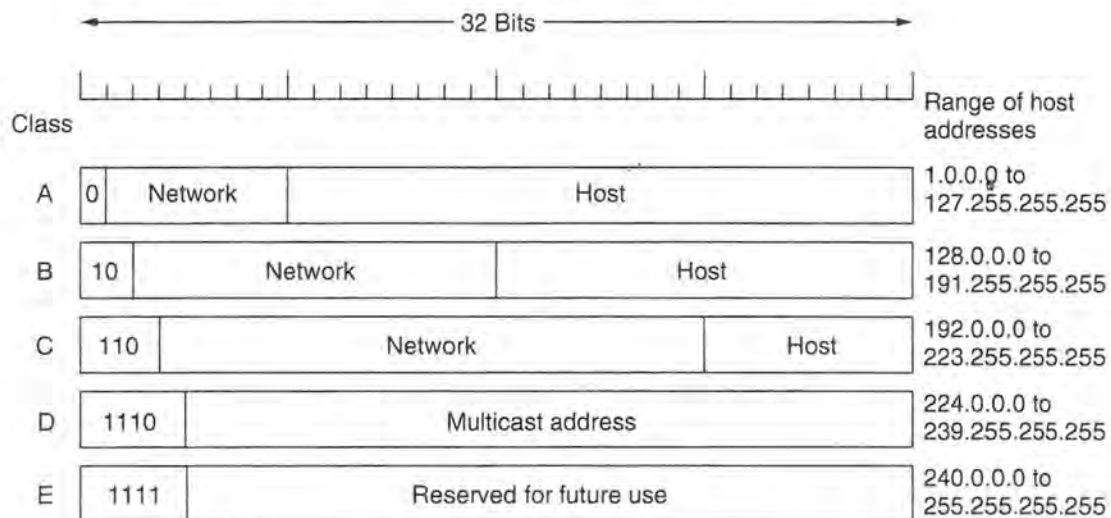


Figure 5-55. IP address formats.

The class A, B, C, and D formats allow for up to 128 networks with 16 million hosts each, 16,384 networks with up to 64K hosts, and 2 million networks (e.g., LANs) with up to 256 hosts each (although a few of these are special). Also supported is multicast, in which a datagram is directed to multiple hosts. Addresses beginning with 1111 are reserved for future use. Over 500,000 networks are now connected to the Internet, and the number grows every year. Network numbers are managed by a nonprofit corporation called **ICANN (Internet Corporation for Assigned Names and Numbers)** to avoid conflicts. In turn, ICANN has delegated parts of the address space to various regional authorities, which then dole out IP addresses to ISPs and other companies.

Network addresses, which are 32-bit numbers, are usually written in **dotted decimal notation**. In this format, each of the 4 bytes is written in decimal, from 0 to 255. For example, the 32-bit hexadecimal address C0290614 is written as 192.41.6.20. The lowest IP address is 0.0.0.0 and the highest is 255.255.255.255.

The values 0 and -1 (all 1s) have special meanings, as shown in Fig. 5-56. The value 0 means this network or this host. The value of -1 is used as a broadcast address to mean all hosts on the indicated network.

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																																This host		
0 0				...																0 0												Host	A host on this network	
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1																																Broadcast on the local network		
Network																1 1 1 1				...										1 1 1 1				Broadcast on a distant network
127								(Anything)																								Loopback		

Figure 5-56. Special IP addresses.

The IP address 0.0.0.0 is used by hosts when they are being booted. IP addresses with 0 as network number refer to the current network. These addresses allow machines to refer to their own network without knowing its number (but they have to know its class to know how many 0s to include). The address consisting of all 1s allows broadcasting on the local network, typically a LAN. The addresses with a proper network number and all 1s in the host field allow machines to send broadcast packets to distant LANs anywhere in the Internet (although many network administrators disable this feature). Finally, all addresses of the form 127.xx.yy.zz are reserved for loopback testing. Packets sent to that address are not put out onto the wire; they are processed locally and treated as incoming packets. This allows packets to be sent to the local network without the sender knowing its number.

Subnets

As we have seen, all the hosts in a network must have the same network number. This property of IP addressing can cause problems as networks grow. For example, consider a university that started out with one class B network used by the Computer Science Dept. for the computers on its Ethernet. A year later, the Electrical Engineering Dept. wanted to get on the Internet, so they bought a repeater to extend the CS Ethernet to their building. As time went on, many other departments acquired computers and the limit of four repeaters per Ethernet was quickly reached. A different organization was required.

Getting a second network address would be hard to do since network addresses are scarce and the university already had enough addresses for over 60,000 hosts. The problem is the rule that a single class A, B, or C address refers to one network, not to a collection of LANs. As more and more organizations ran into this situation, a small change was made to the addressing system to deal with it.

The solution is to allow a network to be split into several parts for internal use but still act like a single network to the outside world. A typical campus network

nowadays might look like that of Fig. 5-57, with a main router connected to an ISP or regional network and numerous Ethernets spread around campus in different departments. Each of the Ethernets has its own router connected to the main router (possibly via a backbone LAN, but the nature of the interrouter connection is not relevant here).

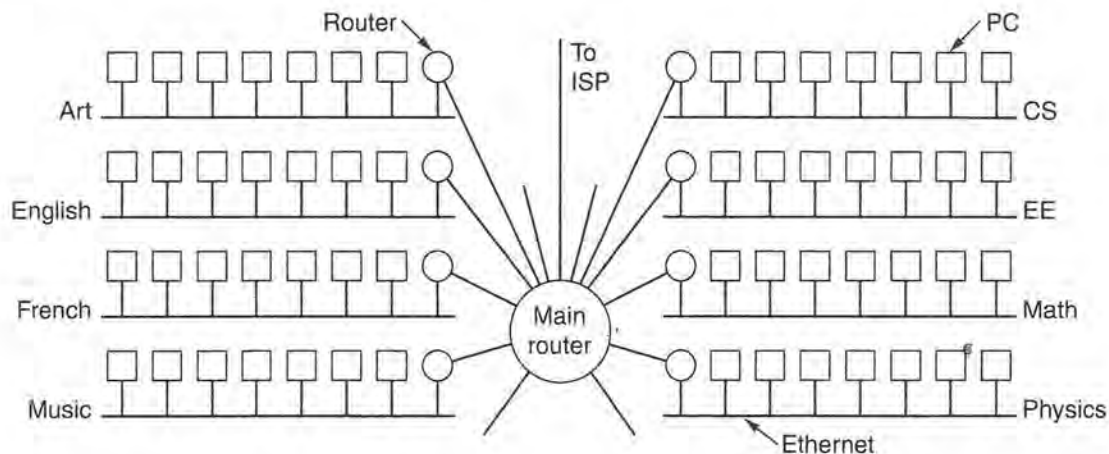


Figure 5-57. A campus network consisting of LANs for various departments.

