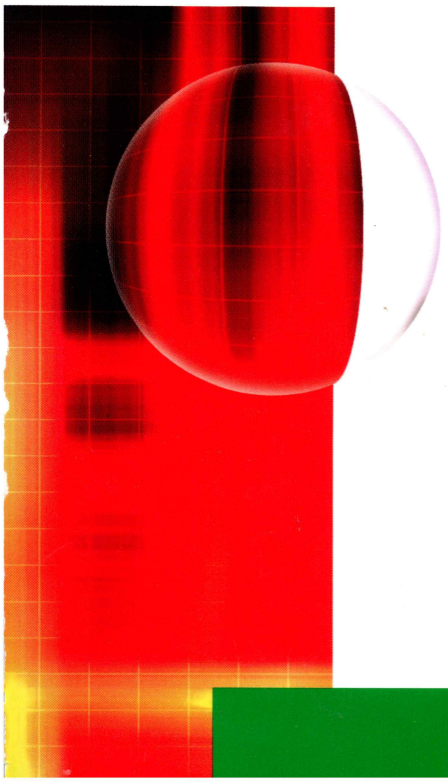
McGraw-Hill NETWORKING

PROFESSIONAL

CARRIER GRADE VOICE OVER IP

SECOND EDITION



- SIP enhancements and extensions
- MPLS
- H.323 version 4
- VoIP/SS7 interworking (SCTP, M2UA, M2PA, M3UA)
- Carrier-grade VoIP network design

DANIEL COLLINS

Motorola - Exhibit 1022 - Page 1

McGRAW-HILL NETWORKING AND TELECOMMUNICATIONS

Build Your Own

Trulove	*Build Your Own Wireless LAN* (with projects)

Crash Course

Louis	*Broadband Crash Course*
Vacca	*I-Mode Crash Course*
Louis	*M-Commerce Crash Course*
Shepard	*Telecom Convergence,* 2/e
Shepard	*Telecom Crash Course*
Bedell	*Wireless Crash Course*
Kikta/Fisher/Courtney	*Wireless Internet Crash Course*

Demystified

Harte/Levine/Kikta	*3G Wireless Demystified*
LaRocca	*802.11 Demystified*
Muller	*Bluetooth Demystified*
Evans	*CEBus Demystified*
Bayer	*Computer Telephony Demystified*
Hershey	*Cryptography Demystified*
Taylor	*DVD Demystified*
Bates	*GPRS Demystified*
Symes	*MPEG-4 Demystified*
Camarillo	*SIP Demystified*
Shepard	*SONET/SDH Demystified*
Topic	*Streaming Media Demystified*
Symes	*Video Compression Demystified*
Shepard	*Videoconferencing Demystified*
Bhola	*Wireless LANs Demystified*

Developer Guides

Vacca	*I-Mode Crash Course*
Guthery	*Mobile Application Development with SMS*
Richard	*Service and Device Discovery: Protocols and Programming*

Professional Telecom

Smith/Collins	*3G Wireless Networks*
Bates	*Broadband Telecom Handbook,* 2/e
Collins	*Carrier Grade Voice over IP*
Harte	*Delivering xDSL*
Held	*Deploying Optical Networking Components*
Minoli/Johnson/Minoli	*Ethernet-Based Metro Area Networks*
Benner	*Fibre Channel for SANs*

Bates	*GPRS*
Sulkin	*Implementing the IP-PBX*
Lee	*Lee's Essentials of Wireless*
Bates	*Optical Switching and Networking Handbook*
Wetteroth	*OSI Reference Model for Telecommunications*
Russell	*Signaling System #7*, 4/e
Minoli/Johnson/Minoli	*SONET-Based Metro Area Networks*
Nagar	*Telecom Service Rollouts*
Louis	*Telecommunications Internetworking*
Russell	*Telecommunications Protocols*, 2/e
Minoli	*Voice over MPLS*
Karim/Sarraf	*W-CDMA and cdma2000 for 3G Mobile Networks*
Bates	*Wireless Broadband Handbook*
Faigen	*Wireless Data for the Enterprise*

Reference

Muller	*Desktop Encyclopedia of Telecommunications*, 3/e
Botto	*Encyclopedia of Wireless Telecommunications*
Clayton	*McGraw-Hill Illustrated Telecom Dictionary*, 3/e
Radcom	*Telecom Protocol Finder*
Pecar	*Telecommunications Factbook*, 2/e
Russell	*Telecommunications Pocket Reference*
Kobb	*Wireless Spectrum Finder*
Smith	*Wireless Telecom FAQs*

Security

Nichols	*Wireless Security*

Telecom Engineering

Smith/Gervelis	*Cellular System Design and Optimization*
Rohde/Whitaker	*Communications Receivers*, 3/e
Sayre	*Complete Wireless Design*
OSA	*Fiber Optics Handbook*
Lee	*Mobile Cellular Telecommunications*, 2/e
Bates	*Optimizing Voice in ATM/IP Mobile Networks*
Roddy	*Satellite Communications*, 3/e
Simon	*Spread Spectrum Communications Handbook*
Snyder	*Wireless Telecommunications Networking with ANSI-41*, 2/e

BICSI

Network Design Basics for Cabling Professionals
Networking Technologies for Cabling Professionals
Residential Network Cabling
Telecommunications Cabling Installation

CARRIER GRADE VOICE OVER IP

Carrier Grade Voice Over IP

Daniel Collins

McGraw-Hill
New York Chicago San Francisco Lisbon
London Madrid Mexico City Milan New Delhi
San Juan Seoul Singapore Sydney Toronto

Motorola - Exhibit 1022 - Page 6

The McGraw-Hill Companies

Cataloging-in-Publication Data is on file with the Library of Congress.

Copyright © 2003 by The McGraw-Hill Companies, Inc. All rights reserved. Printed in the United States of America. Except as permitted under the United States Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a data base or retrieval system, without the prior written permission of the publisher.

2 3 4 5 6 7 8 9 0 DOC/DOC 0 9 8 7 6 5 4 3

ISBN 0-07-140634-4

The sponsoring editor for this book was Scott Grillo and the production supervisor was Pamela Pelton. It was set in Century Schoolbook by MacAllister Publishing Services, LLC.

Printed and bound by RR Donnelley.

This book is printed on recycled, acid-free paper containing a minimum of 50 percent recycled de-inked fiber.

McGraw-Hill books are available at special quantity discounts to use as premiums and sales promotions, or for use in corporate training programs. For more information, please write to the Director of Special Sales, Professional Publishing, McGraw-Hill, Two Penn Plaza, New York, NY 10121-2298. Or contact your local bookstore.

Information contained in this work has been obtained by The McGraw-Hill Companies, Inc. ("McGraw-Hill") from sources believed to be reliable. However, neither McGraw-Hill nor its authors guarantee the accuracy or completeness of any information published herein, and neither McGraw-Hill nor its authors shall be responsible for any errors, omissions, or damages arising out of use of this information. This work is published with the understanding that McGraw-Hill and its authors are supplying information but are not attempting to render engineering or other professional services. If such services are required, the assistance of an appropriate professional should be sought.

To Ann, my wife and friend.

CONTENTS

Preface xvii

Chapter 1 Introduction 1

What Is Meant by Carrier-Grade? 2
What Is Meant by VoIP? 3
A Little about IP 4
Why VoIP? 5
Why Carry Voice? 5
Why Use IP for Voice? 6
Lower Equipment Cost 7
Voice/Data Integration and Advanced Services 9
Potentially Lower Bandwidth Requirements 12
The Widespread Availability of IP 14
The VoIP Market 15
VoIP Challenges 16
Speech Quality 17
Managing Access and Prioritizing Traffic 19
Speech-Coding Techniques 20
Network Reliability and Scalability 21
Overview of the Following Chapters 22

Chapter 2 Transporting Voice by Using IP 25

Overview of the IP Protocol Suite 27
Internet Standards and the Standards Process 30
The Internet Society 30
The Internet Architecture Board (IAB) 31
The Internet Engineering Task Force (IETF) 31
The Internet Engineering Steering Group (IESG) 32
The Internet Assigned Numbers Authority (IANA) 32
The Internet Standards Process 32
The Internet Protocol (IP) 33
The IP Header 34
IP Routing 35
The Transmission Control Protocol (TCP) 38
The TCP Header 39
TCP Connections 42

Motorola - Exhibit 1022 - Page 10

The User Datagram Protocol (UDP) 44
Voice over UDP, not TCP 45
The Real-Time Transport Protocol (RTP) 46
RTP Payload Formats 47
The RTP Header 52
Mixers and Translators 55
The RTP Control Protocol (RTCP) 56
RTCP Sender Report (SR) 58
RTCP Receiver Report (RR) 61
RTCP Source Description Packet (SDES) 62
RTCP BYE Packet 63
Application-Defined RTCP Packet 63
Calculating Round-Trip Time 64
Calculating Jitter 65
Timing of RTCP Packets 65
IP Multicast 67
IP Version 6 (IPv6) 68
IPv6 Header 69
IPv6 Addresses 71
IPv6 Header Extensions 72
Interworking IPv4 and IPv6 76

Chapter 3 Speech-Coding Techniques 79

Voice Quality 81
A Little about Speech 82
Voice Sampling 86
Quantization 87
Types of Speech Coders 88
G.711 90
Adaptive Differential PCM (ADPCM) 90
Analysis-by-Synthesis (AbS) Codecs 91
G.728 Low-Delay CELP (LD-CELP) 92
G.723.1 Algebraic Code-Excited Linear Prediction (ACELP) 94
G.729 96
Selecting Codecs 98
Cascaded Codecs 100
Tones, Signals, and Dual-Tone Multifrequency (DTMF) Digits 100

Chapter 4 H.323 105

The H.323 Architecture 107
Overview of H.323 Signaling 109
Overview of H.323 Protocols 111
H.323 Addressing 113
Codecs 114
RAS Signaling 115
Gatekeeper Discovery 118
Endpoint Registration and Registration Cancellation 119
Endpoint Location 121
Admission 122
Bandwidth Change 125
Status 126
Disengage 127
Resource Availability 128
Service Control 128
Request in Progress 129
Call Signaling 129
Setup 131
Call-Proceeding 132
Alerting 132
Progress 133
Connect 133
Release Complete 133
Facility 134
Interaction Between Call Signaling and H.245 Control Signaling 134
Call Scenarios 135
Basic Call Without Gatekeepers 135
A Basic Call with Gatekeepers and Direct Endpoint Call Signaling 135
A Basic Call with Gatekeeper/Direct Routed Call Signaling 137
A Basic Call with Gatekeeper-Routed Call Signaling 139
Optional Called-Endpoint Signaling 141
H.245 Control Signaling 142
H.245 Message Groupings 144
The Concept of Logical Channels 144

Motorola - Exhibit 1022 - Page 12

H.245 Procedures 145
Fast Connect Procedure 151
H.245 Message Encapsulation 154
Conference Calls 155
Pre-arranged Conference 156
Ad Hoc Conference 156
The Decomposed Gateway 158

Chapter 5 The Session Initiation Protocol (SIP) 161

The Popularity of SIP 162
The SIP Architecture 163
SIP Network Entities 164
SIP Call Establishment 166
SIP Advantages over Other Signaling Protocols 167
Overview of SIP Messaging Syntax 169
SIP Requests 170
SIP Responses 173
SIP Addressing 175
Message Headers 176
Examples of SIP Message Sequences 182
Registration 183
Invitation 185
Termination of a Call 188
Redirect and Proxy Servers 189
Redirect Servers 189
Proxy Servers 190
The Session Description Protocol (SDP) 196
The Structure of SDP 198
SDP Syntax 199
Usage of SDP with SIP 206
Negotiation of Media 208
SIP Extensions and Enhancements 212
The SIP INFO Method 213
SIP Event Notification 214
SIP for Instant Messaging 216
The SIP REFER Method 217
Reliability of Provisional Responses 220
The SIP UPDATE Method 222
Integration of SIP Signaling and Resource Management 224

Motorola - Exhibit 1022 - Page 13

Usage of SIP for Features and Services 231
Call Forwarding 233
Consultation Hold 233
Interworking 236
PSTN Interworking 236
Interworking with H.323 240
Summary 245

Chapter 6 Media Gateway Control and the Softswitch Architecture 247

Separation of Media and Call Control 248
Softswitch Architecture 250
Requirements for Media Gateway Control 252
Protocols for Media Gateway Control 253
MGCP 254
The MGCP Model 255
MGCP Endpoints 255
MGCP Calls and Connections 257
Overview of MGCP Commands 258
Overview of MGCP Responses 264
Command and Response Details 270
Call Setup Using MGCP 274
MGCP Events, Signals, and Packages 278
Interworking Between MGCP and SIP 279
MEGACO/H.248 281
MEGACO Architecture 281
Overview of MEGACO Commands 286
Descriptors 288
Packages 299
MEGACO Command and Response Details 300
Call Setup Using MEGACO 307
Interworking Between MEGACO and SIP 312

Chapter 7 VoIP and SS7 313

The SS7 Protocol Suite 315
The Message Transfer Part (MTP) 316
ISDN User Part (ISUP) and Signaling Connection Control Part (SCCP) 317
SS7 Network Architecture 318
Signaling Points (SPs) 320
Signal Transfer Point (STP) 320

Motorola - Exhibit 1022 - Page 14

Service Control Point (SCP) 321
Message Signal Units (MSUs) 322
SS7 Addressing 323
ISUP 324
Performance Requirements for SS7 327
Sigtran 328
Sigtran Architecture 330
SCTP 337
M3UA Operation 351
M2UA Operation 363
M2PA Operation 367
Interworking SS7 and VoIP Architectures 369
Interworking Softswitch and SS7 369
Interworking H.323 and SS7 374

Chapter 8 Quality of Service (QoS) 377

The Need for QoS 378
End-to-End QoS 380
It's Not Just the Network 381
Overview of QoS Solutions 381
More Bandwidth 382
QoS Protocols and Architectures 383
QoS Policies 386
The Resource Reservation Protocol (RSVP) 387
RSVP Syntax 388
Establishing Reservations 389
Reservation Errors 400
Guaranteed Service 401
Controlled-Load Service 402
Removing Reservations and the Use of Soft State 403
DiffServ 404
The DiffServ Architecture 405
The Need for SLAs 406
Per-Hop Behavior (PHB) 407
Multiprotocol Label Switching (MPLS) 410
The MPLS Architecture 412
FEC and Label Formats 414
Actions at LSRs 417
MPLS Traffic Engineering 419

Motorola - Exhibit 1022 - Page 15

Label Distribution Protocols and Constraint-Based Routing 421
RSVP Traffic Engineering (RSVP-TE) 433
Combining QoS Solutions 434

Chapter 9 Designing a Voice over IP Network 437

Design Criteria 439
Build-Ahead or Capacity Buffer 440
Fundamental Technology Assumptions 440
Network-Level Redundancy 441
Voice Coder/Decoder (Codec) Selection Issues 442
Blocking Probability 444
QoS Protocol Considerations and Layer 2 Protocol Choices 445
Product and Vendor Selection 446
Generic VoIP Product Requirements 446
Element Management 449
Traffic Forecasts 450
Voice Usage Forecast 450
Traffic Distribution Forecast 453
Node Locations and Bandwidth Requirements 455
MG Locations and PSTN Trunk Dimensioning 455
MGC, SG, and EMS Dimensioning and Placement 459
Calculating VoIP Bandwidth Requirements 463
Physical Connectivity 470

Appendix A Table of Erlang B 473

Appendix B Visual Basic® Code for Erlang Calculations 489

Glossary of Acronyms 491

References 499

Index 507

Motorola - Exhibit 1022 - Page 16

PREFACE

I wrote the preface to the first edition of this book in mid-2000. At that time, the United States was still enjoying a long period of economic growth, and the telecommunications industry was one of the greatest beneficiaries of the expansion. New companies were using new technologies, such as *Voice over IP* (VoIP), to compete with traditional telecommunications companies. The result was a wealth of choices for the customer. All was well, and the preface to the first edition of *Carrier Grade over VoIP* painted a very rosy future. Two years later, it seems that my predictions (like those of many others) might have been a little premature.

As I write this preface, we are experiencing an economy that is making a faltering recovery from recession. Although some industries are recovering from the downturn, the telecommunications industry has suffered more than any other, and recovery is not yet in sight. Many of the new telecommunications companies that emerged in the late 1990s no longer exist. In hindsight, one could say that my positive outlook was proven wrong. I agree, but only to a limited extent.

Although the growth of telecommunications has not maintained its breakneck pace, growth is continuing, and much of the growth involves VoIP systems. There are more VoIP-based products and networks in service today than ever before. More multimedia applications exist that merge voice and data. In addition to the new carriers, the established carriers have embraced VoIP technology. A good deal of today's voice traffic is already carried as VoIP, and the percentage is increasing. I remain convinced that the future for VoIP is very bright.

One reason why VoIP is here to stay is the fact that true carrier-grade solutions exist—solutions that provide superior quality, reliability, and scalability. More important, these solutions cost less to purchase and operate than traditional systems. In addition, VoIP systems can enable true multimedia services—something that circuit switching cannot offer. By incorporating advances made in standards groups such as the *Internet Engineering Task Force* (IETF), today's VoIP systems offer everything that circuit-switched systems can offer and more.

The advantages of VoIP systems do not mean that we can expect a sudden wholesale replacement of installed circuit-switched systems. Instead, we can expect traditional systems to remain with us for some time, but eventually they will be phased out. Wherever there is a need for new capacity, new features, or multimedia capabilities, VoIP systems will be deployed. Given the steadily increasing demand for voice service and the

emerging demand for true multimedia, VoIP deployment will continue to grow. Eventually long, VoIP systems will be more prevalent than circuit-switching systems. Perhaps that situation will not materialize as quickly as we once thought, but it will happen. We are already moving in that direction.

Daniel Collins

Motorola - Exhibit 1022 - Page 19

CHAPTER 1

Introduction

The two terms that make up the title of this book, "carrier-grade" and "*Voice over IP* (VoIP)," mean many things to many people, and each term may have several meanings depending on the context. Furthermore, many people might consider the two terms to be mutually exclusive, making the phase carrier-grade VoIP an oxymoron. That used to be the case, but those days are over. The excitement that VoIP has generated and the resources that have been applied to developing technical solutions for VoIP have allowed it to become a serious alternative for voice communications. That is, VoIP is a serious alternative to the circuit-switched telephony that we have known for decades. Not only can VoIP technology now offer straightforward telephony services, but it can offer a lot more besides, and it can do it with the same *quality of service* (QoS) that we have become familiar with in traditional telephony.

The overall objectives of this book are to examine the technical solutions that make VoIP a reality and to illustrate how today's VoIP solutions can offer everything that traditional circuit switching provides, plus a lot more. This first chapter introduces the reader to VoIP in general and looks at some of the reasons why it is such a hot topic. Thereafter, a brief overview of technical challenges and solutions is presented. This chapter can be considered a glimpse of the topics that are covered in more depth in later chapters. Firstly, however, we need to understand what exactly is meant when we talk about terms such as VoIP and carrier-grade.

What Is Meant by Carrier-Grade?

Perhaps the easiest way to describe the term carrier-grade is to think of the last time you picked up your residential phone and failed to get a dial tone. With the exception of extraordinary circumstances, such as an earthquake, most people cannot remember such a time. Basically, the phone always works.

Carrier-grade means extremely high availability. In fact, the requirement for availability in commercial telephony networks is 99.999 percent. In other words, a carrier-grade network should be operational at least 99.999 percent of the time. This corresponds to a down time of no more than about five minutes per year and is known as *five nines reliability*.

Carrier-grade means high capacity, the capability to support hundreds of thousands, perhaps millions, of subscribers and similar numbers of simultaneous calls. For example, at the end of 2001, the Verizon network sup-

ported over 70 million voice access lines.[1] Meanwhile, the AT&T network carries about 300 million voice calls a day.[2]

Carrier-grade means that when you dial a number, you get through to the number you dialed. It means that when you finish dialing, the phone at the other end starts ringing within two to three seconds. It means that when someone answers and conversation takes place, the speech quality is very high, without any perceptible echo, noticeable delay, or annoying noise on the line.

These are tough standards to meet. In terms of the network, these requirements translate to systems that are fully redundant, in some cases self-healing, highly scalable, and manageable. They mean compliance with numerous technical specifications to ensure interoperability with other networks. They also imply a highly skilled network maintenance organization, on duty around the clock, and ready to respond to a network problem in an instant.

What Is Meant by VoIP?

VoIP is simply the transport of voice traffic using the *Internet Protocol* (IP), hardly a surprising definition. However, it is important to note that VoIP does not automatically imply voice over the Internet.

The Internet is a collection of interconnected networks, all using IP. The connections between these networks are used by anyone and everyone for a wide range of applications, from e-mail to file transfer to *electronic commerce* (e-commerce). As we shall see later, one of the greatest challenges to VoIP is voice quality and one of the keys to acceptable voice quality is bandwidth. If we are to ensure that sufficient bandwidth is available to enable high-quality voice, then we need to control and prioritize access to the available bandwidth. Currently, that does not happen on the Internet. In fact, each user of the Internet is at the mercy of all other users. One person transferring a huge file may cause other users' transactions to proceed more slowly. As a result, voice quality over the Internet today may vary from acceptable to atrocious.

[1]Verizon Annual Report, 2001.

[2]www.att.com/network.

Motorola - Exhibit 1022 - Page 22

All is not lost, however. The Internet is changing rapidly in terms of its size, the number of users, and the technology that it uses. As technology changes and as more and more bandwidth is made available, it is possible that high-quality voice over the Internet may become the norm rather than the exception. However, that day is still not at hand. Consequently, while the term carrier-grade VoIP may not be an oxymoron, the term carrier-grade voice over the Internet may well be. Internet telephony may be considered a subset of, or a special case of, VoIP (aka IP telephony). However, within this book, when we refer to VoIP, we generally exclude Internet telephony. We shall see that high voice quality in IP networks requires the use of managed networks, QoS solutions, and *service-level agreements* (SLAs) between providers. Although the Internet might include all those things in the future, today's Internet does not.

In the meantime, a significant opportunity is available to deploy VoIP in networks where access and bandwidth are better managed. One such environment is that of next-generation Telcos, those companies building telephone networks using VoIP from the outset and posing a challenge to traditional carriers.

A Little about IP

IP is a packet-based protocol. This means that traffic is broken into small packets that are sent individually to their destination. In the absence of special technical solutions, the route that each packet takes to its destination is determined independently at each network node on a packet-by-packet basis. IP is not the only packet-based protocol in existence. However, it is by far the most successful. The explosive growth of the Internet proves this.

IP itself provides no guarantees. For example, it is possible for different packets to take different routes from the origination point to the destination point, leading to the possibility of packets arriving at a destination in a different order than originally sent. Not only can packets arrive out of sequence, but some packets might not arrive at all or might be severely delayed.

To combat these shortcomings, other protocols have been developed to operate in conjunction with IP to ensure that packets are delivered to their ultimate destination in the correct sequence and without loss. The most notable such protocol is the *Transmission Control Protocol* (TCP). TCP includes functions for the retransmission of packets that may have been

lost or delayed and the assembly of packets in the correct order at the destination end. TCP is so widespread that the term TCP/IP is almost as common as IP and is sometimes used synonymously with IP. Although the mechanisms used by TCP are appropriate and successful for data transfer (such as file transfers and e-mail), they are not appropriate for the delivery of voice traffic.

Most data traffic is asynchronous and extremely error sensitive. For example, it hardly matters if an e-mail message takes 10 seconds or 30 seconds to reach its destination, but it is critical that every bit is received correctly. On the other hand, voice traffic is synchronous in nature and a little more tolerant of errors. In speech, when someone speaks, the listener should hear it practically immediately, although it is not as critical that every millisecond of speech is heard.

Given that IP provides no guarantees regarding the efficient transport of data packets, one wonders why IP would even be considered as a means for transporting voice, particularly with the stringent delay requirements that voice imposes. One also wonders how VoIP can be made to match the quality, reliability, and scalability of traditional networks. After all, if VoIP is to be a successful competitor to traditional telephony technology, then it must meet all the requirements met by traditional telephony, it must offer new and attractive capabilities beyond traditional telephony, and it must do so at a lower cost.

Why VoIP?

It is probably best to break this question into two parts. The first is, why worry about carrying voice at all when a large and lucrative market for data services exists? The second question is, assuming that a carrier intends to compete in the voice market, why should IP be considered the transport mechanism?

Why Carry Voice?

IP and the Internet have led to many new and exciting services. Whereas in the past it might have taken weeks to access certain types of information, it is now possible to find out practically anything in an instant. It is now possible to communicate at the touch of a button and to share information between colleagues, friends, and family. Many companies use IP technology

to share information between their business systems and those of their suppliers and customers in order to reduce ordering times and streamline production. All of this means that people are using IP technology in new and exciting ways. The technology has led to new revenue opportunities and to the creation of companies and enterprises to capitalize on those opportunities. However, the revenues generated as a result of those opportunities are miniscule compared to the revenues generated by telephone companies who carry voice. Voice is still the killer application.

To illustrate this point, let us look at some real examples. In 2001, Amazon.com had net sales of over $3 billion but still failed to make a profit.[3] Yahoo!, one of the pillars of the Internet, generated net revenues of over $700 million and lost money.[4] By comparison, BellSouth generated net income of over $2.5 billion from revenues of over $24 billion, with about half the revenues coming from local wireline voice services.[5] Meanwhile, Verizon generated an adjusted net income of over $8 billion from revenues of over $67 billion, again with a large portion of the revenue from regular telephony.[6] Voice is big business.

Why Use IP for Voice?

Traditional telephony carriers use circuit switching for carrying voice traffic. Given that these carriers generate large revenues and profits, why should anyone consider using a packet technology such as IP to transport voice instead of using circuit switching? Circuit switching was designed for voice from the outset and does what it is supposed to do very well. So why try to fix something that is clearly not broken?

The fact that circuit switching was designed for voice is both its strength and its weakness. There is no doubt that circuit switching carries voice very well, although it could be argued that it is an expensive solution. Circuit switching is not particularly suitable for much else, however. These days, people want to talk as much as ever, but they also want to communicate in myriad other ways—e-mail, instant messaging, video, and so on. Circuit switching really does not qualify as a suitable technology for this new world

[3]Amazon.com Annual Report, 2001.

[4]Yahoo!, Inc. Annual Report, 2001.

[5]Bellsouth Annual Report, 2001.

[6]Verizon Annual Report, 2001.

of multimedia communications. Furthermore, circuit-switching equipment has a number of characteristics that are less than ideal and that negatively affect vendors and operators of that equipment. IP offers not only another technology choice, but also a new way of doing business, as described shortly.

IP is an attractive choice for voice transport for several reasons. These include

- Lower equipment cost
- Lower operating expense
- Integration of voice and data applications
- Potentially lower bandwidth requirements
- The widespread availability of IP

Lower Equipment Cost

Although thousands of standards and technical specifications for circuit-switched telephony have been made, the systems themselves are generally proprietary in nature. A switch from a given supplier will often use proprietary hardware, and a proprietary operating system and the applications will use proprietary software. A given switch supplier may well design and produce everything, including processors, memory cards, cabling, power supplies, the operating system, and application software, as well as perhaps even the racks to hold the equipment. The opportunities for third parties to develop new software applications on these systems are extremely limited. The systems generally also require extensive training to operate and manage.

Consequently, when an operator chooses to implement systems from a given vendor, it is not unusual to choose just a single vendor for the whole network, at least at the beginning. If the operator wants to replace operating equipment with products from a new vendor, the cost and effort involved are huge. The result of all this is that the equipment vendor, once chosen, is in a position to generate revenue not only from the sale of the equipment, but from training, support, and feature development. In this environment, proprietary systems are the order of the day.

This is very much like the mainframe computer business of several years ago. In those days, computers were large monolithic systems. The system vendor was responsible for all the components of the system, plus installation, training, and support. The choice of vendor was limited and once a system from a particular vendor was installed, it was extremely difficult to

change to another vendor. Since those days, the computer world has changed drastically from the mainframe model. Although many mainframe systems are still in place, many of the big vendors of mainframe systems no longer exist. Those who still exist do so in a very different form.

The IP world is very different from the monolithic systems of mainframe computers and circuit-switching technology. Though some specialized hardware systems exist, much of the hardware is standard mass-produced computer equipment. This offers a greater choice to the purchaser and the opportunity to benefit from the large volumes that are produced. It is common for the operating system to be less tightly coupled to the hardware and for the application software to be quite separate again. This separation enables a greater range of choices for the purchaser of the system and the ability to contract with a separate company to implement unique features. Furthermore, IP systems tend to use a distributed client-server architecture rather than large monolithic systems, which means that it is easy to start small and grow as demand dictates. In addition, this type of architecture means that certain companies make only portions of the network solution, allowing the customer to pick those companies that are the best in the different areas to create a solution that is optimum in all respects. Figure 1-1 gives a pictorial comparison between the mainframe approach of circuit-switched systems and that of packet-switching technologies. In the words of Cisco CEO John Chambers, "When a horizontal business model meets a vertical business model, horizontal wins every time."

Not only are IP architectures more open and competition-friendly, but the same applies to IP standards. They are more open and flexible than telephony standards, allowing for the implementation of unique features, so that a provider can offer new features quicker and be better able to customize service offerings. New features can be developed and deployed in a

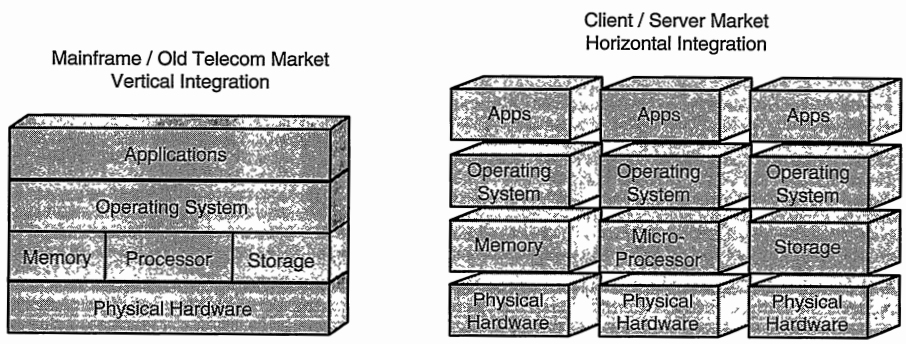


Figure 1-1
Architecture and business model comparison

few months, which is a big difference from the traditional telephony world where new features can take 18 months to 2 years.

One could argue that an *intelligent network* (IN) architecture in telephony also offers great flexibility and rapid feature development. Although that is what IN has promised from the beginning, it does not match the openness and flexibility of IP solutions. With the exception of a few highly successful services, such as toll-free calling, IN has not had the enormous success that was originally envisaged. Of course, those IN services that are successful happen to be extremely important and also need to be provided by an IP-based solution. As we shall see in Chapter 7, "VoIP and SS7," VoIP networks can interwork with *Signaling System 7* (SS7) and take advantage of IN services that are built on SS7. Hence, IP can take advantage of what is already there and can do more besides.

All of the foregoing points mean greater competition, greater flexibility on the part of the network operator, and significantly lower cost. Furthermore, the packet-based systems obey Moore's Law, which states that processing power doubles roughly every 18 months, which is a much faster pace of development than we see with traditional circuit-switched systems. Figure 1-2 shows a comparison between the two. We can clearly see that circuit-switching equipment lags behind packet and frame technologies in the speed of development.

Of course, all the major manufacturers of traditional telephony switches now also produce VoIP solutions, but a quick search on the Internet will reveal a large number of new companies also producing VoIP products. The range of choices is large, the equipment cost is lower than that of circuit-switching products, and the pace of development is faster.

Voice/Data Integration and Advanced Services

IP is the standard for data transactions—everything from e-mail to Web browsing to e-commerce. When we combine these capabilities with the transport of voice, it is easy to imagine advanced features that can be based on the integration of the two. One example is an application where the receiver of an e-mail can call the sender simply by clicking a header within the e-mail. Another example is a button on a web page that a user can click to speak to a customer service representative for assistance in navigating the site or conducting a transaction, as depicted in Figure 1-3. When the user clicks that button, it is possible to direct the call to the most appropriate customer service representative based on what the user was attempting to do just before deciding to seek help.

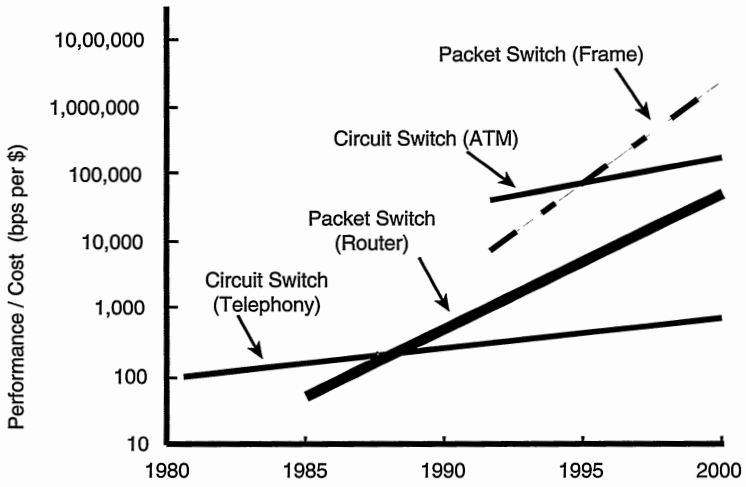


Figure 1-2
Cost performance comparison

Platform	Time to double performance /cost (months)
Frame	10
Router	20
ATM	40
Circuit	80

Source: Peter J. Sevcik, President, NetForecast, peter@netforecast.com
"Why Circuit Switching is Doomed," Business Communications Review, Sep. 1997

Web collaboration is a service whereby one user can navigate the Web and another user at another location is presented with the same web page. The two users may talk to each other at the same time, as depicted in Figure 1-4.