In the Internet literature, the parts of the network (in this case, Ethernets) are called **subnets**. As we mentioned in Chap. 1, this usage conflicts with “subnet” to mean the set of all routers and communication lines in a network. Hopefully, it will be clear from the context which meaning is intended. In this section and the next one, the new definition will be the one used exclusively.

When a packet comes into the main router, how does it know which subnet (Ethernet) to give it to? One way would be to have a table with 65,536 entries in the main router telling which router to use for each host on campus. This idea would work, but it would require a very large table in the main router and a lot of manual maintenance as hosts were added, moved, or taken out of service.

Instead, a different scheme was invented. Basically, instead of having a single class B address with 14 bits for the network number and 16 bits for the host number, some bits are taken away from the host number to create a subnet number. For example, if the university has 35 departments, it could use a 6-bit subnet number and a 10-bit host number, allowing for up to 64 Ethernets, each with a maximum of 1022 hosts (0 and -1 are not available, as mentioned earlier). This split could be changed later if it turns out to be the wrong one.

To implement subnetting, the main router needs a **subnet mask** that indicates the split between network + subnet number and host, as shown in Fig. 5-58. Subnet masks are also written in dotted decimal notation, with the addition of a slash followed by the number of bits in the network + subnet part. For the example of Fig. 5-58, the subnet mask can be written as 255.255.252.0. An alternative notation is /22 to indicate that the subnet mask is 22 bits long.

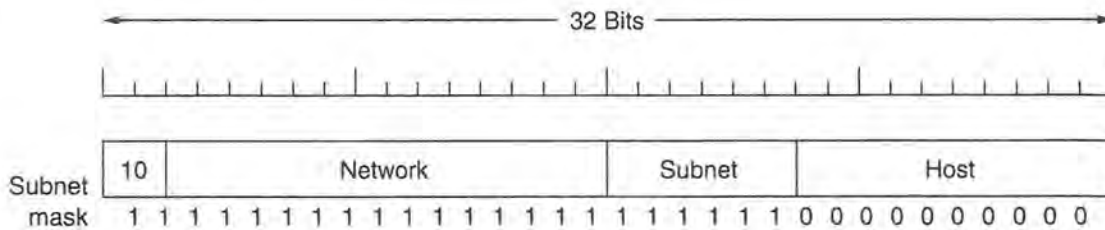


Figure 5-58. A class B network subnetted into 64 subnets.

Outside the network, the subnetting is not visible, so allocating a new subnet does not require contacting ICANN or changing any external databases. In this example, the first subnet might use IP addresses starting at 130.50.4.1; the second subnet might start at 130.50.8.1; the third subnet might start at 130.50.12.1; and so on. To see why the subnets are counting by fours, note that the corresponding binary addresses are as follows:

```
Subnet 1: 10000010  00110010  000001|00  00000001
Subnet 2: 10000010  00110010  000010|00  00000001
Subnet 3: 10000010  00110010  000011|00  00000001
```

Here the vertical bar (|) shows the boundary between the subnet number and the host number. To its left is the 6-bit subnet number; to its right is the 10-bit host number.

To see how subnets work, it is necessary to explain how IP packets are processed at a router. Each router has a table listing some number of (network, 0) IP addresses and some number of (this-network, host) IP addresses. The first kind tells how to get to distant networks. The second kind tells how to get to local hosts. Associated with each table is the network interface to use to reach the destination, and certain other information.

When an IP packet arrives, its destination address is looked up in the routing table. If the packet is for a distant network, it is forwarded to the next router on the interface given in the table. If it is a local host (e.g., on the router's LAN), it is sent directly to the destination. If the network is not present, the packet is forwarded to a default router with more extensive tables. This algorithm means that each router only has to keep track of other networks and local hosts, not (network, host) pairs, greatly reducing the size of the routing table.

When subnetting is introduced, the routing tables are changed, adding entries of the form (this-network, subnet, 0) and (this-network, this-subnet, host). Thus, a router on subnet k knows how to get to all the other subnets and also how to get to all the hosts on subnet k . It does not have to know the details about hosts on other subnets. In fact, all that needs to be changed is to have each router do a Boolean AND with the network's subnet mask to get rid of the host number and look up the resulting address in its tables (after determining which network class it is).

For example, a packet addressed to 130.50.15.6 and arriving at the main router is ANDed with the subnet mask 255.255.252.0/22 to give the address 130.50.12.0. This address is looked up in the routing tables to find out which output line to use to get to the router for subnet 3. Subnetting thus reduces router table space by creating a three-level hierarchy consisting of network, subnet, and host.

CIDR—Classless InterDomain Routing

IP has been in heavy use for decades. It has worked extremely well, as demonstrated by the exponential growth of the Internet. Unfortunately, IP is rapidly becoming a victim of its own popularity: it is running out of addresses. This looming disaster has sparked a great deal of discussion and controversy within the Internet community about what to do about it. In this section we will describe both the problem and several proposed solutions.

Back in 1987, a few visionaries predicted that some day the Internet might grow to 100,000 networks. Most experts pooh-poohed this as being decades in the future, if ever. The 100,000th network was connected in 1996. The problem, as mentioned above, is that the Internet is rapidly running out of IP addresses. In principle, over 2 billion addresses exist, but the practice of organizing the address space by classes (see Fig. 5-55) wastes millions of them. In particular, the real villain is the class B network. For most organizations, a class A network, with 16 million addresses is too big, and a class C network, with 256 addresses is too small. A class B network, with 65,536, is just right. In Internet folklore, this situation is known as the **three bears problem** (as in *Goldilocks and the Three Bears*).

In reality, a class B address is far too large for most organizations. Studies have shown that more than half of all class B networks have fewer than 50 hosts. A class C network would have done the job, but no doubt every organization that asked for a class B address thought that one day it would outgrow the 8-bit host field. In retrospect, it might have been better to have had class C networks use 10 bits instead of eight for the host number, allowing 1022 hosts per network. Had this been the case, most organizations would have probably settled for a class C network, and there would have been half a million of them (versus only 16,384 class B networks).

It is hard to fault the Internet designers for not having provided more (and smaller) class B addresses. At the time the decision was made to create the three classes, the Internet was a research network connecting the major research universities in the U.S. (plus a very small number of companies and military sites doing networking research). No one then perceived the Internet as becoming a mass market communication system rivaling the telephone network. At the time, someone no doubt said: "The U.S. has about 2000 colleges and universities. Even if all of them connect to the Internet and many universities in other countries join, too, we are never going to hit 16,000 since there are not that many universities in

the whole world. Furthermore, having the host number be an integral number of bytes speeds up packet processing.”

However, if the split had allocated 20 bits to the class B network number, another problem would have emerged: the routing table explosion. From the point of view of the routers, the IP address space is a two-level hierarchy, with network numbers and host numbers. Routers do not have to know about all the hosts, but they do have to know about all the networks. If half a million class C networks were in use, every router in the entire Internet would need a table with half a million entries, one per network, telling which line to use to get to that network, as well as providing other information.

The actual physical storage of half a million entry tables is probably doable, although expensive for critical routers that keep the tables in static RAM on I/O boards. A more serious problem is that the complexity of various algorithms relating to management of the tables grows faster than linear. Worse yet, much of the existing router software and firmware was designed at a time when the Internet had 1000 connected networks and 10,000 networks seemed decades away. Design choices made then often are far from optimal now.

In addition, various routing algorithms require each router to transmit its tables periodically (e.g., distance vector protocols). The larger the tables, the more likely it is that some parts will get lost underway, leading to incomplete data at the other end and possibly routing instabilities.

The routing table problem could have been solved by going to a deeper hierarchy. For example, having each IP address contain a country, state/province, city, network, and host field might work. Then each router would only need to know how to get to each country, the states or provinces in its own country, the cities in its state or province, and the networks in its city. Unfortunately, this solution would require considerably more than 32 bits for IP addresses and would use addresses inefficiently (Liechtenstein would have as many bits as the United States).

In short, some solutions solve one problem but create a new one. The solution that was implemented and that gave the Internet a bit of extra breathing room is **CIDR (Classless InterDomain Routing)**. The basic idea behind CIDR, which is described in RFC 1519, is to allocate the remaining IP addresses in variable-sized blocks, without regard to the classes. If a site needs, say, 2000 addresses, it is given a block of 2048 addresses on a 2048-byte boundary.

Dropping the classes makes forwarding more complicated. In the old classful system, forwarding worked like this. When a packet arrived at a router, a copy of the IP address was shifted right 28 bits to yield a 4-bit class number. A 16-way branch then sorted packets into A, B, C, and D (if supported), with eight of the cases for class A, four of the cases for class B, two of the cases for class C, and one each for D and E. The code for each class then masked off the 8-, 16-, or 24-bit network number and right aligned it in a 32-bit word. The network number was then looked up in the A, B, or C table, usually by indexing for A and B net-

works and hashing for C networks. Once the entry was found, the outgoing line could be looked up and the packet forwarded.

With CIDR, this simple algorithm no longer works. Instead, each routing table entry is extended by giving it a 32-bit mask. Thus, there is now a single routing table for all networks consisting of an array of (IP address, subnet mask, outgoing line) triples. When a packet comes in, its destination IP address is first extracted. Then (conceptually) the routing table is scanned entry by entry, masking the destination address and comparing it to the table entry looking for a match. It is possible that multiple entries (with different subnet mask lengths) match, in which case the longest mask is used. Thus, if there is a match for a /20 mask and a /24 mask, the /24 entry is used.

Complex algorithms have been devised to speed up the address matching process (Ruiz-Sanchez et al., 2001). Commercial routers use custom VLSI chips with these algorithms embedded in hardware.

To make the forwarding algorithm easier to understand, let us consider an example in which millions of addresses are available starting at 194.24.0.0. Suppose that Cambridge University needs 2048 addresses and is assigned the addresses 194.24.0.0 through 194.24.7.255, along with mask 255.255.248.0. Next, Oxford University asks for 4096 addresses. Since a block of 4096 addresses must lie on a 4096-byte boundary, they cannot be given addresses starting at 194.24.8.0. Instead, they get 194.24.16.0 through 194.24.31.255 along with subnet mask 255.255.240.0. Now the University of Edinburgh asks for 1024 addresses and is assigned addresses 194.24.8.0 through 194.24.11.255 and mask 255.255.252.0. These assignments are summarized in Fig. 5-59.

University	First address	Last address	How many	Written as
Cambridge	194.24.0.0	194.24.7.255	2048	194.24.0.0/21
Edinburgh	194.24.8.0	194.24.11.255	1024	194.24.8.0/22
(Available)	194.24.12.0	194.24.15.255	1024	194.24.12/22
Oxford	194.24.16.0	194.24.31.255	4096	194.24.16.0/20

Figure 5-59. A set of IP address assignments.