Yet another example is video conferencing and all that it entails. Not only does this include person-to-person video telephony, but it can also include shared whiteboard sessions. Design teams can come together to discuss issues and sketch out solutions while sitting hundreds or thousands of miles from each other. With IP multicasting, virtual auditorium services are available, something of great interest for distance learning in particular.

These types of applications are not just wishful thinking. Many applications exist today, and more are coming. Such applications include IP-based *private branch exchanges* (PBXs), IP-based call centers, and IP-based voice mail, as well as straightforward telephony. IP-based telephony devices exist today, and they are far more feature-rich than the standard 12-button key-



Figure 1-3
Click-to-talk application

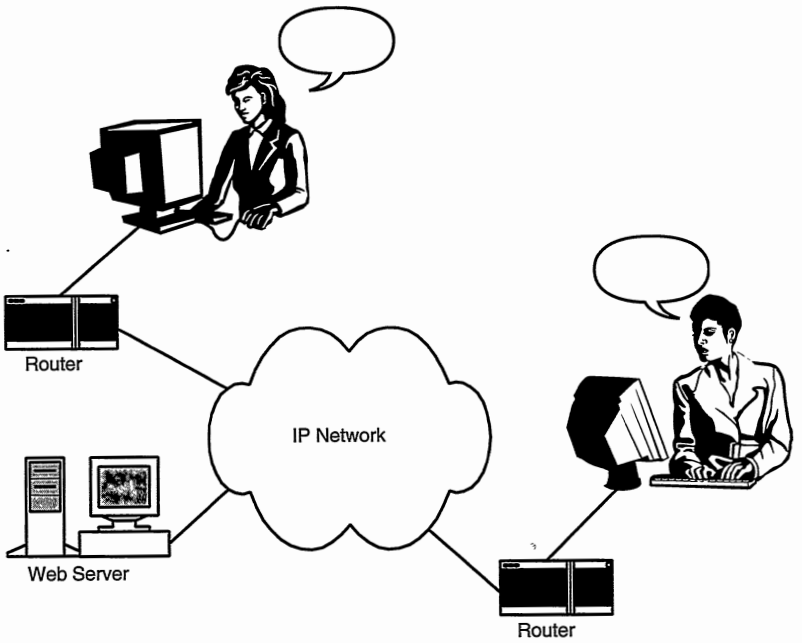
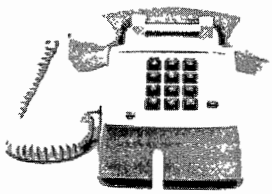


Figure 1-4
Example application: web collaboration

pad that we have known for years. Many include web browsers, address book applications, e-mail clients, and so on. The user interface is graphical and intuitive, and we no longer have to worry about obscure "star codes" to activate features. Figure 1-5 shows an example of two IP-based telephony devices. Contrast these examples with the traditional telephone keypad and ask yourself which user interface you would prefer.

Potentially Lower Bandwidth Requirements

Circuit-switched telephone networks transport voice at a rate of 64 Kbps. This is based on the Nyquist Theorem, which states that it is necessary to sample an analog signal at twice the maximum frequency of the signal in order to fully capture the signal. Typical human speech has a maximum frequency of somewhat less than 4,000 Hz. Therefore, in digitizing human speech, a telephone system takes 8,000 samples per second, as directed by





Pingtel



Congruency, Inc.

Figure 1-5
New versus old phones

the Nyquist Theorem, and each sample is represented with 8 bits. This leads to a bandwidth requirement of 64 Kbps in each direction for a standard telephone call. This voice-coding scheme is standardized in *International Telecommunication Union* (ITU) Recommendation G.711.

Although G.711 is the standard for traditional telephony, many other coding schemes are available. These use more sophisticated coding algorithms, which enable speech to be transmitted at a range of different speeds, such as 32, 16, 8, 6.3, or 5.3 Kbps. Furthermore, some of these coding techniques utilize silence suppression such that traffic is passed over the network only when something is being said. Given that most conversations involve one person listening and one person talking at a given time, the bandwidth saved by not transmitting silence can be significant. Various speech-coding schemes are discussed in detail in Chapter 3, "Speech-Coding Techniques," and all of those described can be used in VoIP systems. Therefore, VoIP can offer significant advantages over circuit-switched telephony from a bandwidth-efficiency point of view.

It should be noted, however, that the availability of these coding schemes is not something unique to VoIP. Strictly speaking, these various coders could also be deployed in traditional telephony networks. However, they are not generally implemented in circuit-switched networks, as it would require an enormous investment to do so. Currently, it is possible to place a call from any telephone to any other telephone in the world. To take advantage of more advanced coding schemes would require that these coding techniques be implemented in practically every telephone switch in the world. Alternatively, it would be necessary that existing networks support the capability for the two ends of a conversation to negotiate the coding scheme to use. At the bare minimum, it would require that calls be routed from the origination to the destination based on the coding choice of the originator so that, if different coders are used at either end, the call could be passed though some interworking device. None of these capabilities are supported in traditional telephony networks. As a result, traditional telephony uses G.711 and is unlikely ever to change.

On the other hand, VoIP solutions are far more recent than traditional telephony. They generally support more efficient coding schemes than G.711 and the standards written for VoIP allow for the two ends of a call to negotiate which coding scheme to use. Thus, VoIP is in a position to significantly reduce bandwidth requirements—to as little as one-eighth of that used in the circuit-switched world. Given that transmission capacity can account for a large percentage of a carrier's initial investment and a very large portion of a carrier's operational costs, such bandwidth savings can mean a big difference to the bottom line.

Motorola - Exhibit 1022 - Page 32

Although it is true that bandwidth-efficient coding schemes can reduce overall bandwidth requirements, we shall see in Chapter 9, "Designing a VoIP Network," that IP and associated protocols add some overhead beyond the bandwidth of the coded voice itself. Therefore, coding voice at, say, 32 Kbps, does not necessarily mean a 50 percent bandwidth reduction compared to 64 Kbps G.711 coding. Beware of anyone who suggests such a simple comparison. The exact bandwidth saving compared to traditional telephony is dependent on a number of factors and requires some calculation. These calculations are described in Chapter 9.

Not only can transmission costs be reduced through the use of more efficient voice-coding schemes, but the fundamental architecture of VoIP systems lends itself to more transmission-efficient network designs. Recall that traditional circuit-switching networks tend to be composed of large monolithic systems. These systems lend themselves to centralized network architectures, with both call control signaling and bearer traffic being routed through the same centralized machines. VoIP systems, however, are designed to be distributed. Although call control signaling can be centralized in a limited number of centralized machines, the bearer traffic does not have to be routed through the same machines. Rather, the bearer traffic can be routed more directly from source to destination, leading to shorter transmission distances and lower cost.

The Widespread Availability of IP

IP is practically everywhere. Every personal computer produced today supports IP. It is used in corporate *local area networks* (LANs) and *wide area networks* (WANs), as well as in dial-up Internet access. IP applications now even reside within hand-held computers and various wireless devices. As a result, IP expertise is widespread and application development companies are numerous. This ubiquitous presence alone makes IP a suitable choice for transporting voice, or any other digital media stream for that matter.

Of course, VoIP is not the only packet-based solution available to a commercial carrier. In fact, *Voice over Frame Relay* (VoFR) and *Voice over Asynchronous Transfer Mode* (VoATM) are powerful alternatives. One of the great disadvantages of such solutions, however, is that they do not have the same ubiquitous presence as IP. Although many users currently use IP, whether they know it or not, the same cannot be said for Frame Relay or ATM. IP is already at the desktop and it is hard to imagine that the same will ever apply to ATM or Frame Relay.

To some extent, IP at the desktop might not matter to a next-generation Telco if the main objective is just to capture market share in the long-

Motorola - Exhibit 1022 - Page 33

distance business. Such a business could be easily served by other technologies besides IP. To ignore IP, however, would be to decline possible opportunities offered by voice and data integration. It would also limit choices and opportunities for advanced applications, and it would reduce the possible benefits from the myriad advances being made almost daily in the IP community. That is not to say that ATM and Frame Relay do not have a role to play. They certainly do, perhaps in the core of the network as an underlying network transport for IP, but not at the user device.

The VoIP Market

It is clear from the previous descriptions that VoIP can offer cost savings, new services, and a new way of doing business, both for carriers and network operators. It is also clear that voice is an important service, but not the only one. Rather, it is part of a suite of services that, when packaged together, can offer exciting new capabilities. All of this translates into serious revenue opportunities. Although the business is currently only in its infancy and much of the revenue is yet to be seen, the future is certainly bright. Figure 1-6 gives one projection of the market for Voice over Packet.

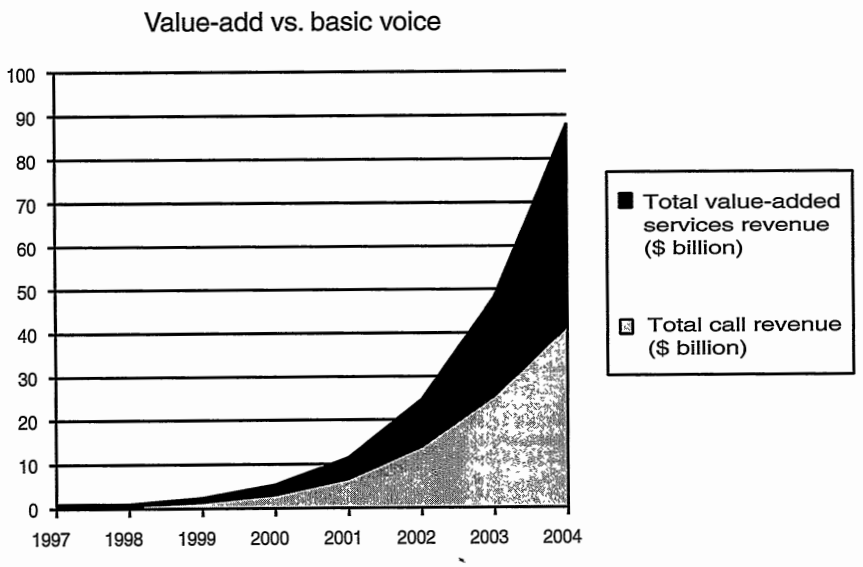


Figure 1-6
Voice over Packet revenue projection

Source: Gartner/Dataquest/Bhawani Shankar/"Voice on the Net"/March 2000

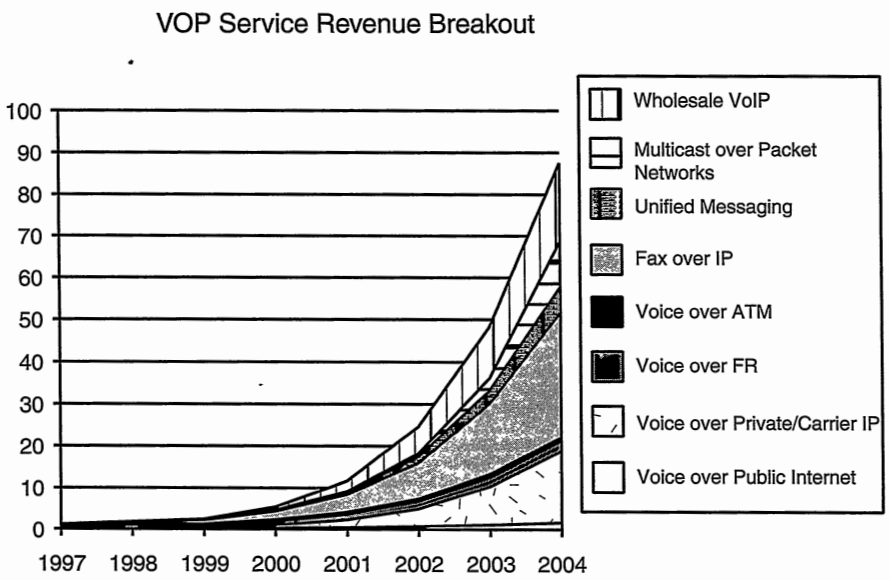


Figure 1-7
Projected breakdown of VoIP-related revenue

Source: Gartner/Dataquest/Bhawani Shankar/"Voice on the Net"/March 2000

It is interesting to note that Figure 1-6 refers to Voice over Packet, as opposed to VoIP. One would be curious to know what the breakdown might be, in particular how IP fares against other technologies such as ATM or Frame Relay. Figure 1-7 provides that breakdown, which shows that IP is expected to be dominant, and that ATM and Frame Relay are expected to comprise a relatively small percentage of the market.

VoIP Challenges

To offer a credible alternative to traditional circuit-switched telephony, VoIP must offer the same reliability and voice quality. In other words, the five nines availability requirement must be met and speech must be of toll quality. Toll quality generally means a *Mean Opinion Score* (MOS) of 4.0 or better on the following scale, as specified in *International Telecommunication Union Telecommunications Standardization Sector* (ITU-T) Recommendation P.800:

- 5—Excellent
- 4—Good

- 3—Fair
- 2—Poor
- 1—Bad

Speech Quality

Perhaps the most important issue in VoIP is ensuring high speech quality. Anyone who has made calls over the Internet can attest to the fact that the quality is variable at best. In order for VoIP to be a commercial challenge to circuit-switched technology, the voice quality must be at least as good as experienced in today's telephony networks and should not vary. The reason why this is such a challenge for VoIP networks is the fact that IP was not originally designed to carry voice or similar real-time interactive media.

IP was designed from the outset for data. Among the characteristics of most data traffic is that it is asynchronous, which means that it can tolerate delay. Data traffic is also extremely sensitive to packet loss, which requires mechanisms for the recipient of data packets to acknowledge a receipt, such that the sender can retransmit packets if an acknowledgment is not returned in a timely manner. Voice, however, is very sensitive to delay but is more tolerant of packet loss. Clearly, voice traffic imposes requirements that are quite different from those imposed by data traffic.

Delay In many ways, real-time communications such as voice are almost the exact opposite of data, at least from a requirements perspective. To begin with, voice is more tolerant of packet loss, provided that lost packets are kept to a small percentage of the total (fewer than five percent). Voice, however, is extremely intolerant of delay. Anyone who has ever made an international call over a satellite knows how annoying delay can be. In a satellite call, it takes about 120 milliseconds for the signal to travel from the earth to the satellite and another 120 to come back down. If we add in some additional delay due to processing and terrestrial transmission, we have a one-way delay from A to B of 250 to 300 milliseconds. This means a round-trip delay of 500 to 600 milliseconds.

It is the round-trip time that is important. Consider, for example, a conversation between A and B. A is speaking and B decides to interrupt. However, because of a delay, what B is hearing is already old by the time he hears it. Furthermore, when B interrupts, it takes time for A to hear the interruption. Therefore, it seems to B that A has ignored the interruption. Therefore, B stops. Now, however, A finally hears the interruption and

stops, so there is silence. Then they may both try to start again and both back off again. The whole experience can be very frustrating if the round-trip delay is much more than about 300 milliseconds.

ITU-T Recommendation G.114 states that the round-trip delay should be 300 milliseconds or less for telephony. Contrast this with the delay requirements for transmission of an e-mail. It hardly matters if an e-mail takes 30 seconds or 3 minutes to reach the recipient, provided that it all arrives exactly as sent.

Jitter Just as important as the actual delay is delay variation, also known as jitter. If some delay occurs in a given conversation, then it is possible for the parties to adjust to the delay fairly quickly, provided that the delay is not too large. However, it is difficult to adjust to delay if it keeps varying. Therefore, another strong requirement is to ensure that jitter is minimized so that whatever delay exists, it is at least kept constant. This can be achieved through the use of jitter buffers, where speech packets are buffered so that they can be played out to the receiver in a steady fashion. The downside of these buffers, however, is the fact that they add to the overall delay.

Jitter is a concern only for real-time communications such as voice. It is not an issue for the types of traffic traditionally carried over IP networks. It is caused in IP networks in two ways. First, packets can take different routes from sender to receiver and consequently experience different delays. Second, a given packet in a voice conversation can experience longer queuing times than the previous packet—even if they both traveled the same route. This is due to the fact that the network resources, and particularly queues within routers, might be used by other traffic as well as voice traffic.

In contrast to the situation in IP networks, jitter is not an issue in circuit-switched networks, because an open, dedicated pipe exists from the sender to the receiver for the duration of the conversation. Thus, all the speech follows the same path and does not have to share resources with any other traffic. What goes in one side comes out the other side after a fixed delay. That delay is generally tiny—at least in the absence of satellite links.

Packet Loss Another factor in high-quality voice is that little or none of the speech should be lost between the speaker and listener. Obviously, it is important to ensure that speech should be received as transmitted. It is possible, however, that queues might overflow in network nodes between sender and receiver, resulting in the loss of packets. Although traditional retransmission mechanisms are used in the case of lost data packets, those

same mechanisms cannot be applied to real-time communications (such as voice). First, it takes some time for the two ends to determine that a packet is missing. Second, it takes more time to retransmit the missing packet.

For example, if packets 1 through 5 and 7 through 9 have been received, then the receiving end has a choice. It can either present what has been received or hold packets 7 through 9 until packet 6 is received and then play packets 6 through 9. The second of these two approaches simply does not work, because the delay would be intolerable. Therefore, if a packet is missing, then the end systems must carry on without it. If, however, a packet turns up out of sequence, then there is the risk that it will be presented to the recipient out of sequence (another no-no).

In a circuit-switched network, all speech in a given conversation follows the same path and is received in the order in which it is transmitted. If something is lost, then that loss is the result of a fault rather than an inherent characteristic of the system. If IP networks are required to carry voice with high quality, then they must emulate the performance of circuit-switched networks for voice traffic. In other words, mechanisms are required to ensure that packet loss is minimized.

Managing Access and Prioritizing Traffic

One of the great advantages that VoIP offers is the capability to use a single network to support a wide range of applications, including data, voice, and video. A single multifunctional network, as opposed to multiple discreet networks, means lower capital cost and lower operating cost. A single network does, however, introduce certain challenges. Because of the fact that voice and data have different quality requirements, as described earlier, it is important that they are not treated the same way within the network. For example, it is critical to make sure voice packets are not waiting in a queue somewhere while a large file transfer is occupying network bandwidth.

The fundamental difference between different types of traffic translates to a number of network requirements. First, it is necessary that a voice call not be connected if not enough network resources are available to handle the call. The last thing that we want is to answer the phone and find that conversation is not possible because not enough bandwidth is available. Therefore, it is necessary to ensure up front that sufficient bandwidth exists before any phone rings.

Secondly, different types of traffic must be handled in the network in different ways. It is necessary to prioritize certain types of traffic to ensure

that the most critical traffic is least effected by network congestion. In practical terms, if a network becomes heavily loaded, e-mail traffic should feel the effects before synchronous traffic (such as voice).

Managing and prioritizing traffic as well as minimizing packet loss, delay, and jitter all fall into the realm of QoS. This is an area of development within the IP community that has seen a huge effort in recent times. Chapter 8, "Quality of Service," describes a number of solutions for QoS in IP networks.

Speech-Coding Techniques

Yet another factor in high-quality voice is the choice of speech-coding technique. It is all very well to reduce bandwidth, minimize delay, and minimize packet loss. However, that is not much good if the speech still sounds tinny or synthetic. Consequently, we need to select a solution that reduces bandwidth while maintaining natural-sounding speech. In general, coding techniques are such that speech quality degrades as bandwidth reduces. However, the relationship is not linear, so it is possible to significantly reduce bandwidth at the cost of only a small degradation in quality. In general, the objective is to offer, or at least approach, toll-quality voice, where toll quality relates to an MOS of 4.0 or better on the ITU-T Recommendation P.800 scale listed earlier in the chapter. Table 1-1 shows MOS values for several common coding schemes.

Although concepts like "good" and "fair" do not sound very scientific, they are not quite as nebulous as they might first appear. The ratings can be considered a summation of various quality parameters such as delay, jitter, echo, noise, crosstalk, intelligibility, and overall lack of distortion. The man-

Table 1-1

Common speech coder bit rates and MOSs

Speech coder	Bit rate (Kbps)	MOS
G.711	64	4.3
G.726	32	4.0
G.723 (celp)	6.3	3.8
G.728	16	3.9
G.729	8	4.0

ner in which various coders are tested and the scoring methodology behind it are well standardized to the extent that the MOS scale is the most common standard for voice-quality measurement today.

Nonetheless, MOS values are still subjective in nature. Therefore, regardless of how well tests are carried out, the subjective character of the evaluation means that one might find slightly different ratings for a given coding scheme. The variations may depend on the details of the test environment and the individual test participants. Therefore, anyone who intends to deploy VoIP should perform his or her own analysis of the available speech coders in a test environment that closely matches the expected network conditions. Chapter 3 provides details regarding voice-coding technologies.

Network Reliability and Scalability

Not only should a voice sound clear and crisp and be free from noticeable delay, but the service should be always available. Recall that the phone system almost never fails; contrast that with office computer networks, some of which seem to fail on a regular basis. If VoIP is to be a commercial challenge to established telephony networks, then the network must be rock solid. In other words, the five nines availability requirement must be met.

Unlike earlier systems, today's VoIP solutions do offer reliability and resilience. Moreover, the solutions are designed to enable redundancy and load-sharing within the network. Therefore, individual network nodes are less likely to fail, and if a node does fail, then another node in the network can assume the load of the failed node. Of course, networks must be designed to take advantage of these capabilities. Building redundancy into a network, however, means that the network will be more expensive. Therefore, a balance must be struck between network cost and network quality. Finding the right balance is the responsibility of the network architect.

Not only must a network be reliable, but it must also be scalable. In other words, it must be possible to increase the capacity of the networks to handle millions of simultaneous calls. The early VoIP systems did not scale to such capacities. Nowadays, however, VoIP systems can handle enormous demands. Some of today's VoIP gateway products, for example, can handle many gigabits per second of voice traffic with equipment that occupies just a single 19-inch shelf.

An advantage of VoIP architectures is that they are relatively easy to start on a small scale and then expand as traffic demand increases. Standardized protocols enable small network nodes to be controlled by the same

Motorola - Exhibit 1022 - Page 40

call-processing systems as used for large network nodes. Therefore, an expansion of the network capacity can be done in small or large steps as needs dictate.

Overview of the Following Chapters

This chapter has provided a brief introduction to VoIP and highlighted some of its attractions and challenges. The remainder of this book is devoted to addressing these issues in further detail.

We begin these detailed discussions in Chapter 2, "Transporting Voice by Using IP," with a review of IP networking in general. This will help the reader to understand what IP offers, why it is a best-effort protocol, and why carrying real-time traffic (such as voice) over IP has significant challenges. We focus particularly on those protocols that run on top of IP and help to make such real-time transport a possibility. Specifically, we review the *Real-Time Transport Protocol* (RTP), the protocol that is used for carrying voice packets over IP.

Chapter 3 focuses on voice-coding techniques. We have already mentioned that IP as a transport protocol can take advantage of sophisticated and efficient voice-coding schemes. Numerous coding schemes exist, however, and choosing the right coding scheme for a particular network or application is not necessarily a simple matter. When choosing a coding scheme, we must strike a balance between the conflicting goals of low bandwidth and high speech quality, which means that we must understand the advantages and disadvantages of all the coding schemes available.

Chapter 4, "H.323," discusses the H.323 architecture and protocol suite. H.323 has been the standard for VoIP for several years. Even though VoIP technology is relatively new, it has already seen significant deployment—both in private VoIP networks and over the Internet. Initially, however, VoIP implementations used proprietary technology, which did not easily enable communication between users of different equipment. The emergence of H.323 has helped to change that situation. As a result, H.323 is the most widely deployed VoIP technology at the moment. Standardized alternatives to H.323 are now available and some experts believe that H.323 will not be the future standard of choice. Nonetheless, H.323 has a large embedded base and is more mature than other technologies. H.323 is likely to play a major role in VoIP networks for some time to come.

Motorola - Exhibit 1022 - Page 41

Chapter 5, "The Session Initiation Protocol (SIP)," discusses the rising star of VoIP technology. Though quite new and not yet widely deployed, SIP is hailed by many as the standard of the future for VoIP. Simple in design and easy to implement, the simplicity of SIP is one of its greatest advantages. However, simplicity does not mean that flexibility is sacrificed. Instead, SIP is extremely flexible and supports a range of advanced features. Many in the VoIP industry are convinced that SIP will overtake H.323 in the marketplace.

Interworking with other networks, such as the *Public Switched Telephone Network* (PSTN), is a major concern in the deployment of VoIP networks. Such interworking requires the use of gateways. Hand in hand with SIP are two protocols that deal with the control of those gateways: the *Media Gateway Control Protocol* (MGCP) and MEGACO.[7] The two protocols address the same issue. MGCP is the standard of today but is to be replaced in the near future with MEGACO (aka ITU-T H.248). The main advantage of each is that they enable a widely distributed VoIP network architecture, whereby call control can be centralized and gateways can be distributed close to the sources and sinks of traffic. This is known as the softswitch architecture. The softswitch architecture, MGCP and MEGACO, are described in Chapter 6, "Media Gateway Control and the Softswitch Architecture."

Of critical importance to a communications system is the signaling system that it uses. In fact, H.323, SIP, MGCP, and MEGACO are all signaling protocols, while the state of the art in PSTN signaling is SS7. SS7 is the signaling system that has made call setup as quick as it is today and it provides the foundation for numerous services such as caller-ID, toll-free calling, calling card services, and mobile telephony. These are all services that need to be supported just as well in a VoIP network as in a traditional network. Moreover, traffic must be able to flow between new VoIP networks and established SS7-based networks. Thus, VoIP networks must be able to interwork with SS7 networks. Chapter 7 "VoIP and SS7" describes the technology used to enable such interworking.

Chapter 8 deals with QoS. We have already described the quality challenges that a VoIP network must face to meet the stringent performance requirements that define a carrier-grade network. A great deal of work has

[7]MEGACO also stands for "Media Gateway Control Protocol," but it is a different protocol from MGCP.

been done in this area and various solutions are discussed in Chapter 8. These solutions can enable VoIP to reach the goal of being the new technology of choice for voice transport.

Understanding technologies and protocols is all very well. At the end of the day, however, those technologies and protocols are simply a set of tools. To design and deploy a carrier-grade VoIP network, we must understand how to use those tools to best effect. We need to address issues of network dimensioning, traffic engineering, and traffic routing. We need to understand how we can build redundancy and diversity into a network without losing sight of the trade-off between network quality and network cost. Chapter 9 "Designing a Voice over IP Network" describes these issues in detail.

Motorola - Exhibit 1022 - Page 43

CHAPTER 2

Transporting Voice by Using IP

The Internet is a collection of networks, both public and private. The private networks include the *local area networks* (LANs) and *wide area networks* (WANs) of various institutions, corporations, businesses, and government agencies. The public networks include those of the numerous *Internet service providers* (ISPs). Although the private networks may communicate internally using various communications protocols, the public networks communicate using the *Internet Protocol* (IP). In order for one private network to communicate with another via a public network, then each private network must have a gateway to the public network and that gateway must speak the language of that network—IP.

We illustrate this situation in Figure 2-1, where a number of private networks are shown, connected to a number of routers. The various entities within a given private network may communicate internally in any way they want, but when communicating with other networks via the Internet, they do so with IP. Routers provide the connectivity between various networks and form the backbone of the Internet.

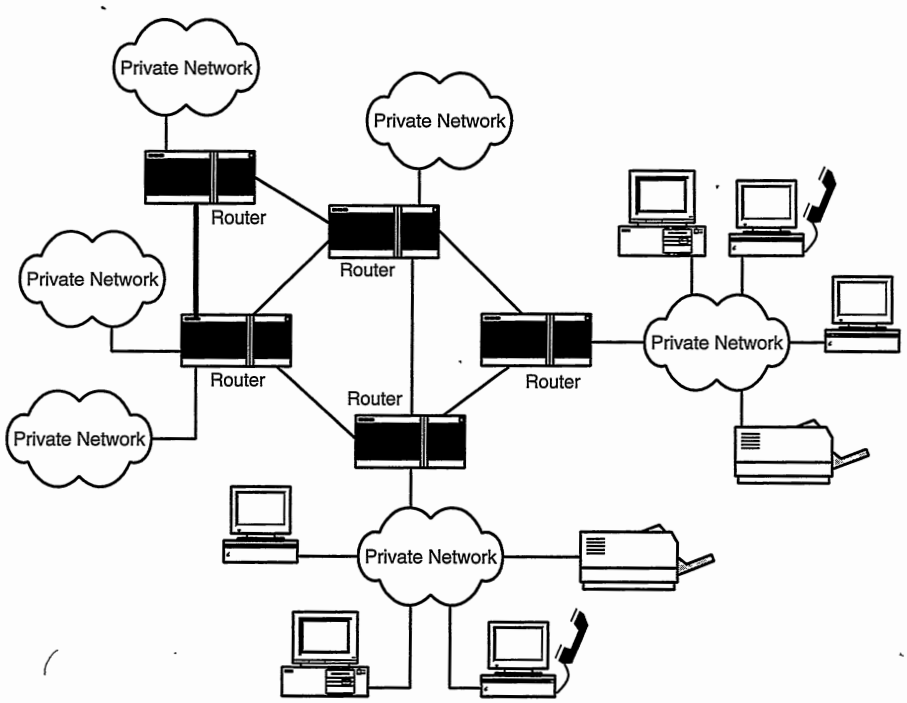


Figure 2-1
Interconnecting networks

The foregoing does not mean to imply that private networks like LANs and WANs do not use IP. The vast majority of them do. The reasons are numerous, including the fact that IP is embraced by all major developers of computer systems, and it supports a vast array of applications. Moreover, compared to some other types of networks, it can be relatively easy to implement an IP-based network.

IP and some of the related protocols would require a separate book to describe in any detail. In fact, numerous books are available. The main intention in this chapter is only to provide a high-level overview of IP in general and of some of the aspects of particular significance to the transport of voice using IP. This will allow a better understanding of the architectures and protocols described in subsequent chapters. A part of this chapter is dedicated to a brief explanation of the process for developing an Internet standard. Given that at the time of writing this book, many new specifications or enhancements for *Voice over IP* (VoIP) are being finalized or enhanced, it is important to have a high-level understanding of the standardization process.

Overview of the IP Protocol Suite

Strictly speaking, IP is a routing protocol for the passing of data packets. Other protocols invoke IP for the purposes of getting those data packets from origination to destination. Thus, IP must work with the cooperation of

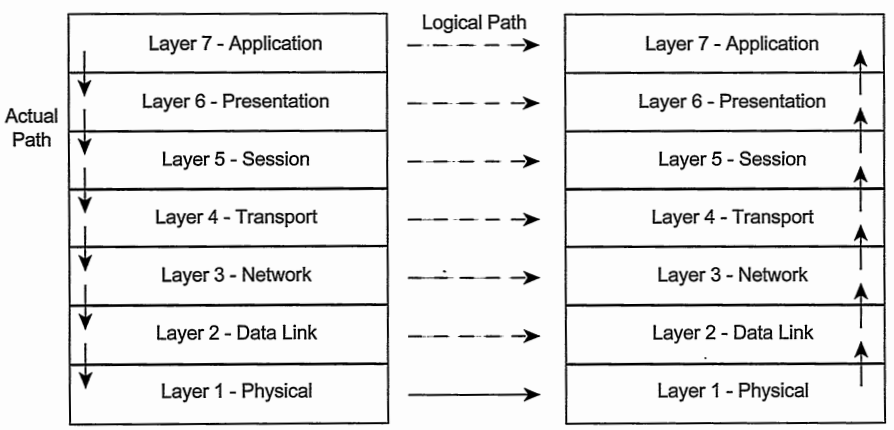


Figure 2-2
OSI seven-layer stack

higher-layer protocols in order for any application or service to operate. Equally, IP does not incorporate any physical transmission facilities, so it must work in cooperation with lower-layer transmission systems.

When we use the term layer, we generally refer to the *Open System Interconnection* (OSI) seven-layer model. Figure 2-2 shows the OSI model. Each side of a communications transaction involves a number of layers. The top layer represents the useable information to be passed to the other side. However, in order to pass this information, it must be packaged appropriately, it must be routed correctly, and it must traverse some physical medium.

Consider, for example, a simple conversation between person A and person B. Person A has a thought to be communicated to person B. In order to communicate that thought, person A must articulate the thought in language using words and phrases. Those are communicated to the mouth using nerve impulses from the brain. The mouth constructs sounds, which incorporate that language. Those sounds are transmitted through the air using vibrations. At person B, the vibrations are collected and the sounds extracted. These are passed to the brain using nerve impulses. At the brain, the original thought or concept is extracted from the language used. Thus, the thought has passed from the mind of person A to the mind of person B, but it has used the services of other functions in order to do so.