The routing tables all over the world are now updated with the three assigned entries. Each entry contains a base address and a subnet mask. These entries (in binary) are:

Address	Mask
C: 11000010 00011000 00000000 00000000	11111111 11111111 11111000 00000000
E: 11000010 00011000 00001000 00000000	11111111 11111111 11111100 00000000
O: 11000010 00011000 00010000 00000000	11111111 11111111 11110000 00000000

Now consider what happens when a packet comes in addressed to 194.24.17.4, which in binary is represented as the following 32-bit string

```
11000010 00011000 00010001 00000100
```

First it is Boolean ANDed with the Cambridge mask to get

```
11000010 00011000 00010000 00000000
```

This value does not match the Cambridge base address, so the original address is next ANDed with the Edinburgh mask to get

```
11000010 00011000 00010000 00000000
```

This value does not match the Edinburgh base address, so Oxford is tried next, yielding

```
11000010 00011000 00010000 00000000
```

This value does match the Oxford base. If no longer matches are found farther down the table, the Oxford entry is used and the packet is sent along the line named in it.

Now let us look at these three universities from the point of view of a router in Omaha, Nebraska, that has only four outgoing lines: Minneapolis, New York, Dallas, and Denver. When the router software there gets the three new entries, it notices that it can combine all three entries into a single **aggregate entry** 194.24.0.0/19 with a binary address and submask as follows:

```
11000010 00000000 00000000 00000000 11111111 11111111 11100000 00000000
```

This entry sends all packets destined for any of the three universities to New York. By aggregating the three entries, the Omaha router has reduced its table size by two entries.

If New York has a single line to London for all U.K. traffic, it can use an aggregated entry as well. However, if it has separate lines for London and Edinburgh, then it has to have three separate entries. Aggregation is heavily used throughout the Internet to reduce the size of the router tables.

As a final note on this example, the aggregate route entry in Omaha also sends packets for the unassigned addresses to New York. As long as the addresses are truly unassigned, this does not matter because they are not supposed to occur. However, if they are later assigned to a company in California, an additional entry, 194.24.12.0/22, will be needed to deal with them.

NAT—Network Address Translation

IP addresses are scarce. An ISP might have a /16 (formerly class B) address, giving it 65,534 host numbers. If it has more customers than that, it has a problem. For home customers with dial-up connections, one way around the problem is to dynamically assign an IP address to a computer when it calls up and logs in and take the IP address back when the session ends. In this way, a single /16

address can handle up to 65,534 active users, which is probably good enough for an ISP with several hundred thousand customers. When the session is terminated, the IP address is reassigned to another caller. While this strategy works well for an ISP with a moderate number of home users, it fails for ISPs that primarily serve business customers.

The problem is that business customers expect to be on-line continuously during business hours. Both small businesses, such as three-person travel agencies, and large corporations have multiple computers connected by a LAN. Some computers are employee PCs; others may be Web servers. Generally, there is a router on the LAN that is connected to the ISP by a leased line to provide continuous connectivity. This arrangement means that each computer must have its own IP address all day long. In effect, the total number of computers owned by all its business customers combined cannot exceed the number of IP addresses the ISP has. For a /16 address, this limits the total number of computers to 65,534. For an ISP with tens of thousands of business customers, this limit will quickly be exceeded.

To make matters worse, more and more home users are subscribing to ADSL or Internet over cable. Two of the features of these services are (1) the user gets a permanent IP address and (2) there is no connect charge (just a monthly flat rate charge), so many ADSL and cable users just stay logged in permanently. This development just adds to the shortage of IP addresses. Assigning IP addresses on-the-fly as is done with dial-up users is of no use because the number of IP addresses in use at any one instant may be many times the number the ISP owns.

And just to make it a bit more complicated, many ADSL and cable users have two or more computers at home, often one for each family member, and they all want to be on-line all the time using the single IP address their ISP has given them. The solution here is to connect all the PCs via a LAN and put a router on it. From the ISP's point of view, the family is now the same as a small business with a handful of computers. Welcome to Jones, Inc.

The problem of running out of IP addresses is not a theoretical problem that might occur at some point in the distant future. It is happening right here and right now. The long-term solution is for the whole Internet to migrate to IPv6, which has 128-bit addresses. This transition is slowly occurring, but it will be years before the process is complete. As a consequence, some people felt that a quick fix was needed for the short term. This quick fix came in the form of **NAT (Network Address Translation)**, which is described in RFC 3022 and which we will summarize below. For additional information, see (Dutcher, 2001).

The basic idea behind NAT is to assign each company a single IP address (or at most, a small number of them) for Internet traffic. *Within* the company, every computer gets a unique IP address, which is used for routing intramural traffic. However, when a packet exits the company and goes to the ISP, an address translation takes place. To make this scheme possible, three ranges of IP addresses have been declared as private. Companies may use them internally as they wish.

The only rule is that no packets containing these addresses may appear on the Internet itself. The three reserved ranges are:

10.0.0.0	– 10.255.255.255/8	(16,777,216 hosts)
172.16.0.0	– 172.31.255.255/12	(1,048,576 hosts)
192.168.0.0	– 192.168.255.255/16	(65,536 hosts)

The first range provides for 16,777,216 addresses (except for 0 and –1, as usual) and is the usual choice of most companies, even if they do not need so many addresses.

The operation of NAT is shown in Fig. 5-60. Within the company premises, every machine has a unique address of the form 10.x.y.z. However, when a packet leaves the company premises, it passes through a **NAT box** that converts the internal IP source address, 10.0.0.1 in the figure, to the company's true IP address, 198.60.42.12 in this example. The NAT box is often combined in a single device with a firewall, which provides security by carefully controlling what goes into the company and what comes out. We will study firewalls in Chap. 8. It is also possible to integrate the NAT box into the company's router.

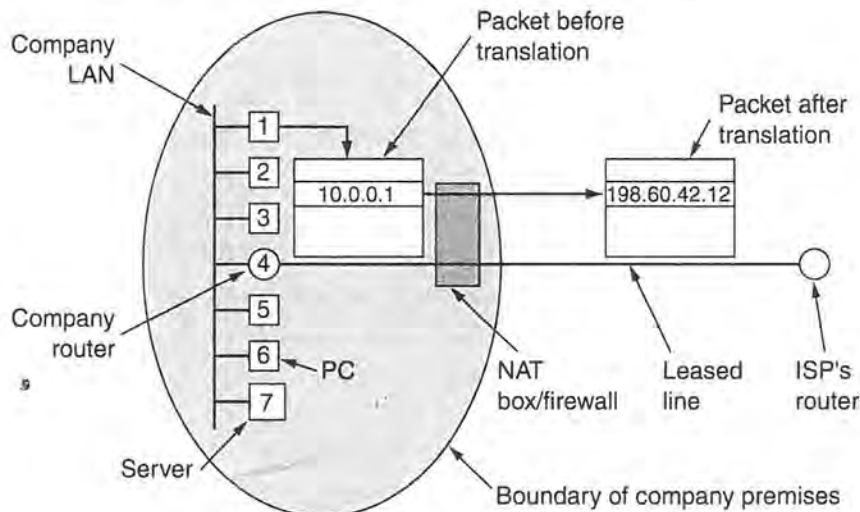


Figure 5-60. Placement and operation of a NAT box.

So far we have glossed over one tiny little detail: when the reply comes back (e.g., from a Web server), it is naturally addressed to 198.60.42.12, so how does the NAT box know which address to replace it with? Herein lies the problem with NAT. If there were a spare field in the IP header, that field could be used to keep track of who the real sender was, but only 1 bit is still unused. In principle, a new option could be created to hold the true source address, but doing so would require changing the IP code on all the machines on the entire Internet to handle the new option. This is not a promising alternative for a quick fix.

What actually happened is as follows. The NAT designers observed that most IP packets carry either TCP or UDP payloads. When we study TCP and UDP in Chap. 6, we will see that both of these have headers containing a source port and a destination port. Below we will just discuss TCP ports, but exactly the same story holds for UDP ports. The ports are 16-bit integers that indicate where the TCP connection begins and ends. These ports provide the field needed to make NAT work.

When a process wants to establish a TCP connection with a remote process, it attaches itself to an unused TCP port on its own machine. This is called the **source port** and tells the TCP code where to send incoming packets belonging to this connection. The process also supplies a **destination port** to tell who to give the packets to on the remote side. Ports 0–1023 are reserved for well-known services. For example, port 80 is the port used by Web servers, so remote clients can locate them. Each outgoing TCP message contains both a source port and a destination port. Together, these ports serve to identify the processes using the connection on both ends.

An analogy may make the use of ports clearer. Imagine a company with a single main telephone number. When people call the main number, they reach an operator who asks which extension they want and then puts them through to that extension. The main number is analogous to the company's IP address and the extensions on both ends are analogous to the ports. Ports are an extra 16-bits of addressing that identify which process gets which incoming packet.

Using the *Source port* field, we can solve our mapping problem. Whenever an outgoing packet enters the NAT box, the 10.x.y.z source address is replaced by the company's true IP address. In addition, the TCP *Source port* field is replaced by an index into the NAT box's 65,536-entry translation table. This table entry contains the original IP address and the original source port. Finally, both the IP and TCP header checksums are recomputed and inserted into the packet. It is necessary to replace the *Source port* because connections from machines 10.0.0.1 and 10.0.0.2 may both happen to use port 5000, for example, so the *Source port* alone is not enough to identify the sending process.

When a packet arrives at the NAT box from the ISP, the *Source port* in the TCP header is extracted and used as an index into the NAT box's mapping table. From the entry located, the internal IP address and original TCP *Source port* are extracted and inserted into the packet. Then both the IP and TCP checksums are recomputed and inserted into the packet. The packet is then passed to the company router for normal delivery using the 10.x.y.z address.

NAT can also be used to alleviate the IP shortage for ADSL and cable users. When the ISP assigns each user an address, it uses 10.x.y.z addresses. When packets from user machines exit the ISP and enter the main Internet, they pass through a NAT box that translates them to the ISP's true Internet address. On the way back, packets undergo the reverse mapping. In this respect, to the rest of the Internet, the ISP and its home ADSL/cable users just looks like a big company.

Although this scheme sort of solves the problem, many people in the IP community regard it as an abomination-on-the-face-of-the-earth. Briefly summarized, here are some of the objections. First, NAT violates the architectural model of IP, which states that every IP address uniquely identifies a single machine worldwide. The whole software structure of the Internet is built on this fact. With NAT, thousands of machines may (and do) use address 10.0.0.1.

Second, NAT changes the Internet from a connectionless network to a kind of connection-oriented network. The problem is that the NAT box must maintain information (the mapping) for each connection passing through it. Having the network maintain connection state is a property of connection-oriented networks, not connectionless ones. If the NAT box crashes and its mapping table is lost, all its TCP connections are destroyed. In the absence of NAT, router crashes have no effect on TCP. The sending process just times out within a few seconds and retransmits all unacknowledged packets. With NAT, the Internet becomes as vulnerable as a circuit-switched network.

Third, NAT violates the most fundamental rule of protocol layering: layer k may not make any assumptions about what layer $k + 1$ has put into the payload field. This basic principle is there to keep the layers independent. If TCP is later upgraded to TCP-2, with a different header layout (e.g., 32-bit ports), NAT will fail. The whole idea of layered protocols is to ensure that changes in one layer do not require changes in other layers. NAT destroys this independence.

Fourth, processes on the Internet are not required to use TCP or UDP. If a user on machine *A* decides to use some new transport protocol to talk to a user on machine *B* (for example, for a multimedia application), introduction of a NAT box will cause the application to fail because the NAT box will not be able to locate the TCP *Source port* correctly.

Fifth, some applications insert IP addresses in the body of the text. The receiver then extracts these addresses and uses them. Since NAT knows nothing about these addresses, it cannot replace them, so any attempt to use them on the remote side will fail. **FTP**, the standard **File Transfer Protocol** works this way and can fail in the presence of NAT unless special precautions are taken. Similarly, the H.323 Internet telephony protocol (which we will study in Chap. 7) has this property and can fail in the presence of NAT. It may be possible to patch NAT to work with H.323, but having to patch the code in the NAT box every time a new application comes along is not a good idea.