The OSI model includes seven layers as follows:

- The *physical layer* includes the physical media used to carry the information, such as cables and connectors. It also includes the coding and modulation schemes used to represent the actual 1's and 0's.
- The *data link layer* enables the transport of information over a single link. This includes the packaging of the information into frames and may also include functions for error detection and/or correction and retransmission.
- The *network layer* provides functions for the routing of traffic through a network from the source to the destination. Since a source of traffic might not be connected directly to the required destination, the information needs to pass through intermediate points, where the information can be forwarded in the right direction until it hits the correct destination.
- The *transport layer* provides functions that ensure the error-free, omission-free, in-sequence delivery of data. To support multiple streams of information from the source to destination, it provides

mechanisms for identifying each individual stream and ensures that each stream is passed upward to the correct application.

- The *session layer* deals with the commencement (logging in) and completion (logging out) of a session between applications. It establishes the type of dialog to be established, such as one way at a time or both ways at the same time. In the case of one way at a time, it keeps track of whose turn is next.
- The *presentation level* specifies the language to be used between applications. This includes the encoding to be used for the various pieces of information, such as whether certain pieces of information are coded as ASCII or binary.
- The *application layer* provides an interface to the user or at least to the user application. For example, file transfer programs and web browsers belong to the application layer.

Figure 2-3 shows a correlation between the IP protocol suite and the OSI model. We can see that IP itself resides at layer three. Above IP is a choice of protocols: the *Transmission Control Protocol* (TCP) and the *User Datagram Protocol* (UDP).

TCP ensures a reliable, error-free delivery of data packets. When multiple packets are being sent, it ensures that all of them arrive at the

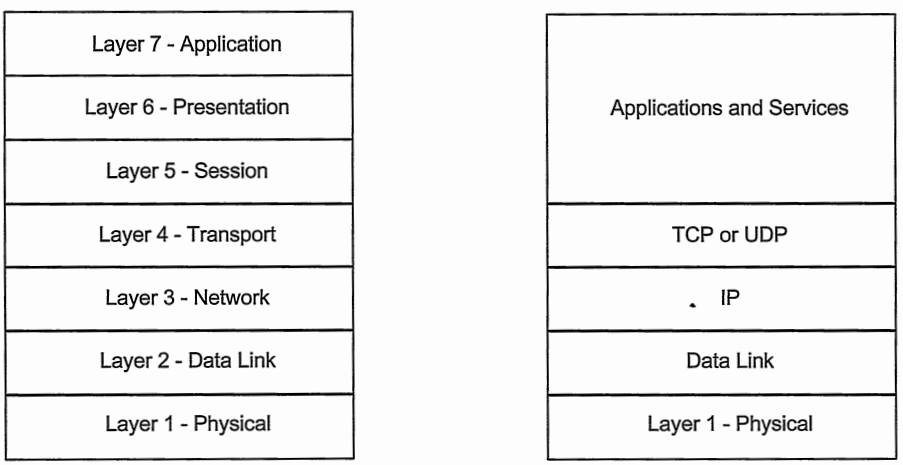


Figure 2-3
The IP suite and the OSI stack

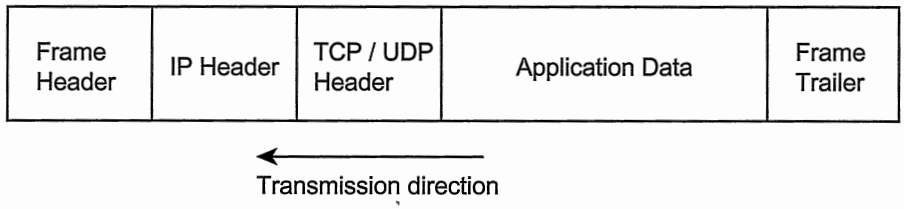


Figure 2-4
Packet structure

destination and are presented in the correct order to the application. UDP is a simpler protocol aimed mainly at one-shot transactions, where a packet is sent and a response received. It does not include sequencing and retransmission functions. Above TCP or UDP, we find a number of different applications and services. The application or service itself determines whether TCP or UDP is used.

Taking these facts into account, Figure 2-4 shows how an IP packet would appear when being passed from one node in a network to another. The application data itself is passed down to the transport layer, where a TCP or UDP header is applied. This combination is passed to the network layer where an IP header is applied. The packet is then passed down to the datalink layer, where it is packaged into a frame structure for transmission. At the destination end, each layer extracts its own header information and passes the remainder to the layer above until the destination application receives just the original data.

Internet Standards and the Standards Process

As we move forward in our discussion, we will make reference to numerous Internet specifications and standards. Before doing so, it is worth discussing Internet standards in general, including naming conventions and the standardization process itself. This information will help to put some of the specifications in a better context.

The Internet Society

The Internet Society is a nonprofit organization with the overall objective of keeping the Internet alive and growing. Its formal mission statement is

"To assure the open development, evolution, and use of the Internet for the benefit of all people throughout the world."

To meet this goal, the Internet Society, among other things, undertakes the following efforts:

- Supports the development and dissemination of Internet standards.
- Supports research and development related to the Internet and internetworking.
- Assists developing countries in implementing and evolving Internet infrastructure and use.
- Forms liaisons with other organizations, government bodies, and the public for education, collaboration, and coordination related to Internet development.

A great many of the solutions described in this book have been developed under the auspices of the Internet Society. It is the Internet Society that is the copyright holder for those specifications. A copy of the copyright statement is included with all standards developed under the auspices of the Internet Society.

The Internet Architecture Board (IAB)

The IAB is the technical advisory group of the Internet Society. It provides technical guidance to the Internet Society, oversees the Internet standards process, and provides a liaison with other organizations on behalf of the Internet Society in technical matters.

The Internet Engineering Task Force (IETF)

The IETF comprises a huge number of volunteers, who cooperate in the development of Internet standards. These individuals come from equipment vendors, network operators, and research institutions and generally have significant technical expertise. They perform the detailed technical work of developing Internet standards.

The work of the IETF is mainly done within working groups, formed to address specific areas or topics (such as security or routing). Most of the work is done by individuals in the working groups who share their work with others in the group through the use of mailing lists. The IETF as a whole meets three times a year.

The Internet Engineering Steering Group (IESG)

The IESG is responsible for management of the IETF's activities. In particular, it is the IESG that approves whether a particular specification becomes an official standard or can even advance within the standards process.

The Internet Assigned Numbers Authority (IANA)

The IANA is responsible for the administration of unique numbers and parameters used in Internet standards. Whenever a new standard proposes to utilize parameters that have specific values and specific meanings associated with those values, then those parameters must be registered with the IANA.

The Internet Standards Process

The process for developing an Internet standard is documented in RFC 2026. The following is meant simply as an overview of what is described in RFC 2026.

An Internet standard begins its life as an Internet draft, which is just the early version of a specification that can be updated, replaced, or made obsolete by another document at any time. It is stored in the IETF's Internet Drafts directory, where it can be viewed by anyone who has web access. If an Internet draft is not revised within six months or has not been recommended for publication as a *Request for Comments* (RFC), it is removed from the directory and officially ceases to exist.

Any reference made to an Internet draft must be done with great care and with emphasis that the draft is a work in progress and that it can change at any time without warning.

Once an Internet draft is considered sufficiently complete, it may be published as an RFC, with an RFC number. Publication as an RFC does not mean that the specification is a standard, however. In fact, several steps need to be taken along the "standards track" before a draft becomes a standard. The first step is where the RFC becomes a proposed standard. A proposed standard is a specification that is stable, complete, and well understood and that has garnered significant interest within the Internet

community. It is not a strict requirement that the specification be implemented and demonstrated before becoming a proposed standard, but such implementation and demonstration may be required by the IESG in the case of particularly critical protocols.

The next step is where the RFC becomes a draft standard. To achieve this status, there must have been at least two independent successful implementations of the specification and interoperability must have been demonstrated. If any portion of the specification has not met the requirement for independent and interoperable implementations, then such a portion must be removed from the specification before it can be granted draft standard status. In this way, a high level of confidence will support the details of the specification.

An RFC becomes a standard once the IESG is satisfied that the specification is stable and mature and can be readily and successfully implemented on a large scale. This step requires that the specification be mature and that there be significant operational experience in using it. Once an RFC becomes a standard, it is given a *standard* (STD) number. The specification retains its original RFC number, however.

Not all RFCs are standards, nor do all RFCs document technical specifications. For example, some document *best current practices* (BCPs), which generally outline processes, policies, or operational considerations related to the Internet or related to the Internet community itself. Others are known as *applicability statements*, which describe how a given specification may be utilized to achieve a particular goal or how different specifications can work together. STD 1 lists the various RFCs and indicates whether they are standards, draft standards, proposed standards, or applicability statements. The list is updated periodically.

The Internet Protocol (IP)

IP itself is specified in RFC 791, with amendments according to RFCs 950, 919, and 920. RFCs 1122 and 1123 give requirements for Internet hosts in their use of IP, and RFC 1812 documents requirements for IP routers (at least for IP version 4).

IP is a protocol for routing packets through a network from origination to destination. A packet of data is equipped with an IP header, which contains information about the originator and the destination address. The data packet with the IP header is known as an IP datagram. The information in the header is used by routers to route the packet to its destination.

IP itself makes no guarantees that a given packet will be delivered, let alone that multiple packets will be delivered in sequence and without omission. IP is known as a "best-effort" protocol. This means that a packet should be delivered under normal circumstances, but that a packet may be lost along the way due to transmission errors, congestion in buffers or transmission facilities, link failures, and so on.

The IP Header

The IP header appears as in Figure 2-5. Although all the fields in the header are there for a reason, a number are of particular importance. The first of these is the version. The currently deployed version of IP is 4 and the header shown in Figure 2-5 applies to IP version 4. The next version to be deployed is version 6.

The Identification, Flags, and Fragment Offset fields are used for cases where a datagram needs to be split into fragments in order to traverse a link, particularly in the case where the size of the datagram is greater than the maximum that can be handled by a given link. The Identification field is used to identify datagram fragments that belong together. The Flags field indicates whether a datagram may be fragmented or not, and in the case of

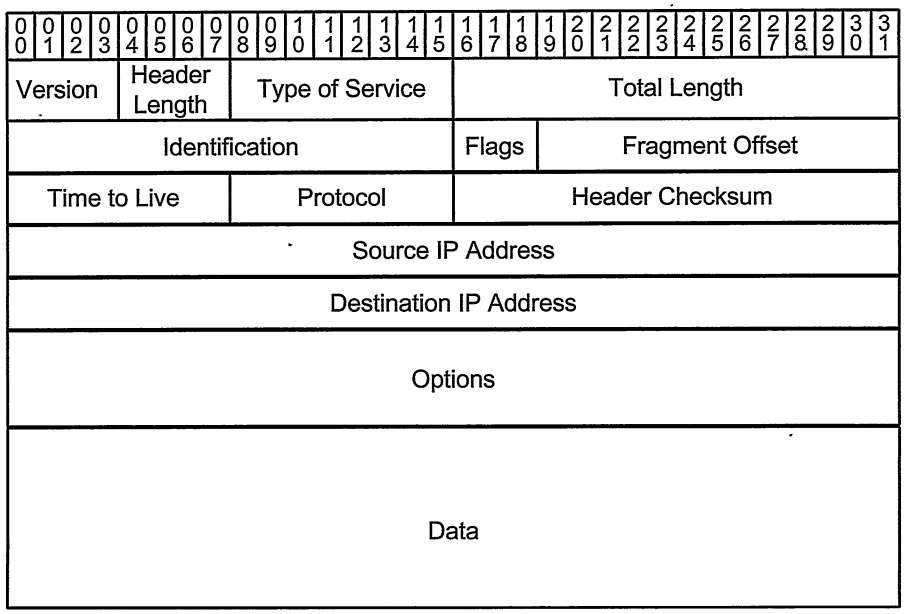


Figure 2-5
IP datagram format

a fragmented datagram, it is used to indicate whether a particular fragment is the last fragment. The fragment offset is a number describing where the fragment belongs in the overall datagram, thereby enabling the destination to put the different pieces together correctly.

The *Time to Live* (TTL) is an indication of how long a datagram should exist in the network before being discarded. Although this field was originally meant to represent a number of seconds, it actually represents a number of hops. It is set by the originating node and is decremented by one at each subsequent node. If the datagram does not reach its destination by the time the TTL reaches zero, then the datagram is discarded.

The Protocol field indicates the higher-layer protocol or application to which this datagram belongs. For example, at the destination node, the value of this field indicates whether this is a TCP or UDP datagram or something else. The protocol value for TCP is 6. The protocol value for UDP is 17.

The source and destination IP addresses in IP version 4 are 32 bits (4 octets) long. An IP address is represented in dotted decimal notation as XXX.XXX.XXX.XXX, with the maximum possible value being 255.255.255.255.

IP Routing

The main function of IP is the routing of data from source to destination. This routing is based upon the destination address contained in the IP header and is the principal function of routers.

Routers can contain a range of different interfaces—Ethernet, Frame Relay, *Asynchronous Transfer Mode* (ATM), and so on. Each interface will provide a connection to another router or perhaps to a LAN hub or even to a device such as a PC. The function of the router is to determine the best outgoing interface to use for passing on a given IP datagram, that is, to determine the next hop. The decision is made based on the content of the routing tables, which basically match destination IP addresses with interfaces on which the datagrams should be sent.

Of course, routers do not have every possible IP address loaded within them. Instead, a match is made between the first portion of the IP address and a given interface. When we say the first portion of the IP address, how much of the address does that mean? The answer varies. For example, one table entry might specify that any address starting with 182.16.16 should be routed on interface A, whereas another table entry might specify that any address starting with 182.18 should be routed on interface B. In the

first case, the IP route mask is 255.255.255.0, while in the second case the IP route mask is 255.255.0.0.

The IP route mask is basically a string of 1's, followed by a string of 0's. A logical AND is performed between the mask and the destination address in the packet to determine if it should be routed over a particular interface. If, after the AND operation, the address of the packet to be routed matches the address in the routing table, then a match occurs and the corresponding interface should be used. For example, if a datagram has a destination address of 182.16.16.7 and a logical AND is performed with mask 255.255.255.0, then the result is 182.16.16.0. This result corresponds to the table entry previously mentioned and the corresponding interface out of the router should be used. If two separate table entries begin with the same prefix, then when a destination address is being compared with the table entries, the entry chosen is the one that provides the longest match with the address in question.

Populating Routing Tables Of course, tables in routers are somewhat more sophisticated than the foregoing suggests, but the basic description still applies and serves the purpose of giving an indication of the large amount of data that might need to be populated in a given router (and, more importantly, kept up to date).

The main issue with routing tables is how to populate them with the correct information in the first place and keep the information current in an environment where new individuals and companies connect to the Internet every day and where new routers are constantly being added. Another issue arises when several paths could lead to a given destination—which is the best one to take?

One could certainly determine from an external source which routing information needs to be populated in routing tables and then add the information manually. However, it would be far better if a given router could somehow automatically keep its own tables up to date with the latest and best routing information. Not only would that take care of the addition of new nodes in the network, but it would also take care of situations where a link to a router is lost and tables need to be reconfigured to send traffic via another router.

Fortunately, mechanisms exist whereby routing tables can be populated automatically. The most common such protocol in large networks is known as *Open Shortest Path First* (OSPF), which is termed a link state protocol. OSPF uses a concept known as an *autonomous system* (AS). An AS is a group of routers that share routing information between them. The AS is

further divided into areas, where an area is defined as a set of networks and where a network is defined by an IP address and mask. For example, the address 182.16.0.0 with mask 255.255.0.0 could represent a network. A router will belong to an area if it has direct connectivity to one of the networks in the area.

A special area, Area 0, is known as the backbone area. This area comprises routers that connect the different areas together. As shown in Figure 2-6, certain border routers will exist in a given area and also in the backbone area to enable routing between different areas.

Within an area, every router will have an identical database describing the topology of the area. The topology describes what is connected to what, that is, the best way to get from one point to another. Whenever the topology changes, such as an interface (link) failure, this information is transmitted from the router attached to the failed interface to all other routers in the same area. Thus, each router within an area will always know the shortest path to a given destination in the area. If the router needs to send packets out of the area, then it needs only to know the best border router to use. It is provided with this information by the border router itself,

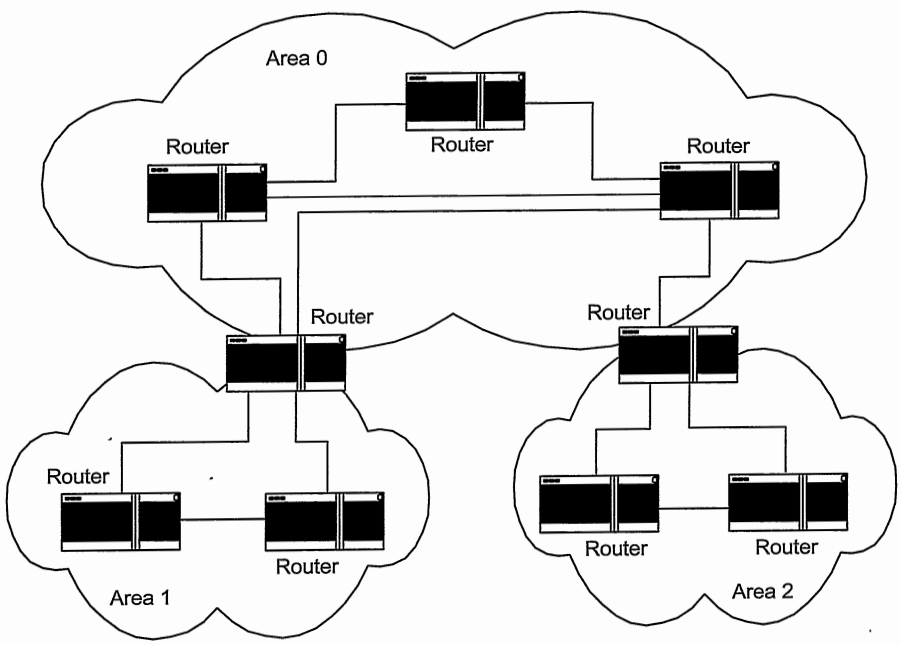


Figure 2-6
OSPF areas

which passes back information regarding the backbone to the routers in its own area and which also passes information about its own area to other routers in the backbone.

Usually associated with OSPF is the *Border Gateway Protocol* (BGP). Recall that an AS is a collection of networks whose routers communicate routing information using a single protocol such as OSPF. The obvious question is how routing information is passed from one AS to another so that a router in one AS knows how to get to an IP address that is outside the AS. BGP is designed to take care of the issue.

Within an AS, a BGP system has visibility of all the networks within the AS. The BGP system, which could be a host or a router, is connected to the BGP system of a neighboring AS and tells the neighboring BGP system about the networks contained in its AS. That system relays the information to the BGP systems of its neighboring ASs and so on. The important part is that each BGP update sent from BGP peer to BGP peer for a given network lists all the ASs that need to be crossed to reach the network in question. Thus, if AS number 6 receives information about the path to network N1, then the information could list AS numbers 5, 4, and 3 as the route that needs to be taken. If AS number 6 is to pass the information on to AS number 7, then the route will specify AS numbers 6, 5, 4, and 3. If a given BGP system receives an update from a neighbor and finds that its own AS number is already in the information that it receives, then a loop has occurred, in which case the information is discarded. The current version of BGP is BGP-4. It is specified in RFC 1771.

The Transmission Control Protocol (TCP)

TCP resides in the transport layer, above the IP layer. Its primary function is to ensure that all packets are delivered to the destination application in sequence, without omissions and without errors. Recall that IP is a best-effort protocol and that packets could be lost in transit from source to destination. The primary function of TCP is to overcome the lack of reliability that is inherent in IP through an end-to-end confirmation that packets have been received. In the case that packets have not been received, then TCP ensures that retransmission occurs. TCP also includes flow control, so that an application at one end does not overwhelm a slower application at the

other end. Applications that use the services of TCP include web browsing, e-mail, and file transfer sessions. TCP is specified in RFC 793.

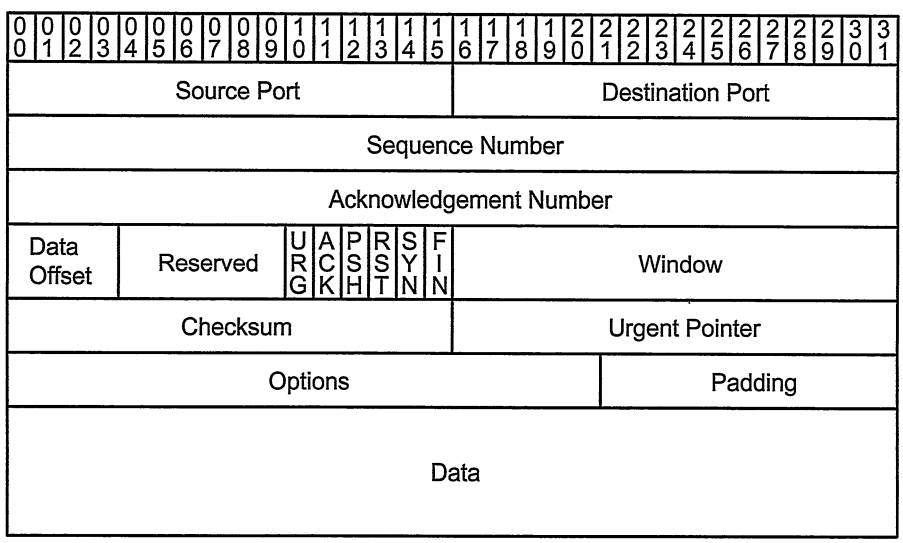
At a high level, TCP works by breaking up a data stream into chunks called segments. Attached to each segment is a TCP header, which includes a sequence number for the segment in question. The segment and header are then sent down the stack to IP, where an IP header is attached and the datagram is dispatched to the destination IP address. At the destination, the segment is passed from IP to TCP. TCP checks the header for errors and, if all is well, sends back an acknowledgement to the originating TCP. Back at the originating end, TCP knows that the data has reached the far end. On the other hand, if the originating end does not receive an acknowledgement within a given timeout period, the segment is transmitted again.

The TCP Header

The TCP header has the format shown in Figure 2-7. The first fields we notice are the source and destination port numbers.

TCP Port Numbers Contrary to what the name implies, a port number does not represent any type of physical channel or hardware. Instead, a port number is a means of identifying a specific instance of a given application.

Figure 2-7
TCP header format



Imagine, for example, a computer where three things are happening at the same time—browsing the web, e-mail, and file transfer. Since each of these uses TCP, how does the computer distinguish between data received for one application versus data received for another? That is where port numbers come into play. Whenever a given application wants to begin a TCP session, a unique port number will be assigned for that particular session. TCP will pass that port number to the far end in any TCP messages relating to that session.

So the near end knows its own port number, but how does it know the port number being used by the far end in advance of any communication with the far end? There are two ways to know this. The easiest is through the use of well-known port numbers. These are port numbers from 0 to 1,023 reserved by the IANA for certain applications. For example, port number 23 is used for TELNET and port number 25 is used for the *Simple Mail Transfer Protocol* (SMTP, used for e-mail).

If a client wants to access a service at a server, and that service is allocated a well-known port number, then that port number will be used as the destination port number. Of course, for a given server, many clients may be trying to access the same type of service. The server can distinguish between the various clients since TCP and IP together include both the destination address and port number, as well as the source address and port number. The combination of all four will be unique. The combination of an IP address and a port number is called a *socket address*.

Sequence and Acknowledgment Numbers The next field we encounter is the sequence number. Although, as the name suggests, a sequence number is used to identify individual segments in a data stream, the sequence number does not actually number the segments. In other words, if a given segment has sequence number 100, then the next sequence will not have sequence number 101. The reason is because the sequence number is actually used to count data octets. The sequence number is a number that is applicable to the first data octet in the segment. Thus, if a given segment has a sequence number of 100 and contains 150 octets of data, then the next segment will have a sequence number of 250.

The acknowledgement number indicates the next segment number that the receiver is expecting. If, for example, a segment has been received with a sequence number of 100 and if the segment contains 150 octets of data, then the acknowledgement returned will contain an acknowledgement number of 250, indicating that the next segment expected should have a sequence number of 250.

Other Header Fields The data offset, also known as the header length, indicates the length of the TCP header in 32-bit words. This field also indicates where the actual user data begins, hence the name given to the field.

The TCP header also contains a number of flags as follows:

- **URG** This is set to 1 if urgent data is included, in which case the urgent pointer field is of significance.
- **ACK** This is set to 1 to indicate an acknowledgement. This flag will be set in all segments sent except for the very first segment sent when establishing communication.
- **PSH** This indicates a push function. This will be set in cases where the data should be delivered promptly to the recipient.
- **RST** Meaning reset, this is used to indicate an error and to abort a session.
- **SYN** Meaning synchronize, this is used in the initial messages when setting up a data transfer.
- **FIN** Meaning finish, this is used to close a session gracefully.

The next field is the Window field. This is used to indicate the amount of buffer space available for receiving data. Data that is received is stored in a buffer until retrieved by the application. When TCP receives data and sends an acknowledgement, it uses the window field to indicate the amount of empty buffer space available at that instant. This enables the sending side to ensure that it does not send more data in the next segment(s) than the receiving end can handle.

The checksum field contains the 1's complement of the 1's complement sum of all 16-bit words in the TCP header, the data, and a pseudoheader, as shown in Figure 2-8. Of course, the pseudoheader is not part of the TCP header, but actually comprises components of the IP header. The checksum field is used to verify that the header and data have been received without error.

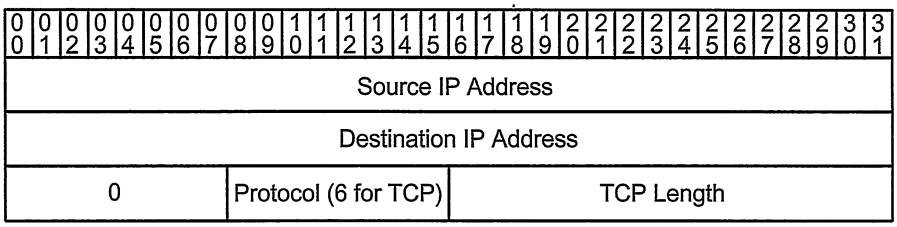


Figure 2-8
TCP pseudoheader

The urgent pointer contains an offset from the current segment number to the first segment after the urgent data, and it indicates the length of the urgent data. This pointer is useful to enable critical information to be sent to the user application as soon as possible.

The only real option defined in RFC 793 is the maximum segment size. This indicates the maximum data size that the receiving system is capable of handling.

TCP Connections

TCP involves the exchange of data streams between a client and a server. This exchange is accomplished through the establishment of a connection. The application at a client decides that it needs to exchange some data with an application at a server. The client application sends data to TCP, where the data is buffered. TCP constructs a TCP segment, including the source and destination port numbers, a sequence number (which is a random number, such as 2,000), the window size, and optionally the maximum segment size. TCP sets the SYN flag to 1 and sends the segment to the far end, as shown in Figure 2-9.

The server receives the segment and responds with a segment of its own. This segment includes the window size, the maximum segment size, a random sequence number of its own (such as 4,000), and an acknowledgement number, which is one greater than the received sequence number (2,001 in our example). The segment has the SYN and ACK flags set.

The client issues another segment containing a sequence number equal to the original sequence number plus 1 (that is, the same as the acknowledgment number received, which is 2,001 in our example). The acknowledgment number in the segment is 1 plus the sequence number received from the server (such as 4,001). The client also sets the ACK flag.

At this point, the TCP connection is open, and both ends are ready to exchange media. If the client were to now send a segment, it would contain a sequence number of 2,001 and an acknowledgment number of 4,001.

During a connection, a server or client will acknowledge the segments received. It is not necessary, however, that every received segment be individually acknowledged. Nor is it necessary for a sender to wait until a given segment is acknowledged before sending another segment. It is only necessary that a given segment is acknowledged before a timeout occurs at the sender of that segment.

For example, let us assume that a client sends 3 segments with segment numbers 100, 200, and 300. Let us also assume that the third of these seg-

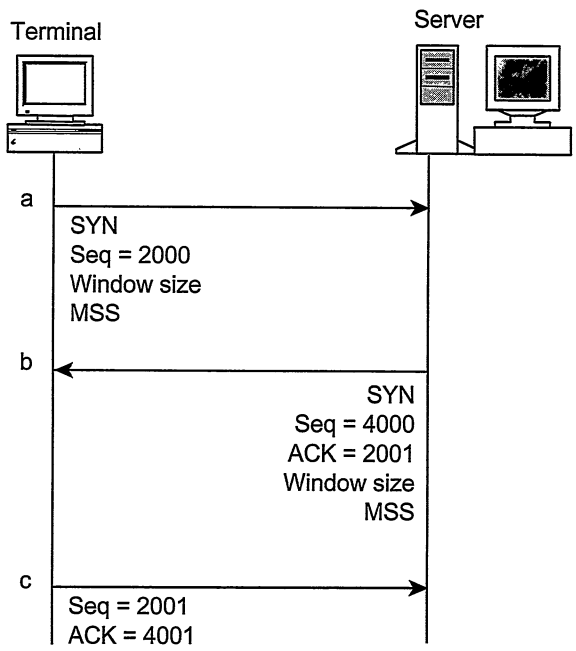


Figure 2-9
Establishing a TCP connection

ments contains 100 octets of data. It is sufficient for the server to send a single acknowledgement for all 3, in which case the acknowledgment number would be 400, indicating the next sequence number expected. This acknowledgement also indicates that all three segments have been safely received.

The client will have a timer running for each segment that has not been acknowledged. In our example, let's suppose that the first of the three segments was lost. In that case, the server would *not* return an ACK, with an acknowledgment number of 400. In fact, let's assume that the server returns nothing. Upon the expiration of the timer at the client, the first segment will be retransmitted. The server could then respond that it has received all 3 segments by sending an ACK, with an acknowledgement number of 400.

Closing a connection is initiated by the sending of a segment with the FIN flag set. Upon receipt of such a segment at the receiver, an ACK will be returned, followed by a FIN. This second FIN (in the reverse direction to the first) will be acknowledged and the connection closure is complete.

The User Datagram Protocol (UDP)

UDP performs a very simple function. At the sending end, it simply passes individual pieces of data from an application to IP to be routed to the far end. At the receiving end, it simply passes incoming pieces of data from IP to the appropriate application.

Unlike TCP, UDP provides no acknowledgment functionality. Therefore, no guarantee exists that anything sent from an application that uses UDP will actually get to the desired destination. UDP is inherently unreliable. Why, then, should there be any interest in using UDP instead of using a more reliable protocol, such as TCP? The answer is that many applications require a quick one-shot transmission of a piece of data or a simple request/response. An example is the *Domain Name System* (DNS), where a client sends a domain name to a domain name server, which responds with an IP address. The establishment and teardown of a TCP connection just for the sake of such a simple transaction would be wasteful and cumbersome.

What happens if a UDP packet is lost? If the application expects a reply to a particular request and does not get one within a reasonable period of time, then the application should retransmit the request. Of course, this could then lead to the far end receiving two requests. In such situations, it would be a smart thing for the application itself to include a request identifier within the data to enable duplicates to be recognized.

Figure 2-10 shows the format of the UDP header. As is the case for TCP, the header contains source and destination port numbers. The source port number is optional and should be used when the sending application expects a response. If not used, then a value of zero is inserted. One should note that a TCP port number of a given value and a UDP port number of the same value represent different ports. Thus, if two applications are running at the same time on a given host (one using UDP and one using TCP), both applications can use the same port number without risk of data being sent to the wrong application.

In the UDP header, we also notice a length field that gives the total length of the datagram, including the header and the user data. The checksum gives the 16-bit 1's complement sum of the header, the data, and a

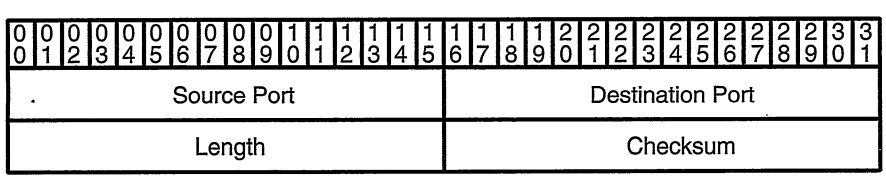


Figure 2-10
UDP header format

pseudoheader. The pseudoheader is not actually part of the UDP header or data at all, but it includes the IP header fields, Source IP Address, Destination IP Address, and UDP Length and Protocol, as shown in Figure 2-11. The checksum is simply a means of checking that the received data is error-free.

Voice over UDP, not TCP

When voice is to be carried on IP, it is UDP that is used rather than TCP. Why should UDP, an inherently unreliable protocol, be chosen over TCP, a protocol than ensures ordered delivery without loss of data? The answer is related to the characteristics of speech itself. In a conversation, the occasional loss of 1 or 2 packets of voice, while not very desirable, is certainly not a catastrophe, since voice traffic will generally use small packets (10 to 40 milliseconds in duration). Obviously, modern voice-coding techniques operate better if no packets are lost, but the coding and decoding algorithms do have the capability to recover from losses, such that the occasional lost packet does not cause major degradation.

On the other hand, voice is very delay sensitive. Anyone who has made a phone call over a satellite connection, where the round-trip time may be many hundreds of milliseconds, knows how annoying a delay can be. Unfortunately, the connection setup routine in TCP and TCP's acknowledgment routines introduce delays, which should be avoided. Even worse, in the case of lost packets, TCP will cause retransmission and thereby introduce even more delay. Tolerating some packet loss is far better than introducing a significant delay in voice transmission.