Sixth, since the TCP *Source port* field is 16 bits, at most 65,536 machines can be mapped onto an IP address. Actually, the number is slightly less because the first 4096 ports are reserved for special uses. However, if multiple IP addresses are available, each one can handle up to 61,440 machines.

These and other problems with NAT are discussed in RFC 2993. In general, the opponents of NAT say that by fixing the problem of insufficient IP addresses with a temporary and ugly hack, the pressure to implement the real solution, that is, the transition to IPv6, is reduced, and this is a bad thing.

5.6.3 Internet Control Protocols

In addition to IP, which is used for data transfer, the Internet has several control protocols used in the network layer, including ICMP, ARP, RARP, BOOTP, and DHCP. In this section we will look at each of these in turn.

The Internet Control Message Protocol

The operation of the Internet is monitored closely by the routers. When something unexpected occurs, the event is reported by the **ICMP (Internet Control Message Protocol)**, which is also used to test the Internet. About a dozen types of ICMP messages are defined. The most important ones are listed in Fig. 5-61. Each ICMP message type is encapsulated in an IP packet.

Message type	Description
Destination unreachable	Packet could not be delivered
Time exceeded	Time to live field hit 0
Parameter problem	Invalid header field
Source quench	Choke packet
Redirect	Teach a router about geography
Echo	Ask a machine if it is alive
Echo reply	Yes, I am alive
Timestamp request	Same as Echo request, but with timestamp
Timestamp reply	Same as Echo reply, but with timestamp

Figure 5-61. The principal ICMP message types.

The **DESTINATION UNREACHABLE** message is used when the subnet or a router cannot locate the destination or when a packet with the *DF* bit cannot be delivered because a “small-packet” network stands in the way.

The **TIME EXCEEDED** message is sent when a packet is dropped because its counter has reached zero. This event is a symptom that packets are looping, that there is enormous congestion, or that the timer values are being set too low.

The **PARAMETER PROBLEM** message indicates that an illegal value has been detected in a header field. This problem indicates a bug in the sending host’s IP software or possibly in the software of a router transited.

The **SOURCE QUENCH** message was formerly used to throttle hosts that were sending too many packets. When a host received this message, it was expected to slow down. It is rarely used any more because when congestion occurs, these packets tend to add more fuel to the fire. Congestion control in the Internet is now done largely in the transport layer; we will study it in detail in Chap. 6.

The REDIRECT message is used when a router notices that a packet seems to be routed wrong. It is used by the router to tell the sending host about the probable error.

The ECHO and ECHO REPLY messages are used to see if a given destination is reachable and alive. Upon receiving the ECHO message, the destination is expected to send an ECHO REPLY message back. The TIMESTAMP REQUEST and TIMESTAMP REPLY messages are similar, except that the arrival time of the message and the departure time of the reply are recorded in the reply. This facility is used to measure network performance.

In addition to these messages, others have been defined. The on-line list is now kept at www.iana.org/assignments/icmp-parameters.

ARP—The Address Resolution Protocol

Although every machine on the Internet has one (or more) IP addresses, these cannot actually be used for sending packets because the data link layer hardware does not understand Internet addresses. Nowadays, most hosts at companies and universities are attached to a LAN by an interface board that only understands LAN addresses. For example, every Ethernet board ever manufactured comes equipped with a 48-bit Ethernet address. Manufacturers of Ethernet boards request a block of addresses from a central authority to ensure that no two boards have the same address (to avoid conflicts should the two boards ever appear on the same LAN). The boards send and receive frames based on 48-bit Ethernet addresses. They know nothing at all about 32-bit IP addresses.

The question now arises: How do IP addresses get mapped onto data link layer addresses, such as Ethernet? To explain how this works, let us use the example of Fig. 5-62, in which a small university with several class C (now called /24) networks is illustrated. Here we have two Ethernets, one in the Computer Science Dept., with IP address 192.31.65.0 and one in Electrical Engineering, with IP address 192.31.63.0. These are connected by a campus backbone ring (e.g., FDDI) with IP address 192.31.60.0. Each machine on an Ethernet has a unique Ethernet address, labeled *E1* through *E6*, and each machine on the FDDI ring has an FDDI address, labeled *F1* through *F3*.

Let us start out by seeing how a user on host 1 sends a packet to a user on host 2. Let us assume the sender knows the name of the intended receiver, possibly something like *mary@eagle.cs.uni.edu*. The first step is to find the IP address for host 2, known as *eagle.cs.uni.edu*. This lookup is performed by the Domain Name System, which we will study in Chap. 7. For the moment, we will just assume that DNS returns the IP address for host 2 (192.31.65.5).

The upper layer software on host 1 now builds a packet with 192.31.65.5 in the *Destination address* field and gives it to the IP software to transmit. The IP software can look at the address and see that the destination is on its own network,

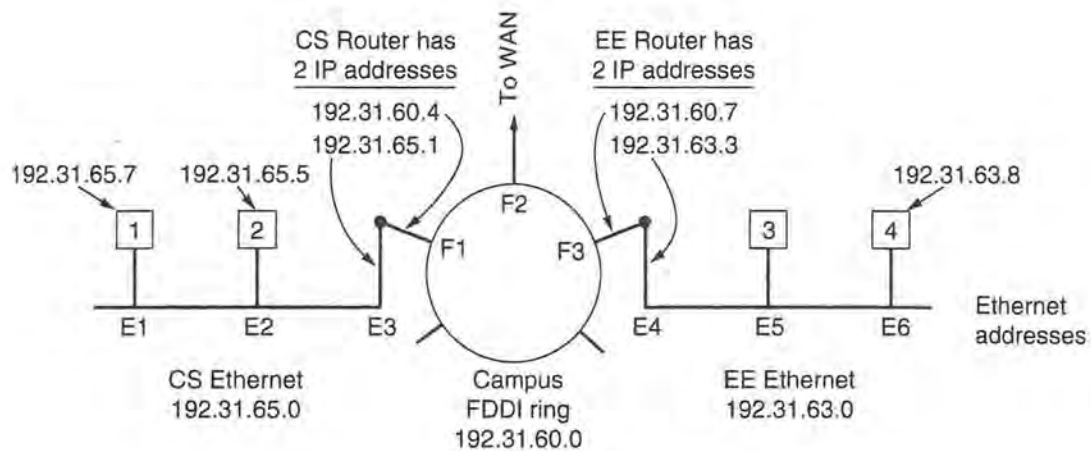


Figure 5-62. Three interconnected /24 networks: two Ethernet and an FDDI ring.