So how much packet loss is tolerable? Packet loss of about 5 percent is generally acceptable, provided that the lost packets are fairly evenly spaced. A problem can arise, however, if groups of successive packets are lost. This problem can be alleviated if resource management and reservation techniques are used. These techniques generally involve the allocation of a suitable amount of resources (bandwidth and buffer space) before a call

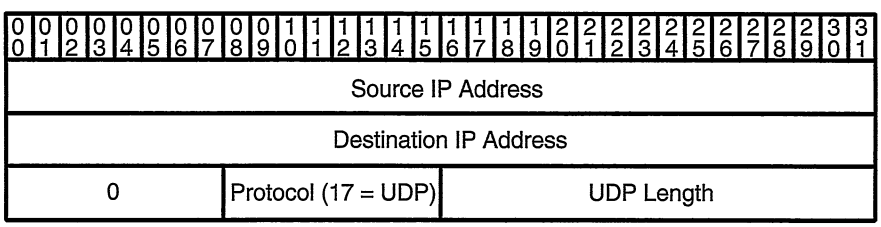


Figure 2-11
UDP pseudoheader

is established. We discuss these issues in some detail in Chapter 8, "Quality of Service." Remember that we focus here on VoIP as an alternative to traditional circuit-switched voice. We are not necessarily addressing voice over the Internet. Although it might be unreasonable to expect controlled access and resource reservation on a grand scale on the Internet, such an expectation is far more reasonable in a managed IP network.

What if packets arrive at a destination in the wrong order? If the application uses UDP, packets will be passed to the application above regardless of the order of arrival. In fact, UDP has no concept of packet order. However, most packets should arrive in sequence. The reason is because all packets in a given session usually take the same route from source to destination. Of course, it is possible that different packets will take different routes, but this will usually occur only if a link fails or if some other topology change occurs in the network. Although these events do happen, the chances of such an event occurring during a particular session are relatively small. Furthermore, *quality of service* (QoS) management techniques can involve establishing a set path through the network for a given session, thus further reducing the probability of an out-of-sequence arrival of packets. Again, these techniques apply to managed networks and not necessarily to the Internet as a whole.

The foregoing arguments are more related to the problems of carrying voice using TCP than they are to the suitability of UDP for voice. In fact, UDP was not designed with voice traffic in mind, and voice traffic over UDP is not exactly a marriage made in heaven. Voice traffic over UDP just happens to be a better match than voice traffic over TCP. Nevertheless, it would be nice if some of the shortcomings of UDP could be overcome without having to resort to TCP. Fortunately, protocols exist that can help mitigate against some of the inherent weaknesses in UDP when used for real-time applications, such as voice or video. We discuss these protocols in the following sections of this chapter.

The Real-Time Transport Protocol (RTP)

RTP is specified in RFC 1889. The official title for this RFC is "RTP: A Transport Protocol for Real-Time Applications." The reason for the somewhat lengthy title is that the RFC actually describes two protocols: RTP and the *RTP Control Protocol* (RTCP). Between the two, they provide the

network transport services required for the support of real-time applications, such as voice and video.

As previously discussed, UDP does nothing in terms of avoiding packet loss or even ensuring ordered delivery. RTP, which operates on top of UDP, helps address some of these functions. For example, RTP packets include a sequence number, so that the application using RTP can at least detect the occurrence of lost packets and can ensure that received packets are presented to the user in the correct order. RTP packets also include a timestamp that corresponds to the time at which the packet was sampled from its source media stream. The destination application can use this time stamp to ensure synchronized play-out to the destination user and to calculate delay and jitter. Note that RTP itself does not do anything about these issues; rather, it simply provides additional information to a higher-layer application so that the application can make reasonable decisions about how the packet of data or voice should be best handled.

As mentioned, RTP has a companion protocol, RTCP. This protocol provides a number of messages that are exchanged between session users and that provide feedback regarding the quality of the session. The type of information includes such details as the numbers of lost RTP packets, delays, and interarrival jitter. Although the actual voice packets themselves are carried within RTP packets, RTCP packets are used for the transfer of this quality feedback.

Under normal circumstances, whenever an RTP session is opened, an RTCP session is also opened. In other words, when a UDP port number is assigned to an RTP session for the transfer of media packets, a separate port number is assigned for RTCP messages. An RTP port number should always be even and the corresponding RTCP port number should be the next highest number (odd). RTP and RTCP may use any UDP port pair between 1025 and 65,535. However, the ports of 5,004 and 5,005 have been allocated as default ports when port numbers are not explicitly allocated.

The use of RTCP with RTP is not mandatory. Some VoIP implementations include RTCP as an option that can be turned off by the system operator. Turning off RTCP, however, eliminates the quality feedback mechanisms that RTCP provides. For that reason, I recommend that RTCP always be enabled.

RTP Payload Formats

RTP carries the actual digitally encoded voice by taking one or more digitally encoded voice samples and attaching an RTP header so that we have

RTP packets comprising an RTP header and a payload of the voice samples. These RTP packets are sent to UDP, where a UDP header is attached. The combination is then sent to IP, where an IP header is attached and the resultant IP datagram is routed to the destination. At the destination, the various headers are used to pass the packet up the stack to the appropriate application.

Given that many different voice and video-coding standards exist, RTP must include a mechanism for the receiving end to know which coding standard is being used, so that the payload data can be correctly interpreted. RTP does this by including a payload type identifier in the RTP header. The interpretation of the payload type number is specified in RFC 1890, "RTP Profile for Audio and Video Conferences with Minimal Control," which is a companion specification to the RTP specification itself. RFC 1890 allocates a payload type number to various coding schemes and provides high-level descriptions of the coding techniques themselves. The RFC also provides references to the formal specifications for the coding schemes.

Since the initial publication of RFC 1890 as a proposed standard, a number of new coding schemes have become available. For example, although RFC 1890 describes the *Global System for Mobile Communications* (GSM) full-rate coder, it makes no reference to the GSM *enhanced full-rate* (EFR) coder. Therefore, recognizing that a number of new coding schemes have become available, a new Internet draft has been prepared, providing an updated RTP profile. This draft is effectively a revised version of RFC 1890 and includes a number of new coding schemes. At the time of writing this book, this Internet draft has the same title as RFC 1890 and it is planned to supersede RFC 1890 as part of the advancement of RFC 1890 from proposed standard to draft standard status. As with all Internet drafts, it should be read with the understanding that it is a work in progress and subject to change at any time.

Table 2-1 and Table 2-2 list the various RTP audio and video payload types respectively defined in the new profile draft.

As can be seen from Table 2-1 and Table 2-2, some coding schemes are given a specific payload type number. These are known as static payload types. Other payload types are marked *Dyn*, meaning dynamic, and are not allocated any specific payload type number.

The main reason for different types of payload types is historical. RTP has existed for several years. When it was originally developed, separate signaling standards for real-time applications did not exist. Therefore, it was not possible for a sending application to let a receiving application know in advance (and out of band) what type of coding was to be used in a given session. Therefore, specific payload type numbers were allocated to a number of

Table 2-1

Payload types for audio encodings

Payload type	Encoding name	Media type	Clock rate (Hz)	Channels
0	PCMU	Audio	8,000	1
1	Reserved	Audio		
2	G726-32	Audio	8,000	1
3	GSM	Audio	8,000	1
4	G723	Audio	8,000	1
5	DVI4	Audio	8,000	1
6	DVI4	Audio	16,000	1
7	LPC	Audio	8,000	1
8	PCMA	Audio	8,000	1
9	G722	Audio	8,000	1
10	L16	Audio	44,100	2
11	L16	Audio	44,100	1
12	QCELP	Audio	8,000	1
13	Reserved			1
14	MPA	Audio	90,000	1
15	G728	Audio	8,000	1
16	DVI4	Audio	11,025	1
17	DVI4	Audio	22,050	1
18	G729	Audio	8,000	1
19	Reserved			1
20	Unassigned			1
21	Unassigned			1
22	Unassigned			1
23	Unassigned			1
dyn	G726-40	Audio	8,000	1
dyn	G726-24	Audio	8,000	1
dyn	G726-16	Audio	8,000	1
dyn	G729D	Audio	8,000	1
dyn	G729E	Audio	8,000	1
dyn	GSM-EFR	Audio	8,000	1
dyn	L8	Audio	Variable	Variable
dyn	RED	Audio	Conditional	1
dyn	VDVI	Audio	Variable	1

Table 2-2 Payload types for video encodings

Payload type	Encoding name	Media type	Clock rate (Hz)
24	Unassigned	Video	
25	CelB	Video	90,000
26	JPEG	Video	90,000
27	Unassigned	Video	
28	nv	Video	90,000
29	Unassigned	Video	
30	Unassigned	Video	
31	H261	Video	90,000
32	MPV	Video	90,000
33	MP2T	Audio/video	90,000
34	H263	Video	90,000
35–71	Unassigned		
72–76	Reserved	N/A	N/A
77–95	Unassigned		
96–127	Dynamic		90,000
Dyn	BT656	Video	90,000
Dyn	H263-1998	Video	90,000

popular coding schemes. When an RTP packet would arrive at a receiver, the payload type number would unambiguously indicate the coding scheme chosen by the sender so that the receiver would know what actions to take to decode the data. That was fine, provided the chosen coding scheme was supported at the receiver. Unfortunately, there was no way for a sender to find out in advance which coding schemes a particular receiver could handle.

In more recent times, separate signaling systems have been developed to enable the negotiation of media streams as part of the call setup process and in advance of actually sending any voice or video. For example, the *Session Initiation Protocol* (SIP) and the *Session Description Protocol* (SDP), both described in Chapter 5, "The Session Initiation Protocol (SIP)," support

such capabilities. Therefore, the two ends of a conversation can now agree in advance on the type of coding scheme to be used. As part of that agreement, they can also agree on the payload type number to be applied to that coding scheme for that call.

For example, let us assume that party A wants to send voice coded according to some coding scheme XXX and party B is able to handle that coding scheme. The parties can agree at the start of the call that RTP packets carrying voice that is coded according to XXX shall have a payload type of 120. When packets arrive with payload type 120, party B knows exactly how to interpret the data. After the call is over, then payload type 120 could be used to indicate some other coding scheme in a subsequent call.

The RTP audio/video profile allocates the payload type numbers 96 to 127 to be used for dynamic payload types. Furthermore, some of the newer coding schemes referenced by the profile specification are not given a fixed (static) payload type, but are considered dynamic. The use of dynamic payload types is very useful. It enables many new coding schemes to be developed in the future without running the risk of exhausting the available payload type number space (a maximum of 127).

Note that the encoding name is also of significance. Whenever a new coding scheme is to be used with RTP, particularly if it is to be a dynamic payload type, then the encoding name should be specified in a separate payload specification document and the encoding name should be registered with the IANA. The reason for this is the fact that external signaling systems such as SDP make use of the encoding name to unambiguously refer to a particular payload specification.

Of the dynamic payload types, one is a little different from all the rest and deserves special mention. This is the *Redundant* (RED) payload type and is specified in RFC 2198. Normally, a single RTP packet contains one or more samples of coded voice or video, with each sample in the packet coded in the same way. If the packet is lost, the receiving application must cope with the loss of data as best it can, such as by replaying the content of a previous packet to fill the void in time. The idea with the redundant payload type is for a single packet to contain one or more voice samples as normal, plus one or more copies of previous samples already sent to the receiver. In other words, each packet contains the current sampled speech, plus a backup copy of the previously sampled speech. If a given packet is lost, then a copy of the lost speech sample arrives in the next packet. Note that the backup copy may use a different coding scheme (more bandwidth-efficient perhaps) than is used for the primary copy. If the backup copy uses a lower-bandwidth coding scheme, then it provides for redundancy to cope with packet loss, but without the need for excessive extra bandwidth.

The RTP Header

As mentioned, RTP carries the actual encoded voice. What this means is that one or more digitally coded samples make up the RTP payload. An RTP header is attached to this payload and the packet is sent to UDP.

The RTP header includes information necessary for the destination application to reconstruct the original voice sample (at least to as close a match as is enabled by the coding scheme used). The RTP header is shown in Figure 2-12.

The meanings of the various header fields are as follows:

- **Version (V)** This field should be set to the value 2, which is the current version of RTP.
- **Padding (P)** This field is comprised of one bit and indicates whether the packet contains one or more padding octets at the end of the payload. Because of the fact that the payload needs to align with a 32-bit boundary, it is possible that the end of the payload contains padding to align with the boundary. If such padding exists, then the P bit is set and the last octet of the padding contains a count of how many padding octets are included. Of course, such padding octets should be ignored.
- **Extension (X)** This field is comprised of one bit and indicates whether the fixed header, as shown in Figure 2-12, contains a header extension. If this bit is set, then the header is followed by exactly one header extension, as described later and with the format shown in Figure 2-13.
- **CSRC Count (CC)** The CC uses a four-bit field indicating the number of contributing source identifiers included in the header and takes a value of 0 to 15. Refer to "Contributing Source Identifiers" described later.

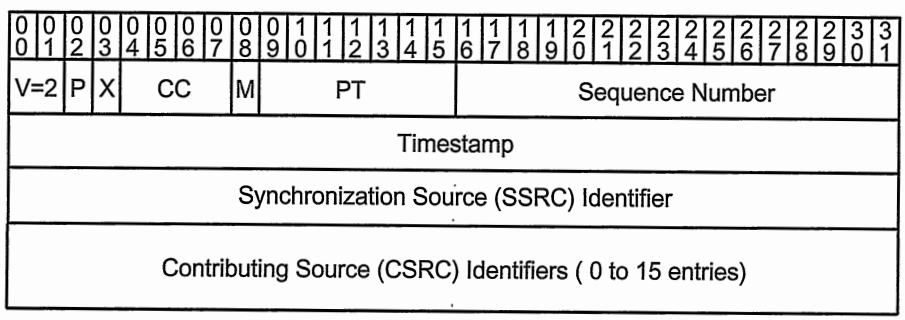


Figure 2-12
RTP header format

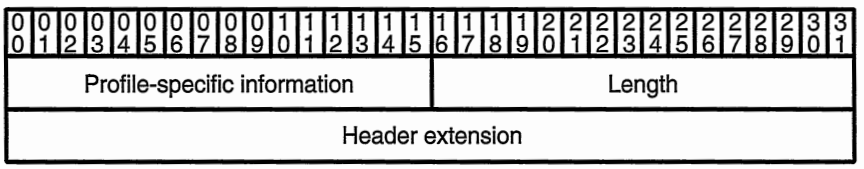


Figure 2-13
RTP header extension

- **Marker (M)** The marker is a one-bit field, the interpretation of which is dependent upon the payload being carried. RFC 1889 does not mandate any specific use of this bit, but RFC 1890 (audio-video profile) does. RFC 1890 states that, for an application that does not send packets during periods of silence, this bit should be set in the first packet after a period of silence (the first packet of a talkspurt). Applications that do not support silence suppression should not set this bit.
- **Payload Type (PT)** This field comprises seven bits and indicates the format of the RTP payload. The content of the PT field is interpreted according to Table 2-1 or Table 2-2. In general, a single RTP packet will contain media coded according to only one payload format. The exception to this rule is the RED profile, where several payload formats may be included. In the case of RED, the payload itself contains information as to how the primary and redundant samples are coded and how they are arranged within the payload. Specifically, the first several octets of the payload contain information about the primary and redundant samples, including the coding schemes used, sample lengths, and timestamp data. The exact number of octets used to convey this information depends on the number of primary and redundant samples contained in the RTP packet. Refer to RFC 2198.
- **Sequence Number** This is a 16-bit field. It is set to a random number by the sender at the beginning of a session and is incremented by one for each successive RTP packet sent. It enables the receiver to detect packet loss and/or packets arriving in the wrong order.
- **Timestamp** This is a 32-bit field indicating the instant at which the first sample in the payload was generated. The sampling instant must be derived from a clock that increases monotonically and linearly in time so that far-end applications may play out the packets in a synchronized manner and so that jitter calculations may be performed. The resolution of the clock needs to be adequate to support synchronized playout. The clock frequency is dependent on the format of the payload data. For static payload formats, the applicable clock

frequency is defined in the RTP profile. For example, the frequency for typical voice-coding schemes is 8,000 Hz.

The increase in the timestamp value from packet to packet will depend on the number of samples contained in a packet. The timestamp is tied to the number of sampling instants that occur from one packet to the next. For example, if 1 packet contains 10 voice samples and a timestamp value of 1, then the next packet should have a timestamp value of 11. Assuming that the sampling occurs at a rate of 8,000 Hz (every 0.125 milliseconds), then the difference of 10 in the timestamp indicates a difference in time of 1.25 milliseconds. If no packets are sent during periods of silence, then the next RTP packet may have a timestamp significantly greater than the previous RTP packet. The difference will reflect the number of samples contained in the first packet and the duration of silence. The initial value of the timestamp is a random number, chosen by the sending application.

- **Synchronization Source (SSRC)** This is a 32-bit field indicating the synchronization source, which is the entity responsible for setting the sequence number and timestamp values and is normally the sender of the RTP packet. The identifier is chosen randomly by the sender so as not to be dependent upon network addresses. It is meant to be globally unique within an RTP session. In the case when the RTP stream is coming from a mixer, then the SSRC identifies the mixer, not the original source of media. Refer to the discussion of mixers and translators later in this chapter.
- **Contributing Source (CSRC)** This is a 32-bit field containing an SSRC value for a contributor to a session. The field is used when the RTP stream comes from a mixer and is used to identify the original sources of media behind the mixer. There may be 0 to 15 CSRC entries in a single RTP header. Refer to the discussion of mixers and translators later in this chapter.

RTP Header Extensions The RTP header is designed to accommodate the common requirements of most, if not all, media streams. Certain payload formats may, however, require additional information. This information may be contained within the payload itself, such as by specifying in the payload profile that the first *n* octets of the payload have some specific meaning. The RED profile uses such an approach. Alternatively, an application can apply an RTP header extension.

The existence of a header extension will be indicated by setting the X bit in the RTP header to 1. If the X bit is set to 1, then a header extension of the

form shown in Figure 2-13 will exist between the CSRC fields and the actual data payload. Note that RTP only requires that the header extension should contain a length indicator in a specific location. Beyond that requirement, RTP does not specify what type of data is to be included in a header extension, or even how long the extension may be. Provided that a length indication exists, it is possible for the boundary between the header extension and the payload to be clearly identified.

Mixers and Translators

Mixers are applications that enable multiple media streams from different sources to be combined into one overall RTP stream. A good example is an audio conference, as depicted in Figure 2-14. The example shows 4 participants, and each is sending and receiving audio at 64 Kbps. If the bandwidth available to a given participant is limited to 64 Kbps in each direction, then that participant does not have sufficient bandwidth to send or receive individual RTP streams to or from each of the other participants. In the example, the mixer takes care of this problem by combining the individual streams into a single stream that runs at 64 Kbps. A given participant does not hear the other three participants individually, but hears the combined audio of everyone.

Another example would be where one of the participants is connected to the conference via a slow-speed connection, while the other participants have a high-speed connection. In such a case, the mixer could also change the data format on the slower link by applying a lower-bandwidth coding scheme. In terms of RTP header values, the RTP packets received by a given participant would contain an SSRC value specific to the mixer and three CSRC values corresponding to the three other participants. The mixer sets the timestamp value.

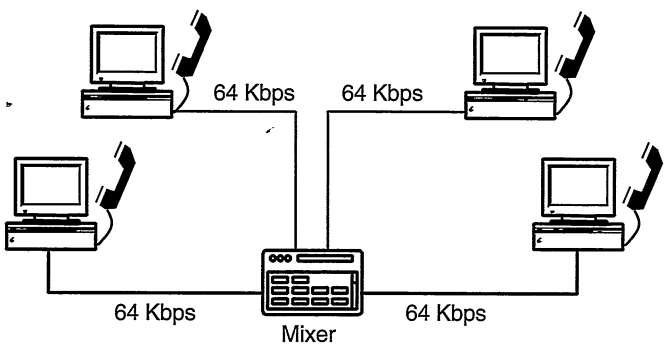


Figure 2-14
An RTP mixer

A translator is used to manage communications between entities that do not support the same media formats or bit rates. An example is shown in Figure 2-15, where participant A can support only 64 Kbps G.711 mu-law encoded voice (payload type 0), while participant B can support only 32 Kbps G.726 *adaptive differential pulse code modulation* (ADPCM) (payload type 2). In terms of RTP headers, the RTP packets received by participant A would have an SSRC value that identifies participant B, not a value identifying the translator. The difference between a mixer and a translator is that a mixer combines several streams into a single stream perhaps with some translation of the payload format, whereas a translator performs some translation of the media format, but does not combine media streams.

The advantage of a mixer over a translator is that, in the case of several streams, extra bandwidth does not need to be allocated for each stream. The disadvantage is that the receiver of the RTP stream cannot mute the stream from one source while continuing to listen to the stream from another.

The RTP Control Protocol (RTCP)

As mentioned previously, RTP comes with a companion control protocol, RTCP. This protocol enables the periodic exchange of control information between session participants, with the main goal of providing quality-related feedback. This feedback can be used to detect and potentially correct distribution problems. By using RTCP and IP multicast, a third party (such as a network operator who is not a session participant per se) can monitor session quality and detect network problems.

RTCP defines five different types of RTCP packets:

- *Sender Report* (SR) is used by active session participants to relay transmission and reception statistics.
- *Receiver Report* (RR) is used to send reception statistics from those participants that receive but do not send media.

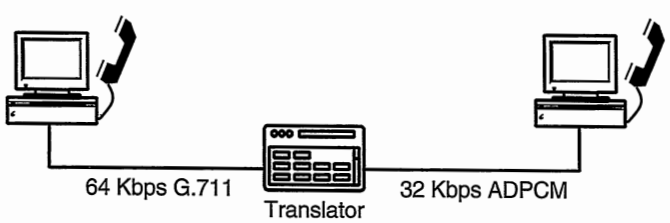


Figure 2-15
An RTP translator

- *Source Description* (SDES) contains one or more descriptions related to a particular session participant. In particular, the SDES must contain a *canonical name* (CNAME), which is used to identify session participants. The CNAME is separate from the SSRC because the SSRC might change in case of a host reset. Furthermore, within a given session, it is possible for a given source to generate multiple RTP streams, as would be the case if both audio and video were being transmitted. In such a case, the two streams would have different SSRC values, but would have the same CNAME value. At a receiver, the CNAME would be used to link the two streams in order to provide a synchronized playout. Within a given session, the CNAME must be unique. Other information may also be within the SDES packet, such as a regular name, e-mail address, or phone number for the participant in question.
- BYE indicates the end of participation in a session.
- APP stands for *application-specific functions*. The APP packet enables RTCP to send packets that convey information specific to a particular media type or application. RTCP does not specify the detailed contents of an APP packet.

Though the previous RTCP packet types are defined, they are never sent individually from one session participant to another. In reality, two or more RTCP packets will be combined into a compound packet. A reason for generating compound packets is that sender and/or receiver reports should be sent as often as practical to allow better statistical resolution. Another reason is that new receivers in a session must receive the identity (CNAME) of various media sources very quickly to allow a correlation between media sources and the received media. In fact, the RTCP specification states that all RTCP packets will be sent as part of a compound packet and that every packet must contain a report packet (SR/RR) and an SDES packet. When a SDES packet needs to be sent, a report packet (SR/RR) is sent even if there is no data to report. In such a case, the compound packet includes an empty RR packet in front of the SDES packet.

An RTCP compound packet would appear as in Figure 2-16. In this particular example, the sender of the compound packet happens to be leaving the session, as indicated by the inclusion of the BYE packet. Note the presence of the optional encryption prefix. If a compound packet is to be encrypted, then a 32-bit encryption prefix must be included in front of the compound packet. The encryption prefix is a random number chosen by the sender of the compound packet. The default encryption algorithm for RTCP is the *Data Encryption Standard* (DES).

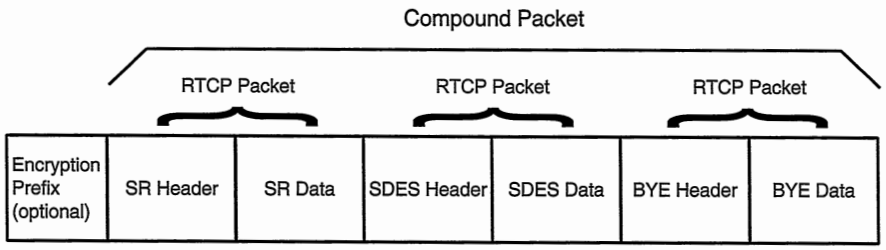


Figure 2-16
Example of an RTCP compound packet

RTCP Sender Report (SR)

The SR is used by session participants who actively send RTP packets as well as receive them. The format of the SR is as shown in Figure 2-17. The SR has three distinct sections: header information, sender information, and a number of receiver report blocks. Optionally, the SR can also have a profile-specific extension.

It can be seen that the RTCP packet format bears quite a resemblance to the format of an RTP packet. As is the case for an RTP packet, we first encounter the version field (V), which has the value 2 for the current version of RTP and RTCP. We also encounter a padding bit (P), which is used to indicate if padding octets are at the end of the packet. If the P bit is set, then the final octet of the packet will contain a count of the number of padding octets. Those octets should be ignored.

The first difference between an RTP packet and an RTCP packet is noticeable in the next field (the RC field), which indicates how many reception report blocks are contained in this report. The RC is a 5-bit field, which means that a maximum of 31 receiver report blocks may be included in a single RTCP sender report packet. What happens in a large conference when a given participant needs to provide receiver report data for more than 31 other session participants? Such as situation is one in which the compounding of RTCP packets is useful, since it is perfectly valid to have a sender report followed by a receiver report in the same RTCP compound packet.

The next field we encounter is the Payload Type field, which has the value 200 for a sender report. This field is followed by a length indicator.

The sender information contains the SSRC of the sender plus some timing information and statistics regarding the number of packets and octets sent by the receiver. Specifically, the sender information includes the following fields:

Motorola - Exhibit 1022 - Page 77

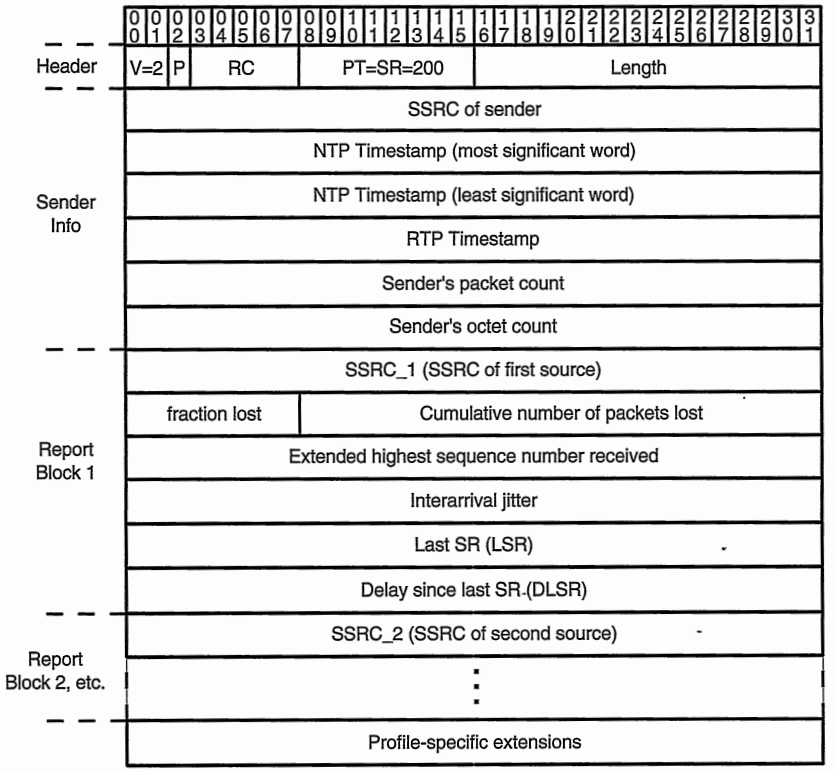


Figure 2-17
RTCP sender report

- **SSRC of Sender**
- **NTP Timestamp** Standing for the Network Time Protocol Timestamp, which is a 64-bit field, indicating the time elapsed in seconds since 00:00 on January 1, 1900 (GMT). The most significant 32 bits represent the number of seconds while the 32 least significant bits represent the fractions of a second, enabling a precision of about 200 picoseconds. Using 32 bits to represent the number of seconds since 1900 might suggest that we will run into difficulty in the year 2036, when the 32 bits used to reflect the number of seconds since January 1, 1900 will wrap around zero. In reality, however, there will not be a problem. For RTP and RTCP purposes, the timestamp is used for relative time measurements. In other words, what matters is the difference between NTP timestamps rather than the absolute value of a single NTP timestamp. Assuming that consecutive RTCP reports are not decades apart, there will not be a problem. The NTP protocol version 3 is specified in RFC 1305.

- **RTP Timestamp** This is a timestamp corresponding to the NTP timestamp but that uses the same units and with the same offset as used for RTP timestamps in RTP packets. Recall that RTP timestamps in RTP packets begin at a random value and increase with each packet sent in units that are dependent on the payload format. Including both the RTP timestamp and the NTP timestamp in the same report enables receivers of the report to better synchronize with the sender of the report, particularly if they share the same timing source.
- **Sender's Packet Count** This is the total number of RTP packets transmitted by the sender from the start of the session up to the time this report was issued. Thus, the packet count is cumulative. The packet count is reset within a given session only if the sender's SSRC value has changed.
- **Sender's Octet Count** This field shows the total number of RTP payload octets transmitted by the sender since the beginning of the session. As is the case for the packet count, the octet count is reset within a session only if the sender's SSRC value changes.

One or more RR blocks follow the sender information. These provide other session participants with information as to how successfully the RTP packets that they have previously sent have been received. For each RR block, the following fields are included:

- **SSRC_n** The source identifier of the session participant to which the data in this particular RR block pertains.
- **Fraction Lost** An 8-bit field indicating what fraction of packets have been lost since the last report issued by this participant. The value is represented as a fixed point number, with the decimal point to the left of the field. The field has 8 bits, with a maximum value of 256. If, for example, the field contained the value 32, it would imply that the fraction lost is 32/256 (12.5 percent).
- **Cumulative Number of Packets Lost** The total number of packets from the source in question that have been lost since the beginning of the RTP session.
- **Extended Highest Sequence Number Received** The sequence number of the last RTP packet received from the source. The low 16 bits contain the last sequence number received. The high 16 bits indicate the number of sequence number cycles, used in the case where the sequence numbers from a particular source cycle through zero one or more times.

- **Interarrival Jitter** An estimate of the variance in RTP packet arrival. See the section on calculating jitter later in this chapter.
- **Last SR Timestamp (LSR)** The middle 32 bits of the 64-bit NTP timestamp used in the last sender report received from the source in question. This field is a means of indicating to the source whether sender reports issued by that source are being received.
- **Delay Since Last SR (DLSR)** The duration, expressed in units of 1/65,536 seconds, between the reception of the last sender report from the source and the issuance of this receiver report block.

RTCP does not define the profile-specific extensions. If they are to be used with a given type of media (a given RTP payload type), then the profile definition of the payload in question should include a specification of the format and usage of any extensions.

RTCP Receiver Report (RR)

The RR is issued by a session participant who receives RTP packets but does not send, or has not yet sent, RTP packets of its own to other session participants. The RR has the format shown in Figure 2-18. The report

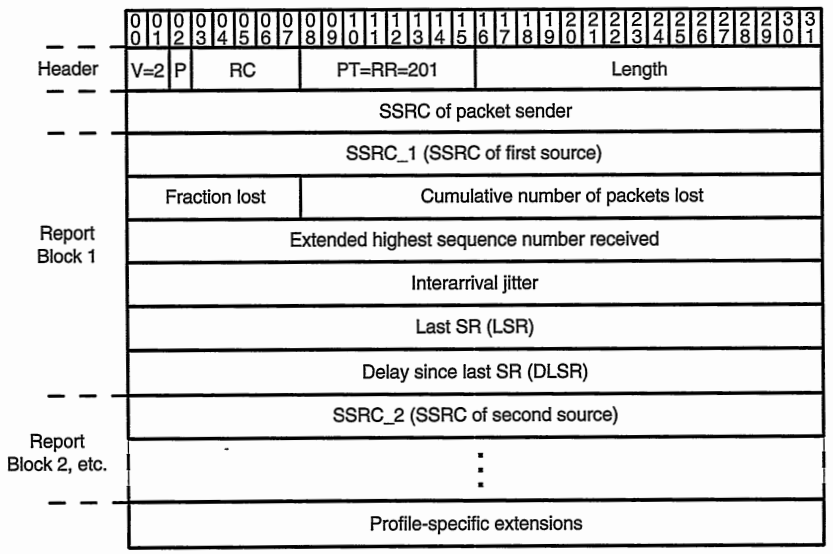


Figure 2-18
RTCP receiver report

packet is almost identical to the sender report with the exception that the payload type has the value 201 and the sender-specific information is not included.

RTCP Source Description Packet (SDES)

The SDES packet provides identification and other information regarding session participants. The SDES packet must exist in every RTCP compound packet. The general format of the SDES packet is shown in Figure 2-19.

The SDES packet is comprised of two main parts: a header and some chunks of information. The header contains a length field, a payload type value field with a value of 202, and a *source count* (SC) field. The SC field occupies five bits and indicates the number of chunks in the packet.

Each chunk contains an SSRC or CSRC value, followed by one or more identifiers and pieces of information that are applicable to that SSRC or CSRC. These pieces of information are known as SDES items and include such items as an e-mail address, phone number, and name (e.g., Jane Doe), among others. All of these SDES items and their structures are defined in RFC 1889. Of the possible SDES items, only one is mandatory—the *canonical name* (CNAME).

The CNAME is meant to be a unique identifier for a session participant and does not change within a given session. Normally, the SSRC value does not change either. It is possible, however, for a given participant's SSRC value to change during a session we find that two participants have chosen the same value of SSRC or if a host resets during a session. In order to ensure that the CNAME is unique, it should have the form user@host,

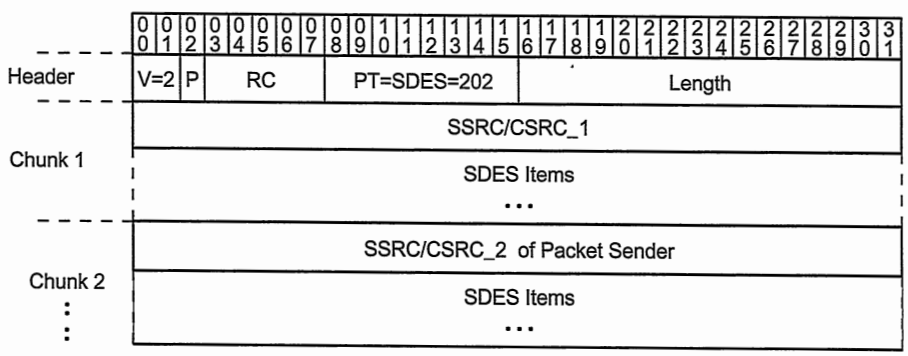


Figure 2-19
SDES packet

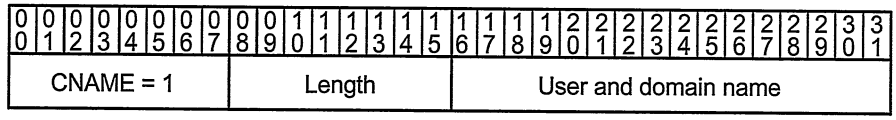


Figure 2-20
CNAME SDES item

where host is a fully qualified domain name. The format of the CNAME parameter is shown in Figure 2-20.

RTCP BYE Packet

The BYE packet is used to indicate that one or more media sources are no longer active. It may also include a text string indicating a reason for leaving the session.

The BYE packet has the format shown in Figure 2-21. The SC field indicates the number of source identifiers included in the packet, that is, the number of SSRC or CRSC entries. After the list of SSRC/CSRC values, an optional text string may be provided, indicating the reason for leaving. In such a case, the text string is preceded by a length field indicating the number of octets in the text string.

Application-Defined RTCP Packet

An analysis of the various RTP and RTCP packet types reveals that RTP and RTCP are written so that packets can be extended with application-specific data. In keeping with this flexible approach, RTCP includes an application-defined packet, with the format shown in Figure 2-22.

Obviously, this type of packet is used for some nonstandardized applications; hence, the format of the application data is not specified. Some identifier is necessary, however, to enable the packet recipient to understand the data; the payload type value of 204 is not sufficient. If, for example, an experimental application were to require two new types of RTCP packets, each carrying different information, then there must be a means to differentiate between the two types.

At this point, the Name field comes into play. This field contains an ASCII string specified by the creator (designer) of the new application and is used to imply a particular coding and significance for the application

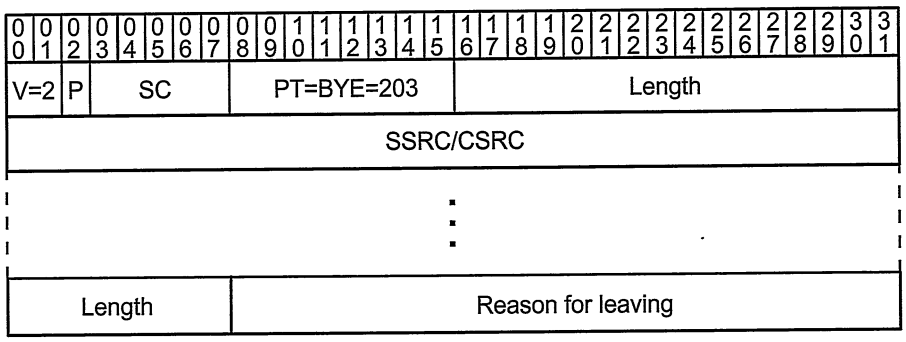


Figure 2-21
RTCP BYE packet

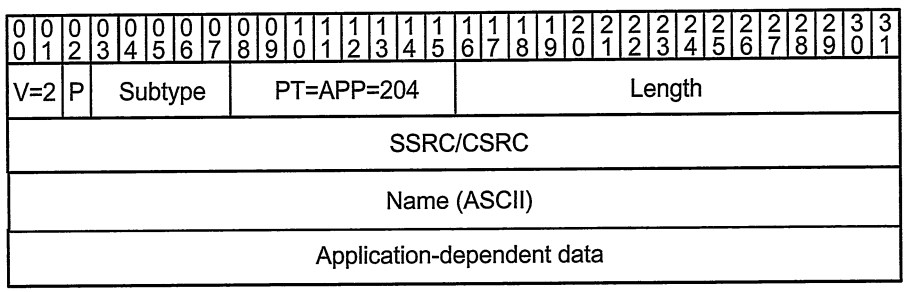


Figure 2-22
RTCP application-defined packet

data. The subtype field is also available for the application creator to indicate the use of any options within the particular application.

Calculating Round-Trip Time

One of the functions enabled by the sender and receiver reports of RTCP is the calculation of round-trip time. This quantity refers to the time that would be taken for a packet to be sent from a particular source to a particular destination and back again. It is a useful metric when measuring voice quality and is calculated from the data in RTCP reports as follows.

Let us assume that participant A issues report A at time T1. This report is received by participant B at time T2. Subsequently, participant B issues report B at time T3 and this report is received by A at time T4. Participant A can calculate the round-trip time as simply

$$T4 - T3 + T2 - T1 \text{ or}$$

$$T4 - (T3 - T2) - T1$$

The data within the RTCP reports either explicitly provide the values for T1, T2, T3, and T4 or provide participant A with sufficient information to determine the values. For example, report B in our example contains an NTP timestamp for when the report is issued (T3). Report B also contains the LSR timestamp, which points participant A to the timestamp included in report A and hence to the value T1. The DLSR timestamp in report B contains the delay between reception by B of report A and the sending from B of report B (T3–T2). Finally, participant A takes note of the instant that report B is received (T4).

Calculating Jitter

As mentioned previously, delay is significant in the perceived voice quality and should be minimized. Delay, however, cannot be eliminated completely. Given that some delay will always exist, the delay should be constant or close to constant. In other words, if a given packet takes 50 milliseconds to get from its source to its destination, then the next packet should take about the same time. Unfortunately, this delay is not always constant, and the variation in delay is known as jitter.

Jitter is defined as the mean deviation of the difference in packet spacing at the receiver compared to the packet spacing at the sender for a pair of packets. This value is equivalent to the deviation in transit time for a pair of packets.

If Si is the RTP timestamp from packet i, and Ri is the time of arrival in RTP timestamp units for packet i, then for the two packets, i and j, the deviation in transit time, D, is given by

$$D(i,j) = (Rj - Ri) - (Sj - Si) = (Rj - Sj) - (Ri - Si)$$

The interarrival jitter is calculated continuously as each data packet, i, is received from a given source SSRC_n, using the difference, D, for that packet and the previous packet, i − 1, in order of arrival according to the formula.

$$J(i) = J(i-1) + (\mid D(i-1,i) \mid - J(i-1))/16$$

The current value of J is included in sender and receiver reports.

Timing of RTCP Packets

RTCP provides useful feedback regarding the quality of an RTP session. RTCP enables participants and/or network operators to obtain information

about delay, jitter, and packet loss and to take corrective action where possible to improve quality. Of course, the better the information, the better the decisions that can be made—and, of course, the quality of the information is linked to how often RTCP reports are issued. Therefore, one would assume that RTCP reports should be sent as often as possible. RTCP packets consume bandwidth, however, and if a large number of RTCP reports are being sent, then these reports themselves contribute to delay, jitter, and RTP packet loss. In other words, too many RTCP packets can make the quality worse. Therefore, we need to strike a balance between the need to provide sufficient feedback and the need to minimize the bandwidth consumed by RTCP packets.

Determining how often to send RTCP packets involves keeping the following considerations in mind. The control traffic should be limited to a small and known fraction of the session bandwidth. The RTCP portion of total bandwidth should be small so that the primary function of the transport protocol to carry data is not impaired. The RTCP portion should be known by all participants, so that the control traffic can be included in the bandwidth specification given to a resource reservation protocol and so that each participant can independently calculate its share. In fact, the total bandwidth consumed by RTCP should be set to 5 percent of the total bandwidth for the session.

RFC 1889 provides an algorithm for calculating the interval between RTCP packets. The algorithm is designed to meet the foregoing criteria. It is also designed to ensure that RTCP can scale from small to large sessions. The algorithm includes the following main characteristics:

- Senders are collectively allowed at least 25 percent of the control traffic bandwidth so that in sessions with a small number of participants, newly arriving participants quickly receive the CNAME for sending participants.
- The calculated interval, which is the mean interval between RTCP packets, is to be greater than a minimum of 5 seconds.
- The interval between RTCP packets varies between 0.5 and 1.5 times the calculated interval. This helps to avoid unintended synchronization where all participants send RTCP packets at the same instant, hence clogging the network.
- A dynamic estimate of the average compound RTCP packet size is calculated, including all those received and sent, to automatically adapt to changes in the amount of control information carried. In other words, if RTCP compound packets start becoming bigger, perhaps

because of more participants, then the rate at which the RTCP packets are sent needs to be adapted to ensure that excessive bandwidth is not consumed.

IP Multicast

Many situations occur where a given IP datagram needs to be sent to multiple hosts. Perhaps the example most applicable to VoIP is a conference, where voice streams from one or more participants need to be distributed to many listeners. Sending packets to each destination individually is possible, but a far better solution is to send the same packet to a single address that is associated with all listeners. This type of solution is known as *multicasting*. The following section provides a brief overview of the concept of multicasting.

Multicasting involves the definition of multicast groups and a given group is assigned a multicast address. All members of the group retain their own IP addresses, but also receive datagrams sent to the multicast address.

In order to support multicasting, a number of requirements must be met. There must be an address space allocated for multicast addresses so that they are not treated as ordinary addresses. Hosts that want to join a particular group must be able to do so and must inform local routers of their membership in a group. Routers must be able to recognize when one or more attached hosts belong to a particular group, as identified by a particular multicast address. Routers must support the capability to forward multicast datagrams correctly. Finally, routing protocols such as OSPF must support the propagation of routing information for multicast addresses. Furthermore, routing tables should be set up so that the minimum number of datagrams is sent.

For example, let's assume that a particular host joins a group and that the host is connected to router A. Let's also assume that other members of the group are connected to router B. Finally, let's assume that some host attached to router C wants to send a datagram to the group. In that case, there needs to be a means for router C to be aware that datagrams for the group's multicast address must be sent to both router A and router B.

As a second example, let's assume that multicast router D exists in the path between router C and routers A and B. In that case, router C should not send two datagrams to routers A and B via router D. Instead, router C should forward a single copy of the datagram to router D. Router D should

then branch out the datagram to routers A and B. In other words, the number of copies of a multicast datagram should be kept to a minimum.

The *IP version 4* (IPv4) address space 224.0.0.0 to 239.255.255.255 is allocated for multicast addresses. Therefore, if a router receives a datagram for such an address, it must be able to route it appropriately. Of these addresses, a number are allocated for special purposes. Examples include the following:

- **224.0.0.1** All hosts on a local subnet
- **224.0.0.2** All routers on a local subnet
- **224.0.0.5** All routers supporting OSPF
- **224.0.0.9** All routers supporting *Routing Information Protocol* (RIP) version 2

Hosts in a particular group use the *Internet Group Message Protocol* (IGMP) to advertise their membership in a group to routers so that multicast datagrams may be forwarded to the appropriate hosts. IGMP version 1 is specified in RFC 1112, while IGMP version 2 is specified in RFC 2236.

Support for multicasting within routing protocols has also been developed. For example, RFC 1584 specifies extensions to OSPF to accommodate the propagation of multicast routing information.

IP Version 6 (IPv6)

The foregoing discussion regarding IP and higher-layer protocols has been based upon IPv4, which is the version of the protocol that is currently deployed throughout the world. The next generation of IP, however, *IP version 6* (IPv6), is starting to be deployed. Accordingly, a brief overview of IPv6 is appropriate here.

Among the biggest drivers for the creation of IPv6 has been the explosive growth of the Internet and IP-based networks in general. It is clear that such growth cannot be sustained indefinitely with an address space of only 32 bits. Furthermore, IP networks are being used more and more for real-time and interactive applications that were not strongly considered in the initial design of IP. IPv6 is designed to address these issues and provide greater flexibility for the support of future applications that are as yet unknown.

Compared to v4, IPv6 offers the following enhancements:

- **Expanded address space** Each address is allocated 128 bits instead of 32.
- **Simplified header format** This enables easier processing of IP datagrams.
- **Improved support for headers and extensions** This enables greater flexibility for the introduction of new options.
- **Flow-labeling capabilities** This enables the identification of traffic flows (and therefore better support at the IP level) for real-time applications.
- **Authentication and privacy** Support for authentication, data integrity, and data confidentiality are included at the IP level rather than through separate protocols or mechanisms above IP.

IPv6 Header

The IPv6 header in Figure 2-23 clearly shows that the header—at least in the basic form shown in Figure 2-23—is less complex than the v4 header. The fields are as follows:

- **Version** The version number is 6.
- **Traffic Class** This field of the IPv6 header corresponds to the *Type of Service* (TOS) field of the IPv4 header. The field comprises 8 bits and is used to enable certain types of traffic to be differentiated from others—and potentially to be given a high priority. For example, real-time applications may be given a higher priority than other applications, such as e-mail. In fact, we will see in Chapter 8 how this field is used to ensure better QoS for real-time applications.
- **Flow Label** This is a 20-bit field, which may be used by a source to label sequences of packets that belong to a single flow. A flow is a sequence of packets sent from a particular source to a particular destination for which the source desires special handling by the intervening routers. For example, a given VoIP stream from one source to another could be considered a flow. The nature of the required special handling for a particular flow might be conveyed to the routers by a control protocol, such as a resource reservation protocol, or by information within the flow's packets themselves. A given flow is identified by a unique combination of source address and a nonzero flow label.

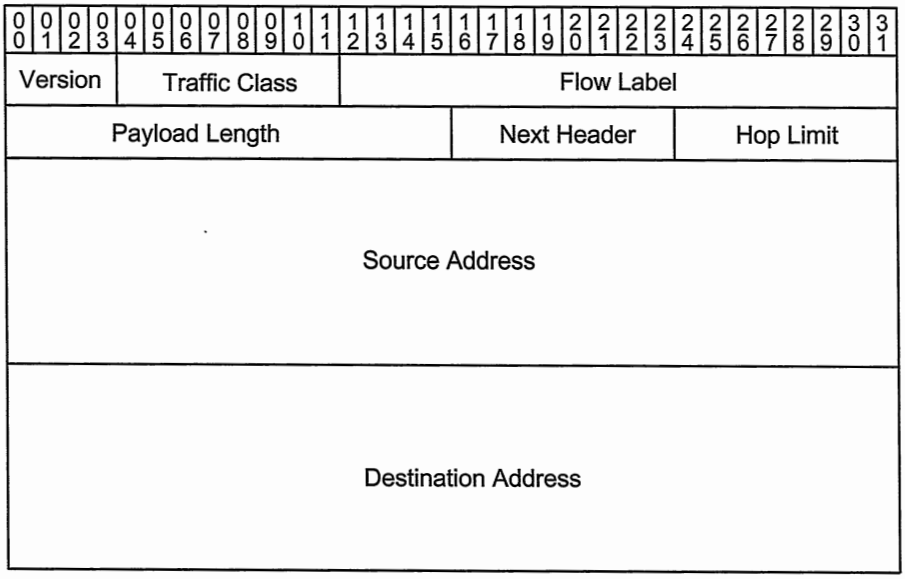


Figure 2-23
IPv6 Header

- **Payload Length** This is a 16-bit unsigned integer indicating the length of the payload in octets. Note that IPv6 enables header extensions to be included after the fixed header shown in Figure 2-23. As far as the length field is concerned, such extensions are part of the payload.
- **Next Header** This field comprises 8 bits, and it serves the same purpose and uses the same values as the Protocol field in the IPv4 header. Thus, the field is used to indicate the next higher layer protocol to which the payload data should be passed at the destination node. The Next Header field is also used to indicate the existence of IPv6 header extensions, as explained later in this chapter.
- **Hop Limit** This field is an 8-bit unsigned integer that indicates the maximum number of hops that an IP packet should experience before being discarded. The value in the Hop Limit field is decremented by 1 at each node that forwards the packet, and the packet is discarded if the value reaches 0. This field is analogous to the TTL field in the IPv4 header.
- **Source and Destination Addresses** Each of these addresses is a 128-bit IPv6 address. The IPv6 addressing structure is described in the following section.

Motorola - Exhibit 1022 - Page 89

IPv6 Addresses

As previously described, IPv4 addresses are generally represented by a dotted decimal notation with a maximum value of 255.255.255.255. IPv6 addresses are significantly different. They have the format XXXX:XXXX :XXXX:XXXX:XXXX:XXXX:XXXX:XXXX, where each X is a hexadecimal character. Thus, the address consists of eight fields, with each field comprising four hexadecimal characters. The maximum possible value for an IPv6 address is therefore FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF :FFFF. Clearly, this value is several orders of magnitude greater than the maximum IPv4 address.

In terms of how we represent an IPv6 address, it is not necessary that leading zeros be included in a given field, but an entry must exist for every field. Therefore, for example, 1511:1:0:0:0:FA22:45:11 is a valid address. The one exception to this rule regarding leading zeros is the case where several contiguous fields have the value zero. In such a case, the symbol "::" can be used to represent a number of contiguous fields with zero values. For example, the address 1511:1:0:0:0:FA22:45:11 could be represented as 1511:1::FA22:45:11. This notation can be applied also in the case where the first several fields contain zero values; the address 0:0:0:0:AA11:50:22:F77 could be represented as: ::AA11:50:22:F77. If the "::" notation appears once in the address, then it is possible to determine the correct content of each of the eight fields of the address. However, that is no longer possible if the "::" notation appears more than once. Therefore, the notation is not allowed to appear more than once in an address. The formatting and usage of IPv6 addresses is specified in RFC 2373.

One important point to note is that the use of IPv6 should not generally require significant changes to upper-layer protocols. An important exception, however, is the use of pseudoheaders when calculating TCP and UDP checksums. Recall that the checksum calculation in UDP and TCP includes a pseudoheader, which incorporates certain fields of the IP header, including the source and destination addresses. Since these are significantly different in IPv6 compared to IPv4, it is important that TCP and UDP checksum calculations be suitably adapted when IPv6 is used as the routing protocol instead of IPv4.

IPv6 Special Addresses IPv6 identifies a number of special addresses. These include the following:

- **The all-zeros address** Meaning an unspecified address, this address is sometimes used when a node does not yet know its address.

This can be represented as either 0:0:0:0:0:0:0:0 or simply :: when using the short notation. The all-zeros address must not be used as a destination address in an IPv6 packet.

- **The loopback address** This is used by a node to send an IPv6 packet to itself. This is an address with the last field set to 1, that is, 0:0:0:0:0:0:0:1 or simply ::1 when using the short notation. This address can be used to route a packet on a virtual internal interface, but it should never be used as a destination address in a packet that is sent outside a physical node.
- **IPv6 address with embedded IPv4 address (type 1)** This is an IPv6 address where the first 96 bits are 0 and the last 32 bits contain an IPv4 address. This type of address is termed an IPv4-compatible IPv6 address and it is used by IPv6 hosts and routers that tunnel IPv6 packets through an IPv4 infrastructure. Such an address can be represented in a manner that indicates the embedded IPv4 address in the familiar dotted decimal format. For example, the address 0:0:0:0:0:0:145.180.15.15, or simply ::145.180.15.15, would be used to represent an IPv6 address with the embedded IPv4 address 145.180.15.15.
- **IPv6 address with embedded IPv4 address (type 2)** This is an IPv6 address where the first 80 bits are 0, the next 16 bits are set to FFFF, and the last 32 bits are set to an IPv4 address. This type of address is used to give an IPv6 address representation to a pure IPv4 address, which would apply to nodes that do not support IPv6 at all. Using the same IPv4 address as in the previous example, the IPv6 representation would be 0:0:0:0:0:FFFF:145.180.15.15 or simply ::FFFF:145.180.15.15.

IPv6 Header Extensions

IPv6 enables extension headers to be placed between the fixed header and the actual data payload. With one exception, these header fields are not acted upon by intermediate nodes and are only considered at the destination node. Recall that the IPv6 header contains a field called Next Header, which is used to indicate the type of payload carried in the IP datagram (and therefore to what upper-layer protocol the packet should be passed at the destination node). For example, if the value in the header is 6, the protocol identifier for TCP, then the payload is a TCP packet and the overall datagram would appear as shown in Figure 2-24.

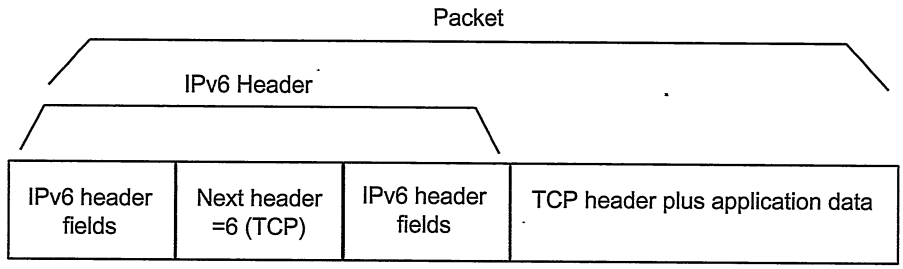


Figure 2-24 Use of the Next Header field

The Next Header field, however, can also be used to indicate the existence of a header extension, which may be used to carry certain optional information. If a header extension exists, then the Next Header value will indicate the type of extension. Furthermore, each extension will have its own Next Header field, which allows for several contiguous header extensions to be included between the IP header and the upper-layer data. The value of Next Header in the IP header will point to the first extension, the value of Next Header in the first extension will point to the second extension, and so on. Finally, the value of Next Header in the last extension will indicate the upper-layer protocol being used. Figure 2-25 shows an example in which the three extensions Hop-by-Hop, Destination Options, and Routing are used with a TCP packet.

Note that with a few exceptions IPv6 itself does not usually specify the details of the various extension headers—at least not the specifics of what optional information and what corresponding functions are to be invoked. Rather, the header extension mechanisms specified by IPv6 offer a means to support applications and requirements that might be as yet unknown.

Hop-by-Hop Extension In general, header extensions are not analyzed by nodes along the delivery path and are meant to be acted upon only by the destination node. The one exception to this rule is the Hop-by-Hop extension, which must be examined and processed by every node along the way. Because of the fact that it must be examined at each node, the Hop-by-Hop extension, if present, must immediately follow the IP header.

The Hop-by-Hop extension is of variable length, with the format shown in Figure 2-26. The Next Header value is used to indicate the existence of a subsequent header extension or the upper-layer protocol to which the packet payload applies. The Length field gives the length of this header extension in units of 8 octets, not including the first 8 octets. The individual options themselves, whatever they might signify, are of the format *type-length-value* (TLV), where the type field is 8 bits long, the length field is 8

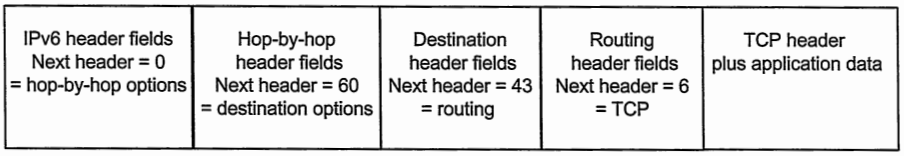


Figure 2-25
Concatenated header extensions

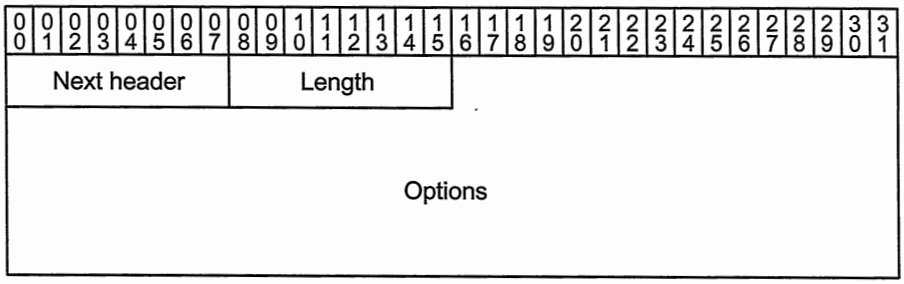


Figure 2-26
IPv6 Hop-by-Hop header extension

bits long, and the data is a variable number of octets according to the length field. This arrangement is shown in Figure 2-27.

Within the Options Type field, the first three bits are of special significance. The first two bits indicate how the node should react if it does not understand the option in question, as follows:

- **00** Skip this option and continue processing the header.
- **01** Discard the packet.
- **10** Discard the packet and send an ICMP Parameter Problem, Code 2 message to the originator of the packet.
- **11** Discard the packet and send an ICMP Parameter Problem, Code 2 message to the originator of the packet, but only if the destination address in the IP header is not a multicast address.

The third bit indicates whether the option data may be changed en-route. If set to 0, then it cannot be changed. If set to 1, then it can change.

The remainder of the Options field is used to carry whatever options the sender of the datagram requires. The IPv6 specification itself does not specify what those options might be.

Destination Options Extension The Destination Options extension has the same format as the Hop-by-Hop extension in that it is of variable length and contains a number of options in the TLV format. These options are to be examined only by the destination node. The Destination Options

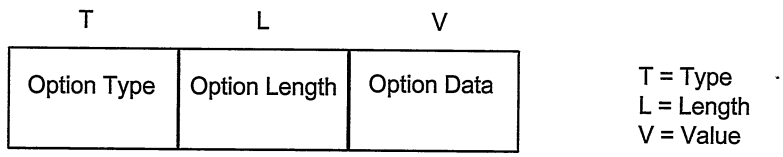


Figure 2-27
Format of options

header extension is identified by the use of the value 60 in the Next Header field in the immediately preceding header.

Routing Extension The Routing header extension includes a Routing Type field to enable various routing options to take place. For the moment, however, IPv6 only defines the routing type 0. The function of this routing type is to specify the nodes that should be visited on the path from the source of a packet to the ultimate destination. When used with routing type 0, the Routing header extension has the format shown in Figure 2-28.

As with all header extensions, the first two fields are the Next Header field and the Length field. The next field is the Routing Type field, which is an 8-bit field indicating a particular variant of the routing header (and which is 0 in the figure). The Segments Left field indicates the number of nodes that still need to be visited before the packet reaches its ultimate destination. Finally, a list of IP addresses that need to be visited along the way is included.

At first glance, one might think that this type of header would need to be analyzed by intermediate nodes between the source IP address and the destination IP address. After all, the purpose of the header is to specify a particular routing, that is which nodes to visit en route. Hence, one would think that this header is another exception to the rule that only destination nodes examine header extensions.

In fact, the way this header is used is quite clever. Let's assume that a packet is to be sent from address A to address Z and must visit addresses B, C, and D in that order along the way. In such a case, the source address in the IP header is address A and the destination address in the IP header is address B. Addresses C, D, and Z are listed in the Route header. Therefore, the node at address B is the destination of the IPv6 packet and as such is required to examine the route header. Upon examination of the header, it swaps address C into the destination address of the IP header, decrements the Segments Left field, and forwards the packet. At address C, the same process is repeated, but this time address D is placed in the destination address of the IP header. The process continues until the packet reaches

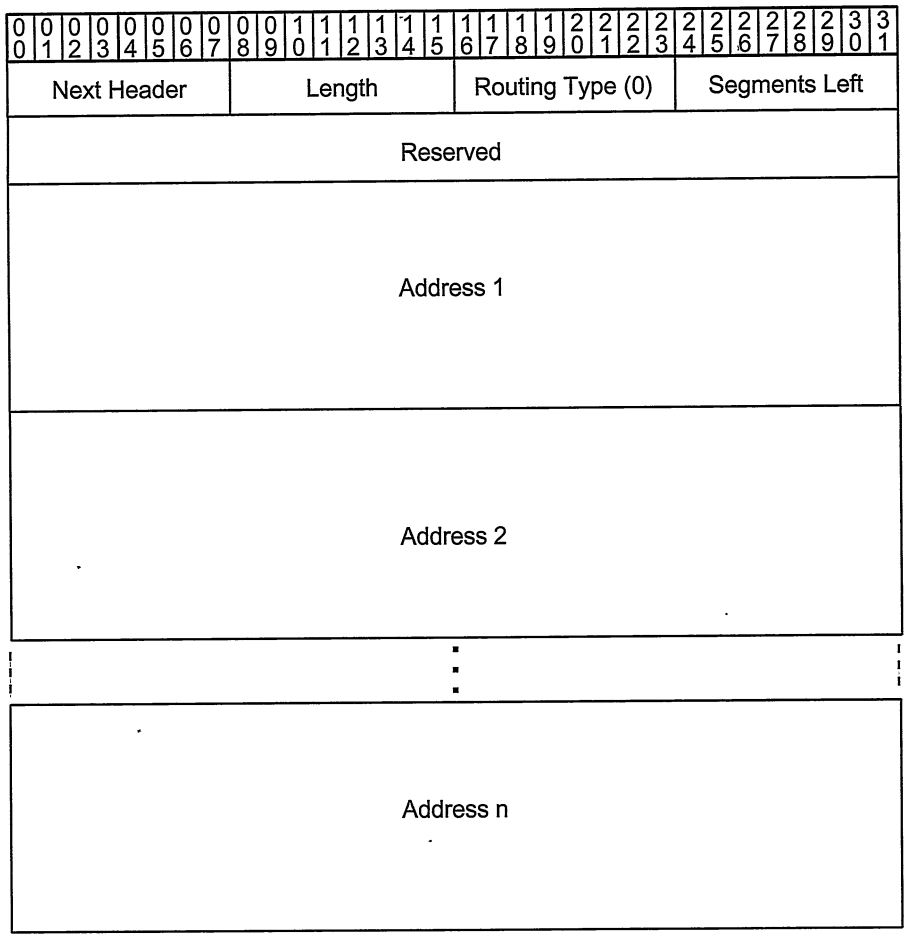


Figure 2-28
Type 0 routing header

address Z, at which point the value in the Segments Left field is 0, indicating that the packet is at its final destination. Furthermore, the Route header now provides a list of nodes visited en route.

Interworking IPv4 and IPv6

IPv6 is the new version of IP, but IPv4 is used everywhere today, and it is not reasonable to expect that all IP networks will be changed from IPv4 to IPv6 at the same time. In fact, it is certain that IPv4 and IPv6 will coexist for a long time and it is likely that IPv6 networks will initially appear as islands in a sea of IPv4. Therefore, a clear requirement exists to support

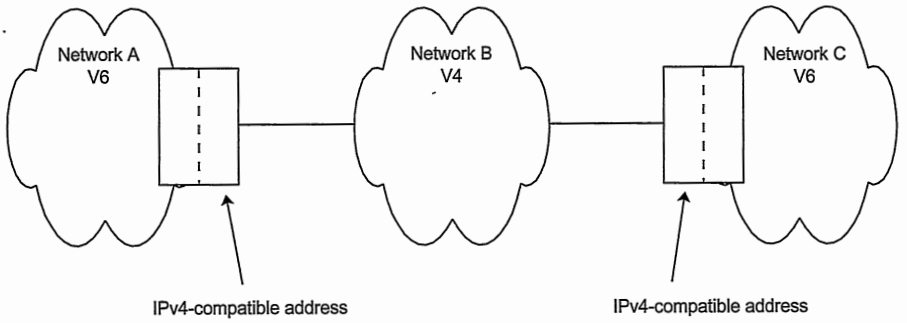


Figure 2-29 Interworking between IPv6 and IPv4

seamless interworking between the two versions. In other words, IPv6 nodes must be able to send datagrams to IPv4 nodes, and IPv4 nodes must be able to send datagrams to IPv6 nodes. Moreover, IPv6 nodes must be able to send datagrams to each other via IPv4 networks, and IPv4 nodes must be able to send datagrams to each other via IPv6 networks.

These requirements can be met by the use of IPv4-compatible addresses. Through the use of IPv4-compatible nodes with IPv4-compatible addresses at the boundaries of IPv6 networks, interworking can be accomplished. Figure 2-29 provides an example of how this is done. By allocating an IPv4-compatible address to the interfaces on an IPv6 network towards the IPv4 network, the IPv6 network can appear as an IPv4 network.

Consider a scenario in which a packet has to go from one IPv6 network to another IPv6 network via an IPv4 network. At the edge of the source network, the packet can be wrapped in an IPv4 header. That header would contain an address for the remote IPv6 network, which would cause the packet to tunnel through the IPv4 network. At the destination network, the IPv4 header is removed, and the remaining IPv6 packet is routed locally within the destination network.