but it needs some way to find the destination's Ethernet address. One solution is to have a configuration file somewhere in the system that maps IP addresses onto Ethernet addresses. While this solution is certainly possible, for organizations with thousands of machines, keeping all these files up to date is an error-prone, time-consuming job.

A better solution is for host 1 to output a broadcast packet onto the Ethernet asking: Who owns IP address 192.31.65.5? The broadcast will arrive at every machine on Ethernet 192.31.65.0, and each one will check its IP address. Host 2 alone will respond with its Ethernet address (*E2*). In this way host 1 learns that IP address 192.31.65.5 is on the host with Ethernet address *E2*. The protocol used for asking this question and getting the reply is called **ARP (Address Resolution Protocol)**. Almost every machine on the Internet runs it. ARP is defined in RFC 826.

The advantage of using ARP over configuration files is the simplicity. The system manager does not have to do much except assign each machine an IP address and decide about subnet masks. ARP does the rest.

At this point, the IP software on host 1 builds an Ethernet frame addressed to *E2*, puts the IP packet (addressed to 192.31.65.5) in the payload field, and dumps it onto the Ethernet. The Ethernet board of host 2 detects this frame, recognizes it as a frame for itself, scoops it up, and causes an interrupt. The Ethernet driver extracts the IP packet from the payload and passes it to the IP software, which sees that it is correctly addressed and processes it.

Various optimizations are possible to make ARP work more efficiently. To start with, once a machine has run ARP, it caches the result in case it needs to contact the same machine shortly. Next time it will find the mapping in its own cache, thus eliminating the need for a second broadcast. In many cases host 2 will need to send back a reply, forcing it, too, to run ARP to determine the sender's

Ethernet address. This ARP broadcast can be avoided by having host 1 include its IP-to-Ethernet mapping in the ARP packet. When the ARP broadcast arrives at host 2, the pair (192.31.65.7, E1) is entered into host 2's ARP cache for future use. In fact, all machines on the Ethernet can enter this mapping into their ARP caches.

Yet another optimization is to have every machine broadcast its mapping when it boots. This broadcast is generally done in the form of an ARP looking for its own IP address. There should not be a response, but a side effect of the broadcast is to make an entry in everyone's ARP cache. If a response does (unexpectedly) arrive, two machines have been assigned the same IP address. The new one should inform the system manager and not boot.

To allow mappings to change, for example, when an Ethernet board breaks and is replaced with a new one (and thus a new Ethernet address), entries in the ARP cache should time out after a few minutes.

Now let us look at Fig. 5-62 again, only this time host 1 wants to send a packet to host 4 (192.31.63.8). Using ARP will fail because host 4 will not see the broadcast (routers do not forward Ethernet-level broadcasts). There are two solutions. First, the CS router could be configured to respond to ARP requests for network 192.31.63.0 (and possibly other local networks). In this case, host 1 will make an ARP cache entry of (192.31.63.8, E3) and happily send all traffic for host 4 to the local router. This solution is called **proxy ARP**. The second solution is to have host 1 immediately see that the destination is on a remote network and just send all such traffic to a default Ethernet address that handles all remote traffic, in this case E3. This solution does not require having the CS router know which remote networks it is serving.

Either way, what happens is that host 1 packs the IP packet into the payload field of an Ethernet frame addressed to E3. When the CS router gets the Ethernet frame, it removes the IP packet from the payload field and looks up the IP address in its routing tables. It discovers that packets for network 192.31.63.0 are supposed to go to router 192.31.60.7. If it does not already know the FDDI address of 192.31.60.7, it broadcasts an ARP packet onto the ring and learns that its ring address is F3. It then inserts the packet into the payload field of an FDDI frame addressed to F3 and puts it on the ring.

At the EE router, the FDDI driver removes the packet from the payload field and gives it to the IP software, which sees that it needs to send the packet to 192.31.63.8. If this IP address is not in its ARP cache, it broadcasts an ARP request on the EE Ethernet and learns that the destination address is E6, so it builds an Ethernet frame addressed to E6, puts the packet in the payload field, and sends it over the Ethernet. When the Ethernet frame arrives at host 4, the packet is extracted from the frame and passed to the IP software for processing.

Going from host 1 to a distant network over a WAN works essentially the same way, except that this time the CS router's tables tell it to use the WAN router whose FDDI address is F2.

RARP, BOOTP, and DHCP

ARP solves the problem of finding out which Ethernet address corresponds to a given IP address. Sometimes the reverse problem has to be solved: Given an Ethernet address, what is the corresponding IP address? In particular, this problem occurs when a diskless workstation is booted. Such a machine will normally get the binary image of its operating system from a remote file server. But how does it learn its IP address?

The first solution devised was to use **RARP (Reverse Address Resolution Protocol)** (defined in RFC 903). This protocol allows a newly-booted workstation to broadcast its Ethernet address and say: My 48-bit Ethernet address is 14.04.05.18.01.25. Does anyone out there know my IP address? The RARP server sees this request, looks up the Ethernet address in its configuration files, and sends back the corresponding IP address.

Using RARP is better than embedding an IP address in the memory image because it allows the same image to be used on all machines. If the IP address were buried inside the image, each workstation would need its own image.