CHAPTER 3

Speech-Coding Techniques

Motorola - Exhibit 1022 - Page 98

Chapter 1, "Introduction," mentions the fact that *Voice over IP* (VoIP) can take advantage of a variety of efficient speech-coding techniques and Chapter 2, "Transporting Voice by Using IP," delves into some detail regarding how digitally encoded speech is carried in an *Internet Protocol* (IP) network by using the *Real-Time Transport Protocol* (RTP). Chapter 2 also makes reference to various RTP payload types, which in turn refer to various speech-coding standards. If it were not already obvious, the discussions in these previous chapters should indicate that speech-coding techniques are of particular significance in VoIP networks. In fact, speech coding is a very important topic.

Though some analog systems still remain, they are rare. These days, voice is digitally encoded and shipped around the network (any network) as a stream of 1's and 0's. Speech/voice coding is simply the process by which a digital stream of 1's and 0's is made to represent an analog voice waveform. The process involves converting the incoming analog voice pattern into a digital stream and converting that digital stream back to an analog voice pattern at the ultimate destination. One of the reasons for implementing VoIP is the opportunity to take advantage of efficient voice coding, where fewer bits are used to represent the voice being transmitted, thereby reducing bandwidth requirements and reducing cost.

The initial conclusion could be that one should implement the most bandwidth-efficient coding scheme possible, thereby saving the most money. Unfortunately, however, there is no such thing as the proverbial "free lunch." As a rough guide, the lower the bandwidth, the lower the quality. So if we reduce the bandwidth significantly, then we run the risk of providing substandard voice quality—not a noble or smart goal and certainly not something that is consistent with a carrier-grade operation.

One should not assume, however, that anything like a linear relationship exists between bandwidth and quality. Another component also effects the relationship. That is the speech-coding scheme used, which relates to processing power.

For example, voice transmission at 64 Kbps sounds better than any voice transmission at 16 Kbps. Several coding schemes might exist, however, that result in a bit rate of 16 Kbps, and one of those techniques might provide better quality than another. You can be certain that the technique that provides the best quality uses a more complex algorithm and hence needs more processing power. Consequently, that technique might be more expensive to implement, emphasizing again that we get nothing for nothing. In the end, the choice of coding scheme is a balance between quality and cost.

In this chapter, we look at some of the issues surrounding speech coding in general and consider some of the coding standards that exist today. Note that this is not by any means an exhaustive analysis of coding standards.

Motorola - Exhibit 1022 - Page 99

It is meant only as an overview of some of the most common standards so that the reader may gain an understanding of some of the technologies involved. Given that this area of telecommunications has undergone great advances in recent years, further advances are likely in the near future.

Voice Quality

We would like to minimize bandwidth while maintaining sufficiently good voice quality. Bandwidth is easily quantified, but how do we quantify voice quality? Surely, this measurement is subjective rather than objective. Standardized methods exist for measuring voice quality, however.

The most common speech quality metric is called the *Mean Opinion Score* (MOS). This measurement scheme is described in *International Telecommunication Union Telecommunications Standardization Sector* (ITU-T) Recommendation P.800. Basically, MOS is a five-point scale as follows:

- Excellent—5
- Good—4
- Fair—3
- Poor—2
- Bad—1

The objective with any coding technique is to achieve as high a ranking on this scale as possible while keeping the bandwidth requirement relatively low. To determine how well a particular coding technique ranks on this scale, a number of people (a minimum of about 30 people and possibly many more) listen to a selection of voice samples or participate in conversations, with the speech being coded according to the technique to be evaluated. They rank each of the samples or conversations according the five-point scale and a mean score is calculated to give the MOS.

On the surface, this does not seem too scientific. Although the basis of the MOS test is a matter of people listening to voice samples, there is more to it than that. ITU-T P.800 makes a number of recommendations regarding the selection of participants, the test environment, explanations to listeners, the analysis of results, and so on. Consequently, different MOS tests performed on the same coding algorithm tend to give roughly similar results. However, because the tests are subjective, variations occur. One test may yield a score of 4.1 for a particular coding algorithm, whereas another test may yield a score of 3.9 for the same algorithm. Therefore, one should

treat MOS values with care, and at the risk of sounding cynical, MOS scores should perhaps be treated with some skepticism depending on who is claming a particular MOS for particular *coder/decoder* (codec).

In general, and within design restrictions such as bandwidth, most coding schemes aim to achieve or approach toll quality. Although the definition of this term can vary, toll quality generally refers to an MOS of 4.0 or higher.

Although MOS is the most common voice quality measurement technique, a number of other methods have been developed. In fact, a great deal of effort has been expended on the development of subjective and objective quality-testing techniques. One of the most common techniques is *Perceptual Speech Quality Measurement* (PSQM) described in ITU-T Recommendation P.861. PSQM is effectively an algorithm that attempts to faithfully represent human judgement and perception processes. The result of PSQM testing is a PSQM score that can be converted to an MOS value.

PSQM testing involves the algorithmic comparison between the output signal of a communications system (such as a voice coder/decoder) and a known input signal. The testing includes an algorithm that determines the combined perceptual effects of a number of variables, such as the type of speaker (male, female, or child), the loudness of the input signal, the delay, the percentage of active/silent speech frames, clipping, and environmental noise. Tests are conducted with a variety of different speech samples that may have been produced artificially or by real speakers. If produced artificially, then the speech samples should be generated according to ITU-T Recommendation P.50.

A number of commercial systems exist for PSQM testing. These test systems are often used in the evaluation of a speech coder during its development and test. A VoIP carrier may also use the test systems in order to choose a coding scheme to use in a network.

A Little about Speech

Speech is generated when air is pushed from the lungs past the vocal cords and along the vocal tract. The basic vibrations occur at the vocal cords, but the sound is altered by the disposition of the vocal tract, that is, by the position of the tongue or the shape of the mouth. The vocal tract can be considered a filter and many codec technologies attempt to model the vocal tract as a filter. As the shape of the vocal tract changes relatively slowly, the transfer function of the filter needs to be changed relatively infrequently

(every 20 milliseconds or so). If the vocal tract can be modeled as a filter, then the excitation of the vibrations at the vocal cords corresponds to the excitation signal applied to the filter.

Speech sounds can be classified into three main types as follows:

- Voiced sounds are produced when the vocal cords vibrate open and closed, thus interrupting the flow of air from the lungs to the vocal tract and producing quasi-periodic pulses of air. The rate of the opening and closing gives the pitch of the sound. This can be adjusted by varying the shape of, and the tension in, the vocal cords and the pressure of the air behind them. Voiced sounds show a high degree of periodicity at the pitch period, which is typically between 2 and 20 milliseconds. This long-term periodicity can be seen in Figure 3-1, which shows a segment of voiced speech sampled at 8 kHz. Here the pitch period is about 8 milliseconds or 64 samples. The power spectral density for this segment is shown in Figure 3-2.
- Unvoiced sounds result when the excitation is a noise-like turbulence produced by forcing air at high velocities through a constriction in the vocal tract while the glottis is held open. Such sounds show little

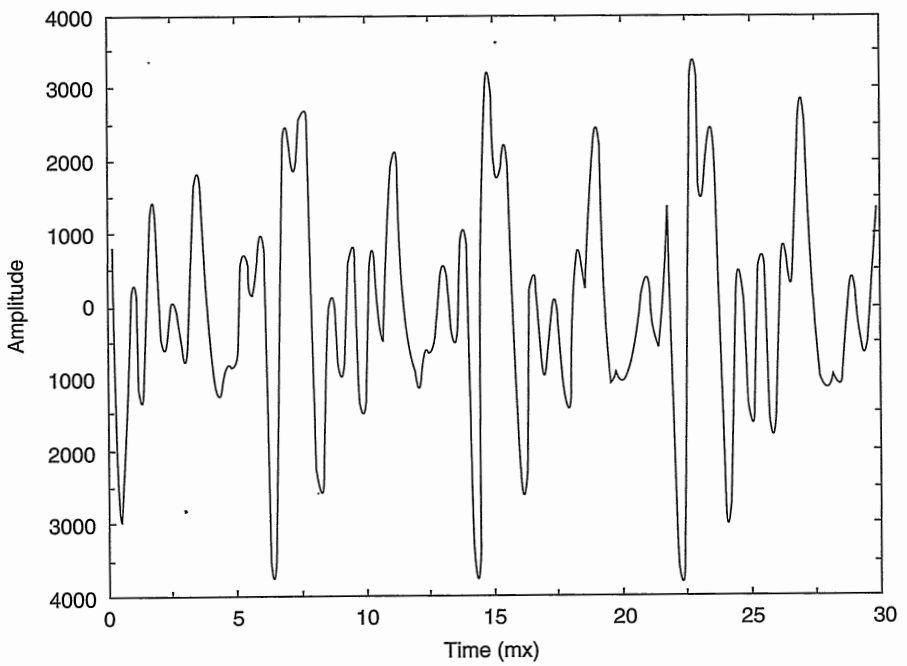


Figure 3-1
Typical segment of voiced speech

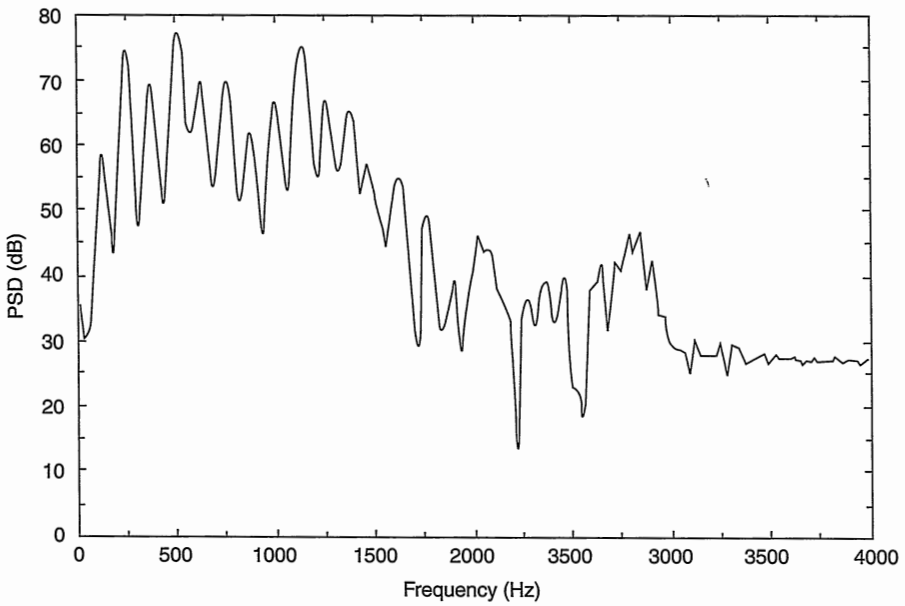


Figure 3-2
Power spectral density for a segment of voiced speech

long-term periodicity, as can be seen in Figures 3-3 and 3-4, although short-term correlations due to the vocal tract are still present.

- Plosive sounds result when a complete closure is made in the vocal tract, and air pressure is built up behind this closure and released suddenly.

Although the combined efforts of the vocal cords and the vocal tract can be made to create a vast array of sounds, the type of vibration at the vocal cords and the shape of the vocal tract change fairly slowly. We can see from the figures that the speech signal is relatively predictable over time. This predictability can be used in the design of voice-coding systems to minimize the amount of data to be transferred and hence reduce bandwidth. For example, if the systems at each end of a conversation can both make reasonably accurate predictions of how the speech signal will change over time, then it is not necessary to transfer the same amount of data as would be required if speech were totally unpredictable. With good voice-coding schemes, leveraging the predictability of speech can lead to a significant reduction in bandwidth, without a huge degradation in perceived quality.

Motorola - Exhibit 1022 - Page 103

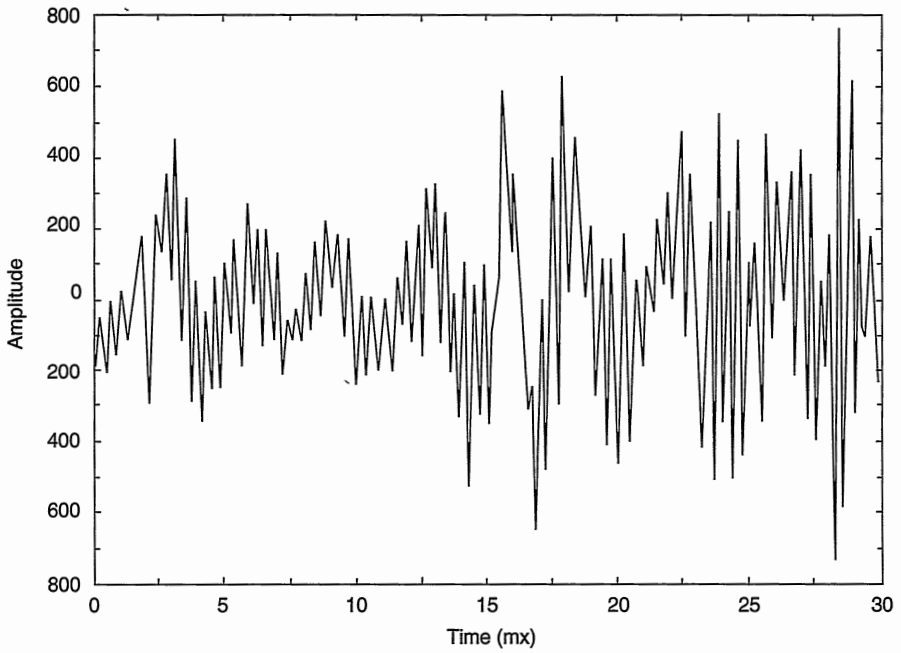


Figure 3-3
Typical segment of unvoiced speech

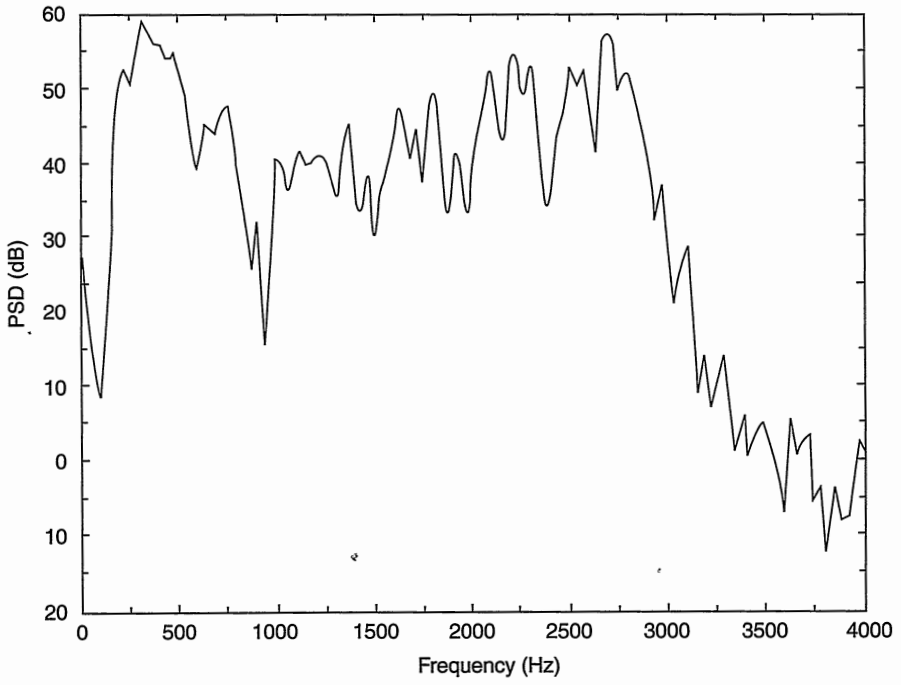


Figure 3-4
Power spectral density for a segment of unvoiced speech

Voice Sampling

In order to create a digital representation of an analog waveform (such as voice), it is first necessary to take a number of discreet samples of the waveform and then represent each sample by some number of bits.

Figure 3-5 gives an example of the sampling of a simple sine wave and what the samples might look like. Obviously, it would take an infinite number of samples in order for those samples to completely recreate the original signal in all respects. Of course, that is not possible, nor is it desirable. Instead, we would prefer to take a sufficient number of samples such that we can use the sample values and suitable mathematics to recreate the original signal (or at least come very close). The Nyquist sampling theorem can help in this regard. This theorem basically states that a signal can be reconstructed if it is sampled at a minimum of twice the maximum frequency of the signal. Thus, if the maximum frequency of a given signal is 4,000 Hz, then we need to take at least 8,000 samples per second.

In fact, this technique is used in all speech codecs. Human speech is generally in the frequency range of 300 and 3,800 Hz. We assume a maximum frequency of less than 4,000 Hz, apply a low-pass filter to take out any frequency components above 4,000 Hz, and then take 8,000 samples per second. Note that this procedure will not properly capture Luciano Pavarotti in full voice or even a less talented individual signing in the shower—just typical conversational speech. For this reason, someone singing on the telephone never sounds quite as good as he or she should, regardless of whether the singer has any talent.

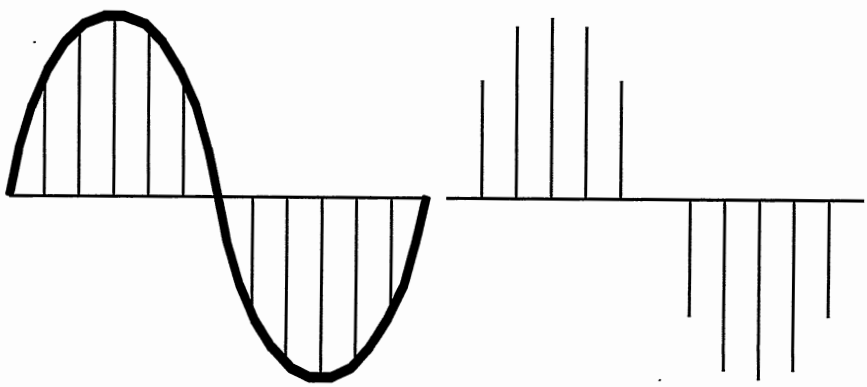


Figure 3-5
Wave sampling

Quantization

So we sample at 8,000 Hz, but the next concern is how to represent each sample. In other words, how many bits are used to represent each sample? The answer is not as simple as one might think, largely because of something called *quantization noise*.

When we take a sample, we have a limited number of bits to represent the value of the sample. By using a limited number of bits to represent each sample, however, we quantize the signal. Figure 3-6 shows an example of quantization, where three bits represent various signal levels. Since we have only three bits, we can represent a maximum of eight different signal levels. If, at a given sampling instant in our example, the analog signal has a value of 5.3, then with our limited number of bits we can only apply the value 5 to the digitized representation. When we transmit the digitized representation to the far end, all that can be recovered is the value 5. The real signal level can never be recovered. The difference between the actual level of the input analog signal and the digitized representation is known as quantization noise.

The easiest way to minimize the effect of quantization noise is to use more bits, thereby providing better granularity. So, if we were to use 6 bits instead of 3, we could at least tell that the sampled value is closer to 5.5

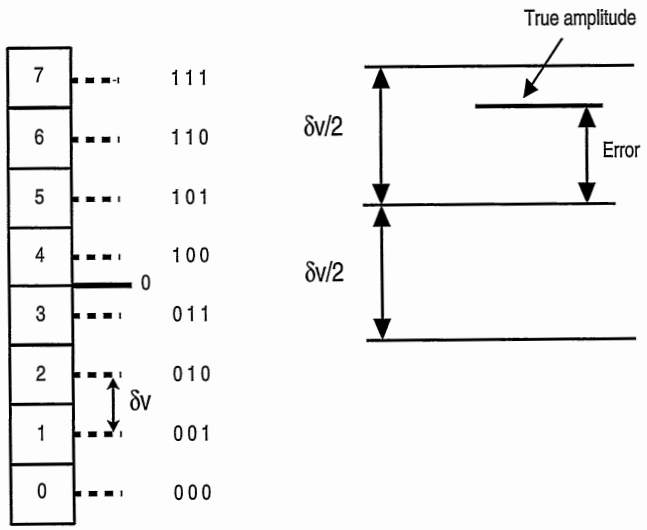


Figure 3-6
A quantization error

than to 5.0 and we could send the bit representation for 5.5. This method can easily approach the stage of diminishing returns. The more accurate we want the samples to be, the more bits we need to represent each sample and the more bandwidth we need. Alternatively, we can use relatively few bits, but we can use them in a more sophisticated manner.

If we look at the possible range of signal levels in speech and if we want to apply uniform quantization levels, then we have two effects. First, we would need many bits to represent each sample, and second, loud talkers would sound better than quiet talkers. The latter is because the effect of quantization noise is less at higher levels than at lower levels. For example, the detrimental effect of digitizing a sample of 11.2 as the value 11 (about 1.8 percent too low) is a lot less than the effect of digitizing a sample of 2.2 as the value 2 (about 9 percent too low). More formally, the signal-to-noise ratio is better for loud talkers than for quiet talkers.

Therefore, we use nonuniform quantization. This approach involves the usage of smaller quantization steps at smaller signal levels and larger quantization steps for larger signal levels. This process gives greater granularity at low signal levels and less granularity at high signal levels. The effect is to spread the signal-to-noise ratio more evenly across the range of different signals and to enable fewer bits to be used compared to uniform quantization. In a coder where the quantization levels are simply sent from one end to the other, this technique translates to a lower bandwidth requirement than for uniform quantization.

Types of Speech Coders

Before we get into descriptions of the various types of codecs available, it is first worth looking at the various categories of codecs. Three types exist: waveform codecs, source codecs (also known as vocoders), and hybrid codecs.

Waveform codecs basically sample and code the incoming analog signal without any thought as to how the signal was generated in the first place. They then transmit quantized values of the samples to the destination end, where the original signal is reconstructed, at least to a very good approximation. Generally, waveform codecs produce a high-quality output and are not very complex. The big disadvantage is that they consume large amounts of bandwidth compared to other codecs. When waveform coders are used at lower bandwidths, the speech quality degrades significantly.

Source codecs attempt to match the incoming signal to a mathematical model of how speech is produced. They usually use a linear predictive filter model of the vocal tract, with a voiced/unvoiced flag to represent the excitation applied to the filter. In other words, the filter represents the vocal tract and the voiced/unvoiced flag represents whether a voiced or unvoiced input is received from the vocal cords. The information that is sent to the far end is a set of model parameters rather than a representation of the signal itself. The far end, using the same modeling technique in reverse, takes the values received and reconstructs an analog signal.

Vocoders operate at low bit rates but tend to produce speech that sounds synthetic. Using higher bit rates does not offer much improvement due to the limitations in the underlying model. Although vocoders are used in private communications systems and particularly in military applications, they are generally not used in public networks.

Hybrid codecs attempt to provide the best of both worlds. Although they attempt to perform a degree of waveform matching, they also utilize knowledge of how people produce sounds in the first place. They tend to provide quite good quality at lower bit rates than waveform coders. Figure 3-7 shows a comparison of the three types of codec with respect to quality and bandwidth.

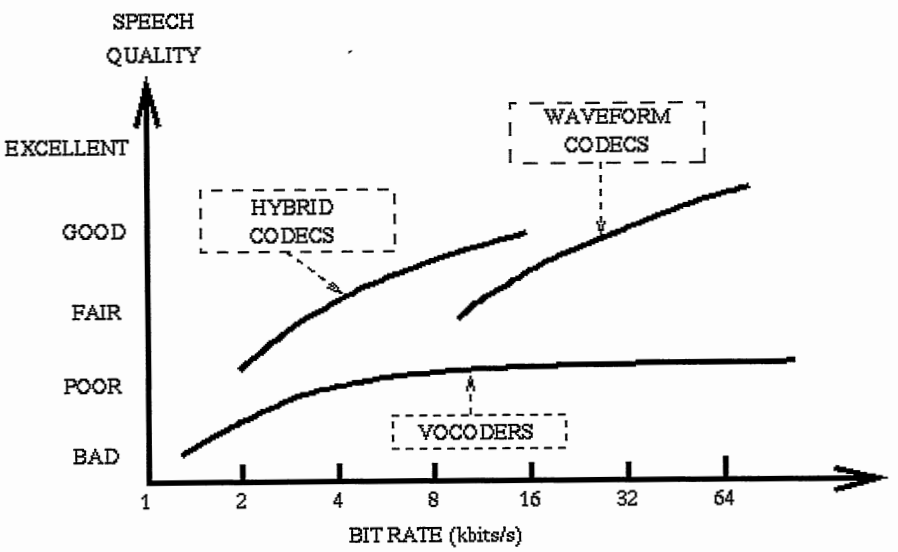


Figure 3-7 Speech quality versus bit rate for common classes of codecs

G.711

G.711 is the most commonplace coding technique used today. It is a waveform codec and is the coding technique that is used in circuit-switched telephone networks all over the world. G.711 has a sampling rate of 8,000 Hz. If uniform quantization were to be used, then the signal levels commonly found in speech would be such that at least 12 bits per sample would be needed, leading to a bit rate of 96 Kbps. Nonuniform quantization is used, however, with 8 bits used to represent each sample. This quantization leads to the well-known 64 Kbps *digital signal 0* (DS0) rate that we have come to know and love. G.711 is often called *pulse code modulation* (PCM).

G.711 has two variants: A-law and μ-law. μ-law is used primarily in North America and A-law is used in most other countries. The difference between the two is the manner is which nonuniform quantization is performed. Both are symmetrical around zero. A-law is skewed to be a little friendlier to lower signal levels than μ-law in that it provides very small quantization intervals for a longer range of low-level signals at the expense of larger quantization levels for a longer range of higher-level signals.

Both A-law and μ-law provide very good quality and have an MOS of about 4.3. The main drawback with G.711, however, is the 64 Kbps bandwidth requirement. Achieving comparable speech quality at a lower bandwidth would be nice.

Adaptive Differential PCM (ADPCM)

PCM codecs such as G.711 send individual samples to the far end, where they are reconstructed to form something close to the original waveform. Since voice changes relatively slowly, however, it is possible to predict the value of a given sample based on the values of previous samples. In such a case, we need only transmit the difference between the predicted value and the actual value of the sample. Because the far end is performing the same predictions, it can determine the original sample value if told the difference between the prediction and the actual sample value. This technique is known as *differential PCM* (DPCM) and can significantly reduce the transmission bandwidth requirement without a drastic degradation in quality. In its simplest form, DPCM does not even make a prediction of the next sample; it simply transmits the difference between sample N and sample

N+1, providing all the information necessary for the far end to recreate sample N+1.

A slightly more advanced version of DPCM is *adaptive DPCM* (ADPCM). ADPCM typically predicts samples' values based on past samples while factoring in some knowledge of how speech varies over time. The error between the actual sample and the prediction is quantized, and this quantized error is transmitted to the far end. Assuming that predictions are fairly accurate, then fewer bits are required to describe the quantized error and hence the bandwidth requirement is lower. The ITU-T Recommendation G.721, which offers ADPCM-coded speech at 32 Kbps, is a good example of ADPCM.

G.721 has now been superseded by ITU-T recommendation G.726, which is a more advanced ADPCM codec. A G.726 coder takes A-law- or μ-law-coded speech and converts it to or from a 16, 24, 32, or 40 Kbps channel. Running at 32 Kbps, G.726 ADPCM has an MOS of about 4.0, which is pretty good.

Both PCM and ADPCM are waveform codecs. As such, they have effectively no algorithmic delay. In other words, the functioning of the algorithm itself does not require that quantized speech samples or quantized error values be held up for a brief time before being sent to the far end. Given that voice is a delay-sensitive form of communication, this absence of algorithm delay is a significant advantage. The disadvantage, of course, is the bandwidth that these codecs consume—especially at the higher end.

Analysis-by-Synthesis (AbS) Codecs

Hybrid codecs attempt to fill the gap between waveform and source codecs. As described previously, waveform coders are capable of providing good-quality speech at higher bit rates. They can even provide relatively acceptable quality at rates down to about 16 Kbps. They are of limited use at lower rates, however. Vocoders, on the other hand, can provide intelligible speech at 2.4 Kbps and lower, but they cannot provide natural-sounding speech at any bit rate. The objective with hybrid coders is to provide good-quality speech at a lower bit rate than is possible with waveform coders.

Although other forms of hybrid codecs exist, the most successful and most commonly used are time domain *Analysis-by-Synthesis* (AbS) codecs. Such codecs use the same linear prediction filter model of the vocal tract as found in *Linear Predictive Coding* (LPC) vocoders. Instead of applying a simple two-state, voiced/unvoiced model to find the necessary input to this

filter, however, the excitation signal is chosen by attempting to match the reconstructed speech waveform as closely as possible to the original speech waveform. In other words, different excitation signals are attempted, and the one that gives a result closest to the original waveform is selected (hence the name AbS). AbS codecs were first introduced in 1982 with what was to become known as the *Multi-Pulse Excited* (MPE) codec. Later, the *Regular-Pulse Excited* (RPE) and the *Code-Excited Linear Predictive* (CELP) codecs were introduced. Variations of the latter are discussed briefly here.

G.728 Low-Delay CELP (LD-CELP)

CELP coders implement a filter, the characteristics of which change over time, and they contain a codebook of acoustic vectors. Each vector contains a set of elements, where the elements represent various characteristics of the excitation signal. This approach is obviously far more sophisticated than a simple voiced/unvoiced flag.

With CELP coders, all that is transmitted to the far end is a set of information indicating filter coefficients, gain, and a pointer to the excitation vector chosen. Because the far end contains the same codebook and filter capabilities, it can reconstruct the original signal to a good degree of accuracy.

ITU-T Recommendation G.728 specifies a *Low-Delay CELP* (LD-CELP coder), which is a backward-adaptive coder because it uses previous speech samples to determine the applicable filter coefficients. G.728 operates on five samples at a time. In other words, the coder takes 5 samples (sampled at 8,000 Hz) and determines a codebook vector and filter coefficients to best match the 5 samples. The choice of filter coefficients is based upon previous and current samples. Because the coder operates on 5 samples at a time, we have a delay of less than 1 millisecond.

Because the receiver also has access to the previous samples and can perform the same determination, all that is necessary to transmit is an indication of the excitation parameters. In fact, an index to the excitation vector is sent. Because there are 1,024 vectors in the codebook (and these are available at both the sender and receiver), the index is only 10 bits long. G.728 uses 5 samples at a time and these samples are taken at a rate of 8,000 per second. For each 5 samples, G.728 needs to transmit only 10 bits. A little arithmetic shows that G.728 results in a transmitted bit rate of 16 Kbps.

The following text is taken directly from the G.728 specification and provides some further information on the coding mechanism. The text and the

content of Figures 3-8 and 3-9 are reproduced here with the prior authorization of the ITU as copyright holder.

> After the conversion from A-law or μ-law PCM to uniform PCM, the input signal is partitioned into blocks of five-consecutive input signal samples. For each input block, the encoder passes each of 1024 candidate codebook vectors (stored in an excitation codebook) through a gain scaling unit and a synthesis filter. From the resulting 1024 candidate quantized signal vectors, the encoder identifies the one that minimizes a frequency-weighted mean-squared error measure with respect to the input signal vector. The 10-bit codebook index of the corresponding best codebook vector (or "codevector"), which gives rise to that best candidate quantized signal vector, is transmitted to the decoder.
>
> The decoding operation is also performed on a block-by-block basis. Upon receiving each 10-bit index, the decoder performs a table look-up to extract the corresponding codevector from the excitation codebook. The extracted codevector is then passed through a gain scaling unit and a synthesis filter to produce the current decoded signal vector.
>
> The decoded signal vector is then passed through an adaptive postfilter to enhance the perceptual quality. The postfilter coefficients are updated periodically using the information available at the decoder. The five samples of the postfilter signal vector are next converted to five A-law or μ-law PCM output samples.

The reason why the G.728 coder is termed LD-CELP is the very small algorithmic delay that it introduces. Due to the fact that it operates on 5 PCM samples at a time and given that these samples are taken at a rate of 8,000 Hz, the coder introduces a delay of 0.625 milliseconds. This delay is so small that it would not be noticeable even to the keenest ear.

The fact that only a 10-bit pointer value is transmitted for every 5 incoming PCM samples means that the actual bandwidth consumed between encoder and decoder is one-quarter of that used by regular PCM (that is, 16 Kbps). Given that the G.728 codec gives an MOS score of about 3.9, the quality reduction is not enormous compared to the bandwidth saved. Figure 3-8 provides a simplified block diagram of the LD-CELP encoder and decoder.

Although the G.728 coder provides high quality at a reasonably low bandwidth, it is quite rare to find the coder used in commercial VoIP products. The reason is the fact that the G.728 coder is quite processor intensive. Consequently, the *digital signal processors* (DSPs) used for G.728 are quite expensive. There are other coders that provide good quality at a low bandwidth and a lower cost than G.728.

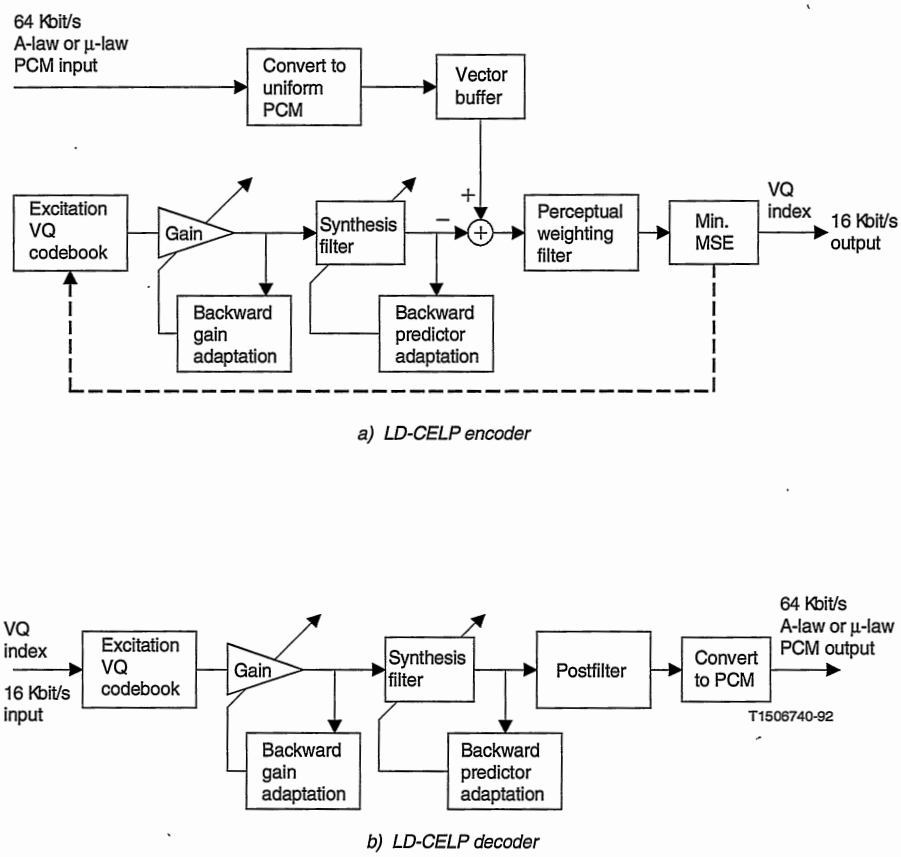


Figure 3-8
A simplified block diagram of the LD-CELP coder

G.723.1 Algebraic Code-Excited Linear Prediction (ACELP)

ITU-T Recommendation G.723.1 specifies a speech coder that can operate at either 6.3 Kbps or 5.3 Kbps, with the higher bit rate providing higher speech quality. Both rates are mandatory parts of the codec and we can change from one mode to another during a conversation.

The coder takes a band-limited input speech signal that is sampled at 8,000 Hz and that undergoes uniform PCM quantization, resulting in a 16-bit PCM signal. The encoder then operates on blocks or frames of 240 samples at a time. Thus, each frame corresponds to 30 milliseconds of speech, which means that the coder automatically causes a delay of 30 milliseconds. The G.723.1 coder also utilizes a look-ahead of 7.5 milliseconds, resulting in

a total algorithmic delay of 37.5 milliseconds. Of course, other small delays will take place within the coder itself as a result of the processing effort involved.

Each frame is passed through a high-pass filter to remove any DC component and then is divided into 4 subframes of 60 samples each. Various operations are performed on these subframes in order to determine the appropriate filter coefficients. *Algebraic Code-Excited Linear Prediction* (ACELP) is used in the case of the lower bit rate of 5.3 Kbps and *Multipulse Maximum Likelihood Quantization* (MP-MLQ) in the case of the higher rate of 6.3 Kbps.

The information transmitted to the far end includes linear prediction coefficients, gain parameters, and excitation codebook index values. The information transmitted comprises 24-octet frames in the case of transmission at 6.3 Kbps and 20-octet frames in the case of transmission at 5.3 Kbps.

Normal conversation involves significant periods of silence (or at least silence from one of the parties). During such periods of silence, it is desirable not to consume significant bandwidth by transmitting the silence at the same rate as speech is transmitted. For this reason, G.723.1 Annex A specifies a mechanism for silence suppression whereby *Silence Insertion Description* (SID) frames can also be used. These are only 4 octets in length, which means that transmission of silence occupies about 1 Kbps. This is significantly better than G.711 where silence is still transmitted at 64 Kbps.

Therefore, three different types of frame can be transmitted by using G.723.1: one for 6.3 Kbps, one for 5.3 Kbps, and an SID frame. Within each frame, the two least significant bits of the first octet indicate the frame size and the codec version in use (as shown in Table 3-1).

G.723.1 has an MOS of about 3.8, which is good considering the vastly reduced bandwidth that it uses. G.723.1 does, however, have the disadvantage of a minimum 37.5-millisecond delay at the encoder. Although this delay is well within the bounds of what is acceptable for good-quality

Table 3-1

G.723.1 frame size and codec version

Bits	Meaning	Octets/frame
00	High-rate speech (6.3 Kbps)	24
01	Low-rate Speech (5.3 Kbps)	20
10	SID frame	4
11		N/A

speech, we must remember that it is round-trip delay that is important, not just one-way delay. Moreover, there will be various other delays in the network, including processing delays and queuing delays such as at routers in a VoIP network, for example.

G.729

The basic ITU-T Recommendation G.729 specifies a speech coder that operates at 8 Kbps. This coder uses input frames of 10 milliseconds, corresponding to 80 samples at a sampling rate of 8,000 Hz. G.729 also includes a 5-millisecond look-ahead, resulting in an algorithmic delay of 15 milliseconds (significantly better than G.723.1). From each input frame, the coder determines linear prediction coefficients, excitation codebook indices, and gain parameters. These pieces of information are transmitted to the far end in 80-bit frames. Given that the input signal corresponds to 10 milliseconds of speech and results in a transmission of 80 bits, the transmitted bit rate is 8 Kbps. G.729 offers an MOS of about 4.0. Figure 3-9 shows a high-level block diagram of the G.729 encoder.

G.729 Annex A . G.729 is a complex codec. In order to reduce the complexity in the algorithm, a number of simplifications were introduced in Annex A to G.729. These include simplified codebook search routines and a simplification to the postfilter at the decoder among other things. G.729A uses exactly the same transmitted frame structure as G.729 and therefore uses the same bandwidth. In other words, the encoder may be operating according to G.729, while the decoder may operate using G.729A or vice versa. Note that G.729A can result in slightly lower quality than G.729. G.729A provides a MOS of about 3.7.

G.729 Annex B Annex B to G.729 is a recommendation for *voice activity detection* (VAD), *discontinuous transmission* (DTX), and *comfort noise generation* (CNG). VAD is simply the decision as to whether voice or noise is present at the input. The decision is based on an analysis of several parameters of the input signal. Note that the determination is not done simply on the basis of one frame; rather, the determination is made on the basis of the current frame, plus the preceding two frames. This mechanism ensures that transmission occurs for at least two frames after a person has stopped speaking.

The next decision is whether to send nothing at all or to send a SID frame. The SID frame contains some information to enable the decoder to

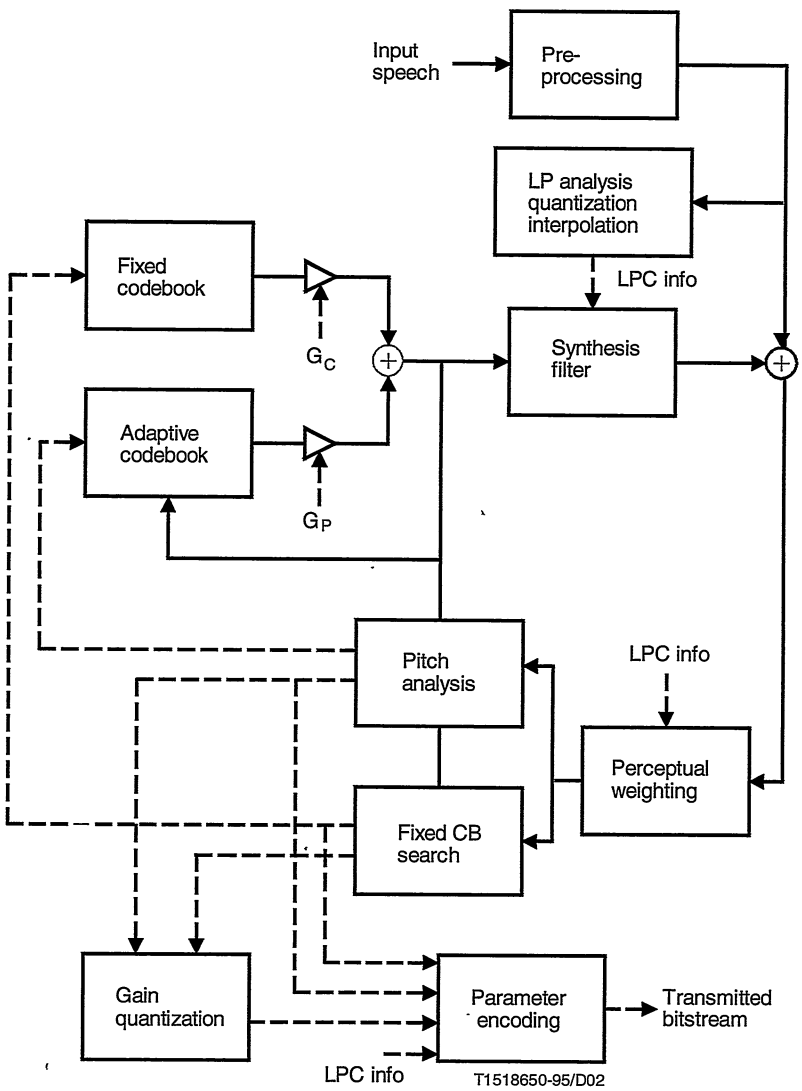


Figure 3-9
Encoding principle of the CS-ACELP model

generate comfort noise that simulates the background noise at the transmission end. The G.729B SID frame is a mere 15 bits long, significantly shorter than the 80-bit speech frame.

Assuming that the silence continues for some time, the encoder keeps watch on the background noise. If no significant change occurs, then nothing is sent and the decoder continues to generate the same comfort noise. If, however, the encoder notices a significant change in the background noise

energy, an updated SID frame is sent to update the decoder on the characteristics of the background noise. This avoids a comfort noise that is constant and that, if it persists for some time, might no longer be very comforting to the listener.

G.729 Annex D G.729 Annex D is intended as a lower-rate extension to the basic G.729 algorithm. Like the basic G.729 algorithm, Annex D operates on 10-millisecond speech samples. Rather than sending 80 bits per frame, however, the Annex D algorithm uses 64 bits per frame, resulting in a bit rate of 6.4 Kbps.

G.729 Annex D provides a MOS that is similar to that of G.723.1 operating at 6.3 Kbps. This value is lower than the MOS of the basic G.729 algorithm. The slight reduction in quality can be considered the cost of achieving a lower bandwidth.

G.729 Annex E G.729 Annex E offers a higher bit rate enhancement to the basic G.729 algorithm. The intention with this enhancement is to provide greater robustness in the presence of significant background noise (particularly music) at the input.

G.729 uses a tenth-order linear prediction filter, which means that the filter contains 10 coefficients. The G.729E coder uses 30 filter coefficients. Moreover, the codebook of G.729E is 44 bits, as opposed to 35 bits in G.729. The net effect of these changes is that G.729E transmits 118 bits for every 10 milliseconds of input signal, resulting in a bit rate of 11.8 Kbps.

Selecting Codecs

The codecs just discussed constitute a small selection of the coding techniques available. Many others add to the menu of choices when choosing a coding technology for use in a given network. Examples of other choices include the following:

- **CDMA QCELP, as defined in IS-733** This is a variable-rate coder and is currently used in IS-95-based *Code Division Multiple Access* (CDMA) wireless systems. *Qualcom code-excited linear predictor* (QCELP) can operate at several rates, but the two most common are the high rate of 13,300 *bits per second* (bps) and the low rate of 6,200 bps. The codec supports silence suppression, however, such that the net bit rate transmitted is less in each case. The packetization of the QCELP codec for use with RTP is defined in RFC 2658.

Motorola - Exhibit 1022 - Page 117

- ***Global System for Mobile Communications* (GSM) *Enhanced Fullrate* (EFR), as defined in GSM specification 06.60** This coder is an alternative to the original GSM *full-rate* (FR) coder. Although the GSM FR coder uses *Regular Pulse-Excited—Long Term Prediction* (RPE-LTP), the EFR coder uses an ACELP-based coding technique. The EFR coder operates at a rate of 12,200 bps, while the FR coder operates at a rate of 13,000 bps. Despite the lower bandwidth, the EFR coder provides better quality than the FR coder. Packetization of GSM EFR for use with RTP is documented in RFC 1890.
- ***Adaptive Multi-Rate* (AMR) codec, as defined in GSM specification 06.90** AMR is a multimode codec that can operate in any of 8 different modes, with bit rates from 4.75 to 12.2 Kbps. When operating at 12.2 Kbps, it is effectively the GSM EFR coder. When operating at 7.4 Kbps, it is effectively the IS-641 coder used in *Time Division Multiple Access* (TDMA) cellular systems. The coder can change to any of its modes as often as every 20 milliseconds in response to channel conditions. AMR supports discontinuous transmission, thereby minimizing the bandwidth consumed during silent periods, which reduces the net bandwidth consumption. The AMR codec is the coding choice for many third-generation wireless systems. The packetization rules for use of this codec with RTP are in the form of an Internet draft as of this writing.

Clearly, many options exist from which to choose. Furthermore, new coding schemes are being developed at a rapid pace. The task of selecting the best codec or set of codecs to use for a given network implementation is a matter of balancing quality with bandwidth consumption. Given the range of codecs available, however, the choice might not be that easy.

When looking at MOS values, one should remember that the MOS value presented for a particular codec is normally the value that is achieved under laboratory situations. The likelihood is significant that a given codec will not provide the same quality in a particular network implementation. For example, G.711, while offering the highest quality under ideal circumstances, does not incorporate any logic to deal with lost packets. In contrast, G.729 does have the capability to accommodate a lost frame by interpolating from previous frames. On the other hand, this causes subsequent speech frames at the decoder to have errors because of the fact that the decoder bases its output on the parameters of previous frames and it can take some time to recover from a lost or corrupt frame.

An important consideration is the delay that a codec will introduce. While waveform codecs do not introduce any algorithmic delay,

predictive coders introduce a delay at the coder and a somewhat smaller delay at the decoder.

Another consideration is the DSP power required, measured in *million instructions per second* (MIPS). Whereas G.728 or G.729 may need to run on a 40 MIPS processor, G.726 could run on a 10 MIPS processor. In general, the more MIPS, the greater the cost of the DSP. Fortunately, the price/performance of DSPs continues to improve such that this issue is becoming less of a factor.

At the end of the day, there is no hard and fast answer. The cost/quality tradeoff needs to be made for each individual circumstance.

Cascaded Codecs

In general, we should minimize the number of times that a given speech segment is coded and decoded. In the ideal case, the original analog signal should be coded only once and extracted only at the ultimate destination. It is certainly not a great idea to code a particular analog stream using one codec at, say, 16 Kbps, then convert that to G.711, and later encode again using G.729. In such cases, the voice quality might not even come close to the MOS of the lowest-quality codec in the chain. The simple reason is that each coder can only generate an approximation of the incoming signal that it receives. At each step along the chain, the approximation bears less and less resemblance to the original analog waveform.

In some cases, a VoIP implementation may be such that cascaded codecs are unavoidable. If that is the case, then a thorough trial of different configurations and codecs is well worth the effort to ensure that the ultimate quality does not degrade to an extent that the loss of quality would negate any benefits that VoIP might give.

Tones, Signals, and Dual-Tone Multifrequency (DTMF) Digits

The most sophisticated codecs available today achieve bandwidth efficiency without losing significant quality due to smart algorithms and powerful DSPs. Those smart algorithms, however, are based upon a detailed understanding of how voice is produced in the first place. In other words, the

codecs are optimized for human speech. Unfortunately, human speech is not always the only information that needs to be transmitted.

Within the telephone network, in addition to voice, all sorts of tones and beeps are transmitted—not the kind of sounds produced naturally as part of human conversation. These tones include fax tones, various tones such as busy tones, and the *dual-tone multifrequency* (DTMF) digits used in two-stage dialing, voice mail retrieval, and other applications. G.711 can generally handle these tones and beeps quite well, because G.711 samples the incoming signal without considering the origin of that signal. Since G.711 is the coder used in traditional telephony, such tones present no problem in circuit-switched networks. The same tones cannot, however, be faithfully reproduced by lower-bandwidth codecs such as G.723.1 or G.729. If such tones are used as input, the output from those codecs can be unintelligible.

Many VoIP systems use gateways to provide an interface between the IP network and the circuit-switched network. Incoming speech, either in analog form or coded according to G.711, is converted to packetized speech at a lower bandwidth using an efficient coder. It is then passed though the IP network to the destination. The destination may be a native VoIP device or another gateway, in which case the packetized speech needs to be converted back to G.711. In either case, if DTMF digits or other tones arrive at the ingress gateway, they need to be intercepted by the gateway and communicated to the far end in a different manner. They should not be encoded in the same way as speech.

Assuming that a gateway has the capability to detect tones of different types, then it can convey those tones to the destination in one of two ways. The first is to use an external signaling system that is separate from the media stream itself. Although various signaling protocols exist (as discussed in Chapter 4, "H.323," Chapter 5, "The Session Initiation Protocol (SIP)," and Chapter 6, "Media Gateway Control and the Softswitch Architecture"), they are not optimized for transmitting tones under every circumstance. For example, it is one thing to collect DTMF digits at the start of a call and convey them in a signaling system as a called party number. That task is relatively straightforward if the VoIP network is expecting to receive digits at the start of a call and it understands that such digits represent a dialed number. DTMF digits in the middle of a call, however, such as those that are produced when someone is trying to retrieve voice mail, are less easy to handle. That is because the VoIP network itself is not involved in the interpretation of the real meaning of such digits.

The other alternative is to encode the tones differently from the speech but send them along the same media path. We can approach that task in two ways. First, if a gateway can recognize a given tone, then it can send one

or more RTP packets that simply provide a name or other identifier for the tone, plus an indication of how long the tone is applied. This effectively means using an RTP packet to signal the occurrence of a tone rather than carrying the tone itself. Assuming that the far end understands the identifier, then it can reproduce the actual tone. The second approach is to use a dynamic RTP profile, where RTP packets carry information regarding the frequency, volume, and duration of the tone. That approach means that the ingress gateway does not need to recognize a particular tone; rather, it only needs to detect and measure the characteristics of a tone and then send a packet describing what was measured.

RFC 2198 describes an RTP payload format for redundant audio data, whereby speech quality can be maintained in the presence of packet loss by sending redundant codings for the same speech signal. Having two methods for communicating the same information regarding a tone means that we can use one of the methods as a backup for the other in accordance with RFC 2198. Thus, the entity (a gateway, for example) that detects a tone can send both types of RTP payload in a given RTP packet. The receiver can then select from the two representations of the tone.

RFC 2833 has been prepared to address the issue of carrying tones within the RTP stream. This specification addresses both of the two methods just described. To support the sending of RTP packets with identifiers for the individual tones, RFC 2833 specifies a large number of tones and events such as DTMF digits, busy tones, congestion tones, and ringing tones. It also provides a numeric identifier for each.

The payload format for these named events is shown in Figure 3-10. The event field contains a number indicating the event in question. The R bit is reserved and the E bit indicates the end of the tone. A sender may delay setting this bit until the tone has stopped. The volume field indicates the power level of the tone from 0 dBm0 to −63 dBm0, with the sign omitted. Therefore, a larger value in the volume field implies a lower volume. The volume field is relevant only for DTMF digits. The duration indicates, in timestamp units, how long the tone has lasted so far. Therefore, the tone event starts at the time indicated in the RTP timestamp, and the duration field indicates how long the tone has lasted. The size of this field can accommodate a duration up to 8 seconds, assuming a sampling rate of 8,000 Hz.

Note that a gateway that detects a tone should send an event packet as soon as the tone is recognized and every 50 milliseconds thereafter while the tone lasts. At least three event packets should be sent even if the tone is very brief, just to make sure that the far end has been informed of the event occurrence, even in the case of packet loss. When several packets are sent for the same tone, the RTP timestamp of each packet should be the

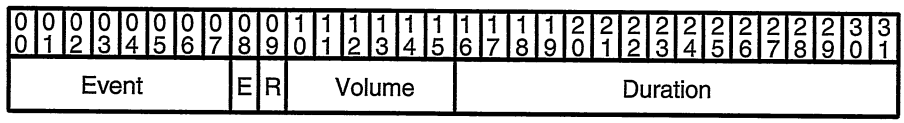


Figure 3-10
The RTP payload format for named events

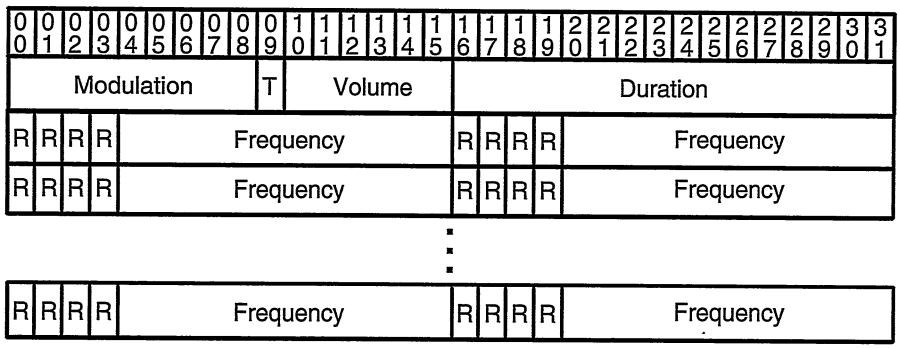


Figure 3-11
The RTP payload format for tones

same, indicating the instant that the tone was detected, with the duration field increased in each packet.

Figure 3-11 shows the payload format to be used for the second method of representing a tone, that is, when sending a description of the tone characteristics. The modulation field indicates the modulation in Hz, up to a maximum of 511 Hz. If no modulation exists, this field is 0. The T bit indicates whether the modulation frequency is to be divided by 3. Some tones are in use where the modulation frequency is not a whole number of Hz. For example, cases occur where modulation at 16 ⅔ Hz is used.

The volume measures the volume of the tone from 0 dBm0 to −63 dBm0. The duration is in RTP timestamp units. The frequency indicates the frequency, in Hz, of one component of the tone. This field support values up to 4,095 Hz, which exceeds the frequency range telephone systems. Note that a given tone can have multiple frequency components. For example, MF (as used in R1-MF signaling) stands for multifrequency. The R field is reserved for future use.

CHAPTER 4

H.323

The foregoing chapters address the issues related to *Internet Protocol* (IP) networks in general and the mechanisms for transporting digitally encoded voice across those networks. Chapter 2, "Transport Voice by Using IP," in particular, describes how voice is carried in *Real-Time Transport Protocol* (RTP) packets between session participants. What is not addressed, however, is the setup and teardown of those voice sessions. Thus far, we have assumed that session participants know of each other's existence and that media sessions are somehow created such that they can exchange voice using RTP packets. So how are those sessions created and ended? How does one party indicate to another a desire to set up a call, and how does the second party indicate a willingness to accept the call? The answer is signaling.

In traditional telephony networks, specific signaling protocols are invoked before and during a call in order to communicate a desire to set up a call, to monitor call progress, and to gracefully bring a call to an end. Perhaps the best example is the *ISDN User Part* (ISUP), a component of the *Signaling System 7* (SS7) signaling suite. In *Voice over IP* (VoIP) systems, signaling protocols are necessary for exactly the same reasons.

The very first VoIP systems used proprietary signaling protocols. The immediate drawback was that two users could communicate only if they both used systems from the same vendor. This lack of interoperability between systems from different vendors was a major inconvenience and impeded the early adoption of VoIP. In response to this problem, the *International Telecommunications Union Telecommunications Standardization Sector* (ITU-T) recommendation H.323 was developed, which serves as a standardized signaling protocol for VoIP. H.323 is the most widely deployed standard in VoIP networks today and is the focus of this chapter.

The first version of H.323 was released in 1996 and bore the title "Visual Telephone Systems and Equipment for Local Area Networks Which Provide a Non-Guaranteed Quality of Service" (quite a mouthful). Its scope was multimedia communications over *local area networks* (LANs). As such, many found the first version of H.323 to be lacking the functions needed for supporting VoIP in a broader environment.

Consequently, the ITU-T produced version 2 of H.323, which was released in 1998. This bears the friendlier title, "Packet-based Multimedia Communications Systems," and has seen greater acceptance and deployment. Since the publication of H.323 version 2, additional enhancements have been specified, which have led to additional revisions of the specification. The most recent version is H.323 version 4. Wherever we make reference to H.323 in this chapter, we are implying version 4 (unless otherwise stated).

Motorola - Exhibit 1022 - Page 125

The H.323 Architecture

H.323 is one of those ITU recommendations that specifies an overall architecture and methodology, and that incorporates several other recommendations. H.323 should be read in conjunction with several other recommendations and, equally, those other recommendations should be read in conjunction with H.323. Among the other most important recommendations are H.225.0 and H.245, though many others exist.

The scope of H.323 is illustrated in Figure 4-1. The architecture involves H.323 terminals, gateways, gatekeepers, and *multipoint controller units* (MCUs). The overall objective of H.323 is to enable the exchange of media streams between H.323 endpoints, where an H.323 endpoint is an H.323 terminal, a gateway, or an MCU.

An H.323 terminal is an endpoint that offers real-time communications with other H.323 endpoints. Typically, a terminal is an end-user communications device that supports at least one audio *coder/decoder* (codec) and may optionally support other audio codecs and/or video codecs.

A gateway is an H.323 endpoint that provides translation services between the H.323 network and another type of network, such as an

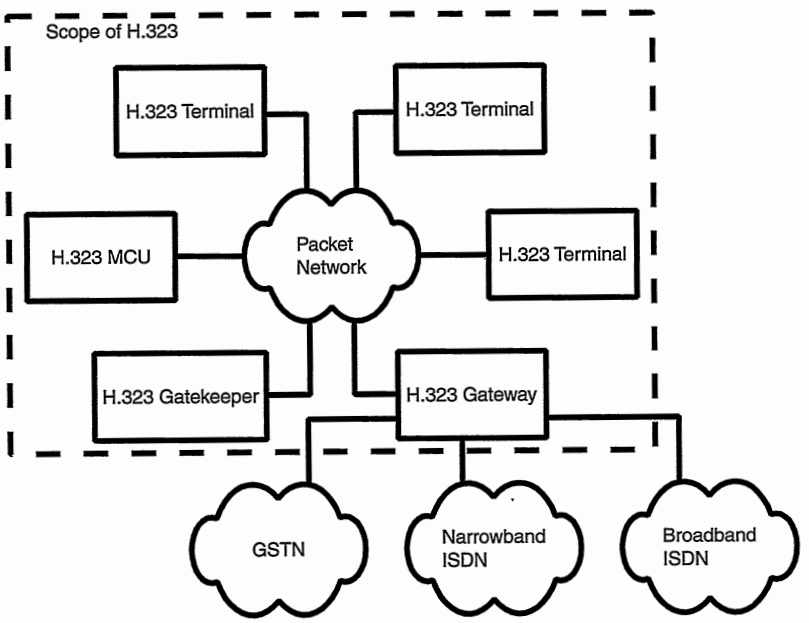


Figure 4-1
The scope of H.323 and interoperability of H.323 terminals

Integrated Services Digital Network (ISDN) or the regular phone network, which in ITU parlance is known as the *General Switched Telephone Network* (GSTN). One side of the gateway supports H.323 signaling and terminates packet media according to the requirements of H.323. The other side of the gateway interfaces with a circuit-switched network and supports the transmission characteristics and signaling protocols of that network. On the H.323 side, the gateway has the characteristics of an H.323 terminal. On the circuit-switched side, it has the characteristics of a node in the circuit-switched network. Translation between the signaling protocols and media formats of one side and those of the other side is performed internally within the gateway. This translation is transparent to other nodes in the circuit-switched network and in the H.323 network. Gateways can also serve as a conduit for communications between H.323 terminals that are not on the same network, where the communication between the terminals needs to pass via an external network such as the *Public Switched Telephone Network* (PSTN).

A gatekeeper is an optional entity within an H.323 network. When present, a gatekeeper controls a number of H.323 terminals, gateways, and *multipoint controllers* (MCs). By control, we mean that the gatekeeper authorizes network access from one or more endpoints and may choose to permit or deny any given call from an endpoint within its control. A gatekeeper may offer bandwidth control services, which can help to ensure high *quality of service* (QoS) if used in conjunction with bandwidth and/or resource management techniques. A gatekeeper also offers address translation services, enabling the use of aliases within the network.

The set of terminals, gateways, and MCs that a single gatekeeper controls is known as a *zone*. Figure 4-2 shows a representation of such a zone. A zone can span multiple networks or network segments, and it is not necessary that all entities within a zone be contiguous.

An MC is an H.323 endpoint that manages multipoint conferences between three or more terminals and/or gateways. For such conferences, the MC establishes the media that may be shared between entities by transmitting a capability set to the various participants. The MC can change the capability set in the event that other endpoints join the conference or if existing endpoints leave the conference. The MC itself does not, however, perform translation or mixing functions. Those functions are performed by a *multipoint processor* (MP), which the MC controls. An MC may reside within a separate MCU or may be incorporated within the same platform as a gateway, a gatekeeper, or an H.323 terminal.

For every MC, there is at least one MP that operates under the control of the MC. It is the MP that processes the actual media streams, creating a

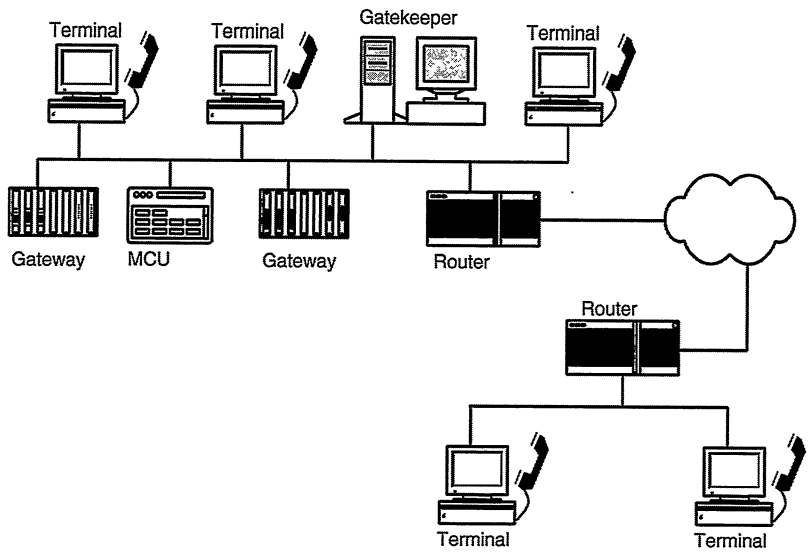


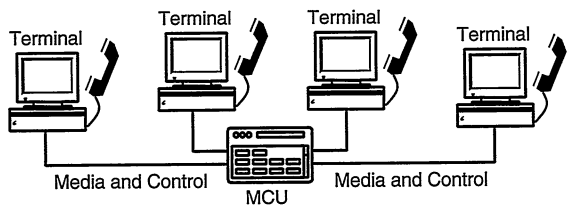
Figure 4-2
An example of an H.323 zone

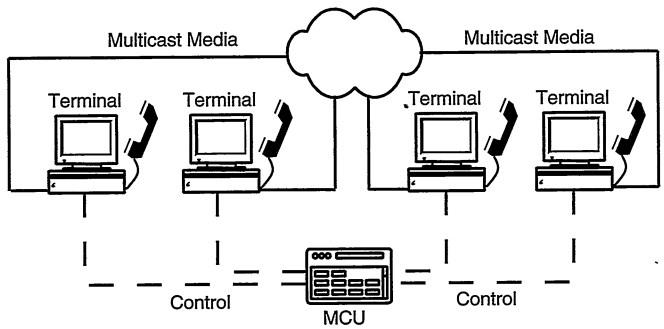
number of output media streams from a number of input streams. The MP performs such manipulation by switching, mixing, or a combination of the two. The control protocol between the MC and MP is not standardized.

MCs can support two main types of multipoint conferences: centralized and decentralized. These two arrangements are depicted in Figure 4-3. In a centralized configuration, every endpoint in the conference communicates with the MC in a sort of hub and spoke arrangement. In a decentralized configuration, each endpoint in the conference exchanges control signaling with the MC in a point-to-point manner, but they can share media with the other conference participants through multicast. The MC also enables a mixed conference (also shown in Figure 4-3), where some participants use multicast transmission while other participants use unicast.

Overview of H.323 Signaling

Figure 4-4 shows the H.323 protocol stack. Upon examination, we find a number of protocols already discussed, such as RTP, *RTP Control Protocol* (RTCP), *Transmission Control Protocol* (TCP), and *User Datagram Protocol* (UDP). It is clear from the figure that the exchange of media is performed using RTP over UDP and, of course, wherever we find RTP, we also find RTCP.