A disadvantage of RARP is that it uses a destination address of all 1s (limited broadcasting) to reach the RARP server. However, such broadcasts are not forwarded by routers, so a RARP server is needed on each network. To get around this problem, an alternative bootstrap protocol called **BOOTP** was invented. Unlike RARP, BOOTP uses UDP messages, which are forwarded over routers. It also provides a diskless workstation with additional information, including the IP address of the file server holding the memory image, the IP address of the default router, and the subnet mask to use. BOOTP is described in RFCs 951, 1048, and 1084.

A serious problem with BOOTP is that it requires manual configuration of tables mapping IP address to Ethernet address. When a new host is added to a LAN, it cannot use BOOTP until an administrator has assigned it an IP address and entered its (Ethernet address, IP address) into the BOOTP configuration tables by hand. To eliminate this error-prone step, BOOTP was extended and given a new name: **DHCP (Dynamic Host Configuration Protocol)**. DHCP allows both manual IP address assignment and automatic assignment. It is described in RFCs 2131 and 2132. In most systems, it has largely replaced RARP and BOOTP.

Like RARP and BOOTP, DHCP is based on the idea of a special server that assigns IP addresses to hosts asking for one. This server need not be on the same LAN as the requesting host. Since the DHCP server may not be reachable by broadcasting, a **DHCP relay agent** is needed on each LAN, as shown in Fig. 5-63.

To find its IP address, a newly-booted machine broadcasts a DHCP DISCOVER packet. The DHCP relay agent on its LAN intercepts all DHCP broadcasts. When it finds a DHCP DISCOVER packet, it sends the packet as a unicast

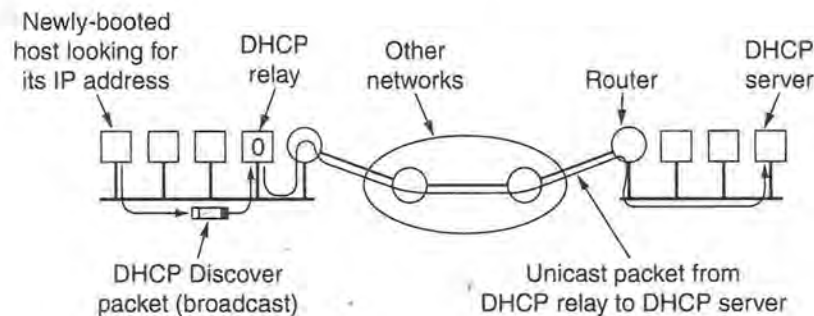


Figure 5-63. Operation of DHCP.

packet to the DHCP server, possibly on a distant network. The only piece of information the relay agent needs is the IP address of the DHCP server.

An issue that arises with automatic assignment of IP addresses from a pool is how long an IP address should be allocated. If a host leaves the network and does not return its IP address to the DHCP server, that address will be permanently lost. After a period of time, many addresses may be lost. To prevent that from happening, IP address assignment may be for a fixed period of time, a technique called **leasing**. Just before the lease expires, the host must ask the DHCP for a renewal. If it fails to make a request or the request is denied, the host may no longer use the IP address it was given earlier.

5.6.4 OSPF—The Interior Gateway Routing Protocol

We have now finished our study of Internet control protocols. It is time to move on the next topic: routing in the Internet. As we mentioned earlier, the Internet is made up of a large number of autonomous systems. Each AS is operated by a different organization and can use its own routing algorithm inside. For example, the internal networks of companies X, Y, and Z are usually seen as three ASes if all three are on the Internet. All three may use different routing algorithms internally. Nevertheless, having standards, even for internal routing, simplifies the implementation at the boundaries between ASes and allows reuse of code. In this section we will study routing within an AS. In the next one, we will look at routing between ASes. A routing algorithm within an AS is called an **interior gateway protocol**; an algorithm for routing between ASes is called an **exterior gateway protocol**.

The original Internet interior gateway protocol was a distance vector protocol (RIP) based on the Bellman-Ford algorithm inherited from the ARPANET. It worked well in small systems, but less well as ASes got larger. It also suffered from the count-to-infinity problem and generally slow convergence, so it was replaced in May 1979 by a link state protocol. In 1988, the Internet Engineering

ABOUT THE AUTHOR

Andrew S. Tanenbaum has an S.B. degree from M.I.T. and a Ph.D. from the University of California at Berkeley. He is currently a Professor of Computer Science at the Vrije Universiteit in Amsterdam, The Netherlands, where he heads the Computer Systems Group. He is also Dean of the Advanced School for Computing and Imaging, an interuniversity graduate school doing research on advanced parallel, distributed, and imaging systems. Nevertheless, he is trying very hard to avoid turning into a bureaucrat.

In the past, he has done research on compilers, operating systems, networking, and local-area distributed systems. His current research focuses primarily on the design and implementation of wide-area distributed systems that scales to a billion users. This research, being done together with Prof. Maarten van Steen, is described at www.cs.vu.nl/globe. Together, all these research projects have led to over 100 refereed papers in journals and conference proceedings and five books.

Prof. Tanenbaum has also produced a considerable volume of software. He was the principal architect of the Amsterdam Compiler Kit, a widely-used toolkit for writing portable compilers, as well as of MINIX, a small UNIX clone intended for use in student programming labs. This system provided the inspiration and base on which Linux was developed. Together with his Ph.D. students and programmers, he helped design the Amoeba distributed operating system, a high-performance microkernel-based distributed operating system. The MINIX and Amoeba systems are now available for free via the Internet.

His Ph.D. students have gone on to greater glory after getting their degrees. He is very proud of them. In this respect he resembles a mother hen.

Prof. Tanenbaum is a Fellow of the ACM, a Fellow of the the IEEE, and a member of the Royal Netherlands Academy of Arts and Sciences. He is also winner of the 1994 ACM Karl V. Karlstrom Outstanding Educator Award, winner of the 1997 ACM/SIGCSE Award for Outstanding Contributions to Computer Science Education, and winner of the 2002 Texty award for excellence in textbooks. He is also listed in *Who's Who in the World*. His home page on the World Wide Web can be found at URL <http://www.cs.vu.nl/~ast/>.