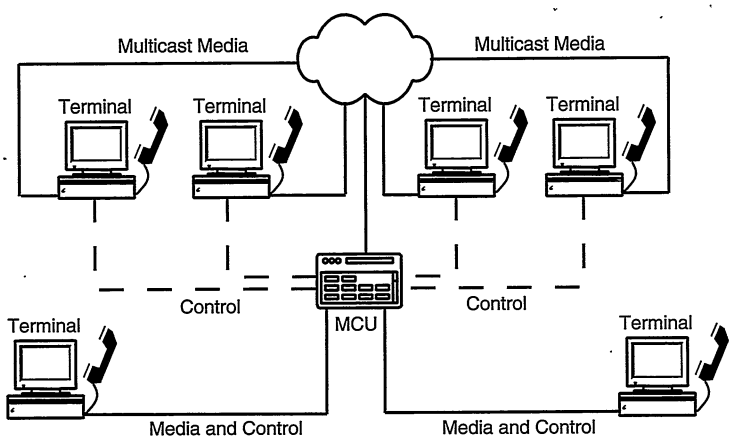


Figure 4-3
An example of H.323 conference configurations

In Figure 4-4, we also find two protocols that have not yet been discussed: namely H.225.0 and H.245. These two protocols define the actual messages that are exchanged between H.323 endpoints. They are generic protocols in that they could be used in any of a number of network architectures. When it comes to the H.323 network architecture, the manner in

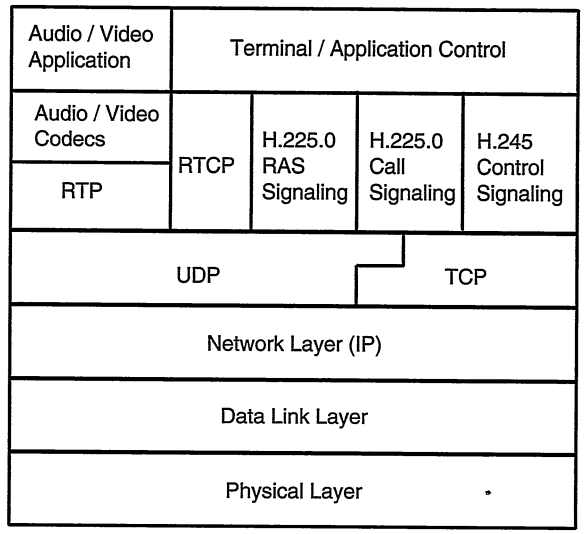


Figure 4-4
H.323 protocol stack

which the H.225.0 and H.245 protocols are applied is specified by recommendation H.323.

Overview of H.323 Protocols

As mentioned previously, the actual signaling messages exchanged between H.323 entities are specified in the ITU recommendations H.225.0 and H.245. H.225.0 is a two-part protocol. One part is effectively a variant of ITU-T recommendation Q.931, the ISDN layer 3 specification, and should be quite familiar to those with knowledge of ISDN. This signaling is used for the establishment and teardown of connections between H.323 endpoints. This type of signaling is known as call signaling or Q.931 signaling. The other part of H.225.0 is known as *Registration, Admission, and Status* (RAS) signaling. This signaling is used between endpoints and gatekeepers, and it enables a gatekeeper to manage the endpoints within its zone. For example, an endpoint uses RAS signaling to register with a gatekeeper, and a gatekeeper uses RAS signaling to allow or deny an endpoint access to network resources.

Note that RAS signaling is always carried over UDP, while call signaling may be carried over either UDP or TCP. H.323 version 2 mandates that call signaling use TCP. The establishment of a TCP connection takes a little time, however, which can lead to a delay in call setup. To speed things up, Annex E to H.323 version 4 specifies a mechanism whereby either TCP or UDP can be used for call signaling. In fact, both TCP and UDP can be used

in parallel. The sending entity sends the first message using UDP and simultaneously establishes a TCP connection. If, by the time the TCP connection is established, no response has been received to the UDP-based message, then the TCP connection is used. If, however, a response has been received to the UDP-based message, then the original sender can assume that the far end supports H.323 version 4 and can continue to use UDP for call signaling. The sender then closes the TCP connection.

H.245 is a control protocol used between two or more endpoints. The main purpose of H.245 is to manage the media streams between H.323 session participants. To that end, H.245 includes functions such as ensuring that the media to be sent by one entity is limited to the set of media that can be received and understood by another. H.245 operates by the establishment of one or more logical channels between endpoints. These logical channels carry the media streams between the participants and have a number of properties such as media type, bit rate, and so on.

All three signaling protocols—RAS, call signaling, and H.245—can be used in the establishment, maintenance, and teardown of a call. The various messages may be interleaved. For example, consider an endpoint that wants to establish a call to another endpoint. First, the endpoint might use RAS signaling to obtain permission from a gatekeeper. The endpoint might then use call signaling to establish communication with the other endpoint to set up the call. Finally, the endpoint might use H.245 control signaling to negotiate media parameters with the other endpoint and set up the media transfer.

H.323 messages are sent on various types of channels, depending on the message (and, in some cases, the context). For example, RAS messages are sent over the RAS channel, call-signaling messages are sent over the call-signaling channel, H.245 control messages are sent over the H.245 control channel, and the actual media streams are sent on one or more logical channels. Although this approach might appear to involve a lot of different channels, it is worth noting that these channels are not necessarily related to particular physical interfaces or hardware. Instead, a channel, when used in an IP environment, is simply a reference to a socket address (that is, an IP address and port number). For example, if a given endpoint uses a particular IP address and port number to receive RAS messages, then any messages that arrive at that IP address and port are said to arrive on the endpoint's RAS channel.

Protocol Syntax As we move forward in this chapter, we will describe the main points of each of the protocols. We will also provide a number of

illustrations to explain how the different protocols fit together. We will not, however, describe the detailed syntax of the protocols—and for good reason.

As is the case for many ITU-T recommendations, H.323 signaling is specified using *Abstract Syntax Notation 1* (ASN.1). Although this syntax is eminently suitable for interpretation by software tools and is in fact a formalized language, it is certainly not overly friendly to the human reader. Explaining the syntax of the protocols in detail would require a great deal of ASN.1 syntax in this book and a tutorial on the interpretation of the syntax. Such material would most likely clutter the text and detract from the main objective, which is to explain the functionality offered and how the protocols support that functionality. Consequently, little ASN.1 syntax is included here; instead, we provide a textual description.

H.323 Addressing

Every entity in the H.323 network has a network address, which uniquely identifies that particular entity. In an IP environment, the network address is an IP address. If *Domain Name Service* (DNS) is available, this IP address may be specified in the form of a *Uniform Resource Locator* (URL), according to RFC 1738 and RFC 2396. For example, the URL ras://GK1 @somedomain would be a valid URL for a gatekeeper, since RAS is the protocol supported by a gatekeeper. Any URL can have a port number appended. It the case of RAS, a default port number (1719) exists and should be used in the case that a port number is not specified.

For convenience of identification, entities such as terminals, gateways, and MCUs should have a domain name that is common with their controlling gatekeeper. Note, however, that although applying URLs to endpoints and gatekeepers is a convenient means of identification, the actual IP address is passed in messages between H.323 entities.

For each network address, an H.323 entity can have one or more *Transport Service Access Point* (TSAP) identifiers. In generic terms, a TSAP identifier is an identifier for a particular logical channel at a given entity. In IP terms, it is the same as a socket address.

In general, the port numbers to be used for signaling transactions or media exchanges are assigned dynamically. Important exceptions are the gatekeeper UDP discovery port with value 1718, the gatekeeper UDP registration and status port with value 1719, and the call-signaling TCP or UDP port with value 1720. These port numbers are registered with the *Internet Assigned Numbers Authority* (IANA). The first port number is used

when an endpoint wants to determine the gatekeeper with which it should be registered. The second port number is used for RAS signaling to a gatekeeper. The third port number is used when sending call-signaling (Q.931) messages.

In addition to network addresses and TSAP identifiers, terminals and gateways can have one or more aliases. Given that messages must be addressed with real IP addresses, the capability for nodes to have aliases implies that a means to translate between an alias and an IP address must be established. This translation is one of the functions of a gatekeeper and is supported by RAS signaling.

H.323 is flexible when it comes to the allocation of alias addresses. They can take any number of forms, and a given endpoint may have multiple aliases. The only real restriction is that a given alias must be unique within a zone, that is, within the set of nodes controlled by a given gatekeeper (which makes sense). One useful application of aliases is the use of E.164 numbers as aliases. For example, a given gateway that is connected to a regular *private branch exchange* (PBX) over a *primary rate interface* (PRI) could be allocated a set of E.164 numbers as aliases. These numbers could correspond to the telephone numbers that are reachable at the PBX. If another H.323 endpoint wants to call a number on the PBX, the use of the alias makes the task easier. Only the gatekeeper and the gateway need to know the relationship between the alias and the gateway address. The calling endpoint and other entities can avail themselves of the gatekeeper to take care of the necessary translation.

Codecs

In Figure 4-4 we see reference to audio and video codecs. Video support is optional, but when video is supported, an H.323 endpoint must, at a minimum, support video according to H.261 *Quarter Common Intermediate Format* (QCIF).

The support of audio is mandatory and H.323 mandates that the G.711 codec be supported (both A-law and μ-law). Given its relatively high bandwidth requirement, G.711 is certainly not considered the first choice for audio codecs. G.711 is specified as mandatory, however, so that all H.323 endpoints support at least one audio codec. In reality, most endpoints will support a range of more efficient codecs. After all, if a system implements only G.711, then the potential bandwidth efficiencies of VoIP are negated. After all, we would like to carry voice as efficiently as possible, provided that quality objectives can be met.

Motorola - Exhibit 1022 - Page 133

RAS Signaling

RAS signaling is used between a gatekeeper and the endpoints that it controls. RAS is the signaling protocol through which a gatekeeper controls the endpoints within its zone. Note that a gatekeeper is an optional entity in an H.323 network. Consequently, RAS signaling is also optional. If an endpoint wants to use the services of a gatekeeper, then it must implement RAS. If the endpoint chooses not to use the services of a gatekeeper, then the functions that a gatekeeper normally provides must be provided within the endpoint itself.

RAS signaling is defined in H.225.0 and supports the following functions:

- *Gatekeeper discovery* enables an endpoint to determine which gatekeeper is available to control it.
- *Registration* enables an endpoint to register with a particular gatekeeper and hence to join the zone of that gatekeeper.
- *Unregistration* enables an endpoint to leave the control of a gatekeeper or enables a gatekeeper to cancel the existing registration of an endpoint, thereby forcibly removing the endpoint from the zone.
- *Admission* is used by an endpoint to request access to the network for the purpose of participating in a session. A request for admission specifies the bandwidth to be used by the endpoint, and the gatekeeper may choose to accept or deny the request based on the bandwidth requested.
- *Bandwidth change* is used by an endpoint to request the gatekeeper to allocate extra bandwidth to the endpoint, or it is used by a gatekeeper to instruct an endpoint to reduce the amount of bandwidth consumed.
- *Endpoint location* is a function where the gatekeeper translates an alias to a network address. An endpoint will invoke this function when it wants to communicate with a particular endpoint for which it only has an alias identifier. The gatekeeper will respond with a network address to be used to contact the endpoint in question.
- *Disengage* is used by an endpoint to inform a gatekeeper that it is disconnecting from a particular call. Disengage can also be used from gatekeeper to endpoint to force the endpoint to disconnect from a call.
- *Status* is used between the gatekeeper and endpoint to inform the gatekeeper about the health of an endpoint or about certain call-related data, such as current bandwidth usage.

- *Resource availability* is used between an endpoint and a gatekeeper to inform the gatekeeper of an endpoint's currently available capacity, such as the amount of bandwidth it has available. This procedure can also be used by an endpoint to inform a gatekeeper that the endpoint has run out of (or is about to run out of) capacity.
- *Non-standard* is a mechanism by which proprietary information may be passed between an endpoint and a gatekeeper. Of course, the message contents and functions to be invoked are not defined in H.225.0.

Table 4-1 provides a list of the various RAS messages that are used to support these functions and gives a brief description of the purpose of each message. Subsequent sections of this chapter provide more detail regarding the most common messages and functions.

Table 4-1

RAS messages that support RAS signaling functions

Message	Function
GatekeeperRequest (GRQ)	Used by an endpoint when trying to discover its gatekeeper.
GatekeeperConfirm (GCF)	Used by a gatekeeper to indicate that it is available to be the gatekeeper for a given endpoint.
GatekeeperReject (GRJ)	Used by a gatekeeper to indicate that it will not be the gatekeeper for a given endpoint.
RegistrationRequest (RRQ)	Used by an endpoint to register with a gatekeeper.
RegistrationConfirm (RCF)	A positive response from gatekeeper to endpoint, indicating a successful registration.
RegistrationReject (RRJ)	A negative response to a RegistrationRequest.
UnregistrationRequest(URQ)	Used either by a gatekeeper or endpoint to cancel an existing registration.
UnregistrationConfirm (UCF)	Used by a gatekeeper or endpoint to confirm the cancellation of a registration.
UnregistrationReject (URJ)	A negative response to an UnregistrationRequest.
AdmissionRequest (ARQ)	Sent from an endpoint to a gatekeeper to request permission to participate in a call.
AdmissionConfirm (ACF)	Used by a gatekeeper to grant permission to an endpoint to participate in a call.
AdmissionReject (ARJ)	Used by a gatekeeper to deny an endpoint permission to participate in a call.
BandwidthRequest (BRQ)	Sent from an endpoint or a gatekeeper to request a change in allocated bandwidth.

Motorola - Exhibit 1022 - Page 135

Table 4-1 cont.

RAS messages that support RAS signaling functions

Message	Function
BandwidthConfirm (BCF)	Positive response to a BandwidthRequest.
BandwidthReject (BRJ)	Used by a gatekeeper or endpoint to deny a change in bandwidth. BRJ should be used by an endpoint only if the new bandwidth cannot be supported.
InfoRequest (IRQ)	Sent by a gatekeeper to an endpoint in order to request status information.
InfoRequestResponse (IRR)	Sent from an endpoint to a gatekeeper to provide status information. Can be sent on demand or autonomously.
InfoRequestAck (IACK)	Sent by a gatekeeper as confirmation of the receipt of an IRR.
InfoRequestNak (INAK)	Sent by a gatekeeper upon receiving an IRR in an error situation, such as from an unregistered endpoint.
DisengageRequest (DRQ)	Sent by an endpoint or gatekeeper to request disconnection of a call at an endpoint.
DisengageConfirm (DCF)	Positive response to a DRQ.
DisengageReject (DRJ)	Negative response to a DRQ, such as one from an unregistered endpoint.
LocationRequest (LRQ)	Sent to a gatekeeper to request translation from an alias to a network address.
LocationConfirm (LCF)	Response to an LRQ, including the required address.
LocationReject (LRJ)	Response to an LRQ when translation was not successful.
Non-Standard Message (NSM)	Vendor-specific.
UnknownMessage Response (XRS)	Sent in response to a message that is not recognized.
RequestInProgress (RIP)	Sent by an endpoint or gatekeeper as an interim response when a request is taking a long time to process.
ResourceAvailableIndicate (RAI)	Sent by a gateway to a gatekeeper to inform the gatekeeper of the gateway's current capacity.
ResourceAvailableConfirm (RAC)	Sent by a gatekeeper as acknowledgment to an RAI.
ServiceControlIndication (SCI)	Sent by a gatekeeper or endpoint in order to initiate a separate service control session related to advanced features and services. This message was introduced in H.323 version 4.
ServiceControlResponse (SCR)	Sent to acknowledge the receipt of an SCI message, but does not necessarily mean that the service client will initiate the session as given in SCI.

Gatekeeper Discovery

In a network that has one or more gatekeepers, an endpoint should register with one of the gatekeepers. In order to perform this registration, the endpoint must first find a suitably accommodating gatekeeper—one that is willing to take control of the endpoint. Of course, the endpoint may be configured in advance with the address of the gatekeeper that should be used. In that case, no discovery process takes place per se. Rather, the endpoint simply registers with the gatekeeper in question. Although such an approach might lead to a quicker registration, it lacks flexibility. In a given network, several gatekeepers may be in a load-sharing mode or a backup gatekeeper might need to assume the role of a failed gatekeeper. A static endpoint-gatekeeper relationship is not best suited for such scenarios. Therefore, an automatic gatekeeper discovery process is available.

The gatekeeper discovery process is available for an endpoint to determine its gatekeeper in the event that it does not already have that information. In order to discover which gatekeeper is to exercise control over a given endpoint, the endpoint sends a *gateway-request message* (GRQ). This message can be sent to a number of known addresses, or it can be sent to the gatekeeper discovery multicast address and port 224.0.1.41:1718.

The GRQ message contains a number of parameters. One of these is a gatekeeper identifier. If this parameter is empty, the GRQ indicates to those gatekeepers that receive the message that the endpoint is soliciting gatekeepers and is willing to accept any available gatekeeper. In other words, the endpoint is asking, "Will someone be my gatekeeper?" In the case that the parameter contains an identifier, the endpoint is asking a specific gatekeeper to be the endpoint's gatekeeper.

One or more of the gatekeepers may respond with a *gatekeeper-confirmation* (GCF) message, indicating that the gatekeeper is willing to control the endpoint. Alternatively, a gatekeeper may respond with a *gatekeeper-reject* (GRJ) message, indicating that the gatekeeper is not willing to assume control of that endpoint. In the case that a gatekeeper sends a GRJ message, it will include a reason for the rejection, such as lack of resources at the gatekeeper.

The process is illustrated in Figure 4-5. In this scenario, the terminal wanting to determine its gatekeeper sends just one GRQ message, but sends it to the gatekeeper discovery multicast address. Therefore, more than one gatekeeper can receive the message. In Figure 4-5, gatekeeper 1 sends a GRJ message, whereas gatekeeper 2 sends a GCF message. The terminal will now proceed to register with gatekeeper 2.

Motorola - Exhibit 1022 - Page 137

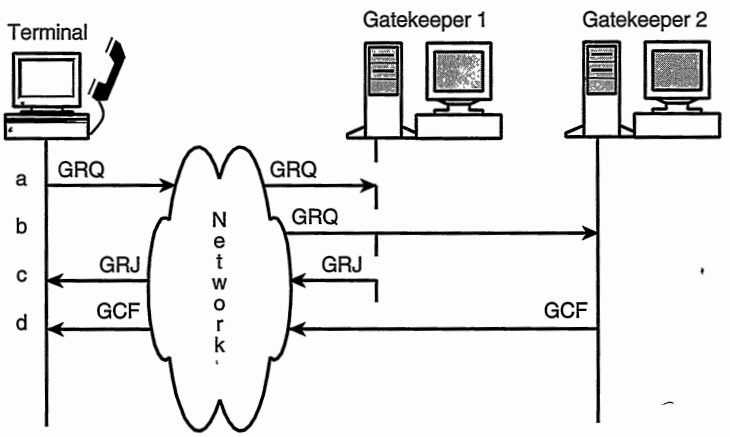


Figure 4-5
Gatekeeper discovery

Optionally, a given gatekeeper may send a GCF message indicating one or more other gatekeepers to try. This indication is included within the GCF message by the presence of the optional parameter AlternateGatekeeper. In such an event, the GCF message does *not* mean that the gatekeeper issuing the GCF message is willing to take control of the endpoint. Rather, the GCF message is somewhere between a positive response and a complete rejection and is equivalent to saying something like "I can't help you, but try the gatekeeper next door." This mechanism can be useful for load sharing or redundancy schemes.

Only one gatekeeper can control a given endpoint at a given time. If an endpoint receives multiple GCF messages from gatekeepers willing to accept control, then the choice of gatekeeper is left to the endpoint. If, on the other extreme, an endpoint receives no positive response to the GRQ message within a timeout period, then the endpoint can retry the GRQ, but should not issue a GRQ within 5 seconds of the previous request.

Endpoint Registration and Registration Cancellation

Gatekeeper discovery is simply a means for an endpoint to determine which gatekeeper is available to assume control of the endpoint. The fact that a given gatekeeper has responded to a GRQ with a GCF does not mean that the endpoint is now under control of that gatekeeper. The process by which

an endpoint becomes controlled by a gatekeeper and joins the zone of that gatekeeper is known as *registration*.

Registration begins with the issuance of a *RegistrationRequest* (RRQ) message from an endpoint to a gatekeeper. The message is directed to a gatekeeper at an address configured within the endpoint or determined by the endpoint through the gatekeeper discovery procedure. The port number to be used for the message is 1719—the well-known RAS signaling port of a gatekeeper.

The endpoint includes in the RRQ an address to be used by the endpoint for RAS messages and an address to be used for call-signaling messages. The RRQ can include an alias, a name by which the endpoint wants to be called. The RRQ can also include one or more alternative addresses available for use as backup or redundant addresses. In such cases, these addresses can each have their own aliases.

The gatekeeper can choose to reject a registration request for any number of reasons and does so by responding with a *RegistrationReject* (RRJ) message. One reason, for example, would be if the endpoint wants to use an alias already in use within the zone. If all is well, however, and the gatekeeper is willing to accept the registration, then it responds with a *RegistrationConfirmation* (RCF) message. If the original RRQ does not specify an alias for the endpoint, then the gatekeeper can assign one or more aliases, in which case the assigned aliases are returned as a parameter in the RCF message.

Registrations can have a limited lifetime. In an RRQ, an endpoint can include a timeToLive parameter, indicating the requested duration for the registration in seconds to a maximum value of 4294967295 (hexadecimal FFFFFFFF), which translates to about 136 years. In the RCF, the gatekeeper may choose a lower timeToLive value than that received in the RRQ. If, after some time, the registration period is about to expire, then the endpoint can send another RRQ that includes the optional parameter keepAlive, which amounts to a request to reset the timeToLive timer and extend the registration. An RRQ message that contains the keepAlive parameter is known as a lightweight registration request, because it is designed only to extend an existing registration.

Once registered with a gatekeeper, the endpoint can choose to cancel its registration at some later time. The endpoint does so by sending an *UnregistrationRequest* (URQ) message. Normally, a gatekeeper will respond with a positive confirmation by sending an *UnregistrationConfirmation* (UCF) message. It might, however, send a rejection if the endpoint was not registered with that Gatekeeper in the first place or if the endpoint is attempting to cancel a registration while still involved in a call. The rejection takes the form of an *UnregistrationReject* (URJ) message.

Motorola - Exhibit 1022 - Page 139

The gatekeeper can also choose to cancel the registration of a particular endpoint, in which case the gatekeeper will send a URQ message to the endpoint. One reason for this would be when the registration timeToLive has expired. Upon receipt of a URQ from a gatekeeper, the endpoint should respond with a UCF.

Figure 4-6 depicts an example of registration and registration cancellation. Note that this is just an example and that a URQ message may be issued by an endpoint as well as by a gatekeeper.

Endpoint Location

Endpoint location is a service that enables an endpoint or gatekeeper to request a real address when only an alias is available. In other words, endpoint location is a translation service.

An endpoint that wants to obtain contact information for a given alias can send a *LocationRequest* (LRQ) message to a gatekeeper. The message may be sent to a particular gatekeeper or can be multicast to the gatekeeper discovery multicast address 224.0.1.41. The LRQ contains the aliases for which address information is required. In the multicast case, the procedure is equivalent to asking "Does anyone know who has alias XXX?"

Not only can an endpoint send the LRQ, but a gatekeeper can send an LRQ to another gatekeeper to determine the address of an endpoint. This would happen when the call signaling from an endpoint is to pass through the calling endpoint's gatekeeper.

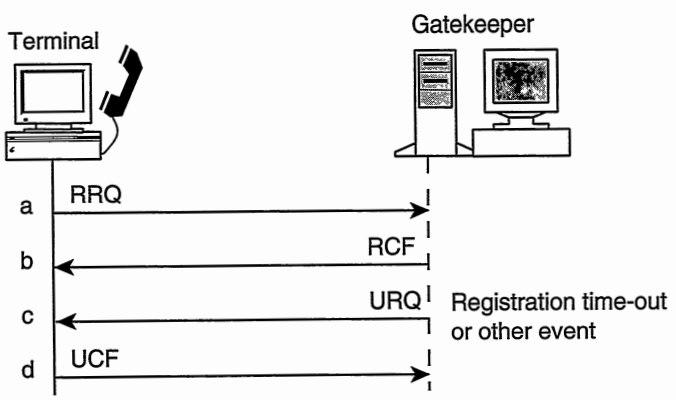


Figure 4-6
An example of registration and registration cancellation

A *LocationConfirm* (LCF) message indicates a positive response to the LRQ. This message should be sent from the gatekeeper at which the endpoint is registered. Among some other optional parameters, the LCF contains a call-signaling address and an RAS signaling address for the located endpoint.

If a gatekeeper receives an LRQ message and the endpoint is not registered at that gatekeeper, then the gatekeeper should respond with a *LocationReject* (LRJ) if the original LRQ had been received by the gatekeeper via unicast on its RAS channel. If, however, the gatekeeper receives an LRQ by multicast on the gatekeeper discovery multicast address, then the gatekeeper should not send an LRJ.

Admission

Admission is the process by which an endpoint requests permission from a gatekeeper to participate in a call. The endpoint does so by sending an *AdmissionRequest* (ARQ) message to the gatekeeper. The endpoint indicates the type of call in question (two-party or multiparty), the endpoint's own identifier, a call identifier (a unique string), a call reference value (an integer value also used in Q.931 messages for the same call), and information regarding the other party or parties to participate in the call. The information regarding other parties to the call includes one or more aliases and/or a signaling address.

One of the most important mandatory parameters in the ARQ is the bandwidth parameter. This parameter specifies the amount of bandwidth required in units of 100 bits/second. Note that the endpoint should request the total media stream bandwidth needed, excluding overhead. Thus, if a two-party call takes place, with each party sending voice at 64 Kbps, then the bandwidth required is 128 Kbps and the value carried in the bandwidth parameter is 1,280. The purpose of the bandwidth parameter is to enable the gatekeeper to reserve resources for the call. Through the use of the optional parameter transportQOS, however, the endpoint may indicate that it is willing to undertake resource reservation rather than having the gatekeeper do it.

The gatekeeper indicates successful admission by responding to the endpoint with an *AdmissionConfirm* (ACF) message. This includes many of the same parameters that are included in the ARQ. The difference is that when a given parameter is used in the ARQ, it is simply a request from the endpoint, whereas a given parameter value in the ACF is a firm order from the

gatekeeper. For example, the ACF includes the bandwidth parameter, which may be a lower value than that requested in the ARQ, in which case the endpoint must stay within the bandwidth limitations that the gatekeeper imposes. Equally, although the endpoint might have indicated that it will perform its own resource reservation, the gatekeeper might decide that it will take that responsibility.

Another parameter of interest in both the ARQ and the ACF is the callModel parameter, which is optional in the ARQ and mandatory in the ACF. In the ARQ, the parameter indicates whether the endpoint wants to send call signaling directly to the other party, or whether it prefers that call signaling be passed via the gatekeeper. In the ACF, the parameter represents the gatekeeper's decision as to whether call signaling is to pass via the gatekeeper. The first scenario is known as direct call signaling. In that case, the calling endpoint sends an ARQ to its gatekeeper. After the ACF is returned from the gatekeeper, the endpoint sends call signaling directly to the remote endpoint. In the case of gatekeeper-routed call signaling, the calling endpoint's gatekeeper returns a callModel parameter in the ACF, indicating that call signaling should be sent via the gatekeeper. The endpoint then sends the call signaling to the gatekeeper and the gatekeeper forwards the call signaling to the remote endpoint.

Figure 4-7 shows an example of direct call signaling, and Figure 4-8 shows an example of gatekeeper-routed call signaling. Both diagrams assume that the two endpoints are connected to the same gatekeeper, which

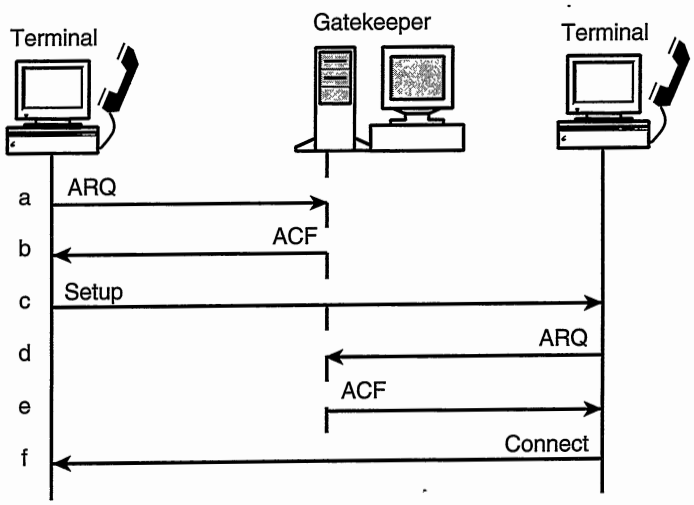


Figure 4-7
Direct call signaling

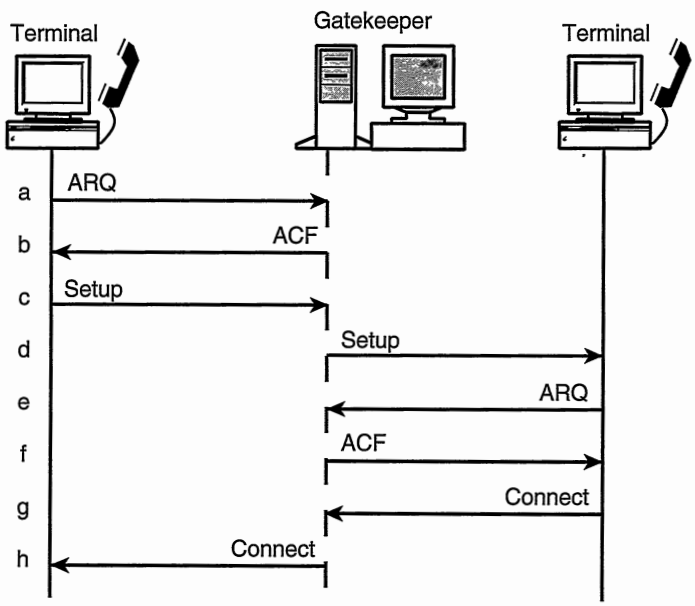


Figure 4-8
Gatekeeper-routed call signaling

might or might not be the case. In the event that the endpoints are connected to different gatekeepers, then each gatekeeper determines independently whether it would like to be in the path of the call signaling. Therefore, in the case of two gatekeepers, the signaling may pass via none, one, or both gatekeepers.

Both Figure 4-7 and Figure 4-8 show Setup and Connect messages. These are part of call signaling (Q.931), as described later in this chapter. The Setup message is the first message in call establishment between endpoints, and the Connect message indicates that a call is accepted by the called endpoint. It is sent when the call is answered.

Of course, a gatekeeper may decide to deny a particular admission request. In such an event, it responds to the ARQ with an *AdmissionReject* (ARJ) message, which includes a reason for the denial. Possible reasons include a lack of available bandwidth, an incapability to translate a destination alias to a real address, or an endpoint not being registered.

Pregranted Admission In order to minimize call setup delay, a gatekeeper can provide an endpoint with admission in advance. When the endpoint registers with a gatekeeper, the RCF can include the parameter *preGrantedARQ*. This parameter can provide permission for the endpoint to make or receive a call without asking for explicit permission through the

ARQ/ACF mechanism. The parameter also indicates whether the endpoint should use direct call signaling or gatekeeper-routed call signaling.

Bandwidth Change

At any time during a call, an endpoint can request either an increase or a decrease in allocated bandwidth. An endpoint itself can change the bit rate of any given channel without approval from the gatekeeper, provided that the total bandwidth does not exceed the limit specified by the gatekeeper. Consequently, an endpoint is not required to request a reduction in bandwidth, though it should do so as a means of informing the gatekeeper of reduced bandwidth requirements, thereby allowing the gatekeeper to release certain resources for potential use in other calls. If a change is to occur in a call, however, and that change will lead to an aggregate bandwidth in excess of the limit specified by the gatekeeper, then the endpoint must first ask the gatekeeper for permission. The endpoint may increase the bandwidth only after confirmation from the gatekeeper.

A change in bandwidth is requested by the use of a *BandwidthRequest* (BRQ) message. The message indicates the new bandwidth requested and the call to which the bandwidth applies. The gatekeeper can either approve the new bandwidth with a *BandwidthConfirm* (BCF) message or deny the request with a *BandwithReject* (BRJ) message. If the request is denied, then the endpoint must live with whatever bandwidth has already been allocated, perhaps through the use of flow-control mechanisms.

Not only can an endpoint request a change in bandwidth, but the gatekeeper can also request that an endpoint change its bandwidth limit. For example, the gatekeeper might determine that network resource utilization is approaching capacity and it might order an endpoint to reduce its bandwidth usage. If the gatekeeper requests a reduction in bandwidth, then the endpoint must comply with the request and must respond with a BCF message.

Note that changes in bandwidth are closely tied to H.245 signaling. Recall that H.245 specifies the signaling that occurs between endpoints that are involved in a call and relates to the actual media streams that the endpoints share. Also recall that H.245 defines the use of logical channels that carry the media between the endpoints. If a change in bandwidth is to occur, then one or more logical channels will experience a direct impact. In fact, a reduction in bandwidth will require an existing logical channel to be closed and reopened with different parameter settings related to the new bandwidth limitation.

Figure 4-9 provides an example of the interaction between H.225.0 RAS signaling and H.245 control signaling for changes of bandwidth. In this example, an endpoint wants to increase the bandwidth used on a call. The endpoint first uses the BRQ message to request permission from its gatekeeper. Assuming that the request is acceptable, the gatekeeper responds with a BCF message. The endpoint closes the existing logical channel to the remote endpoint and opens a new logical channel, which has a new bit rate. The remote endpoint sees that a higher bit rate is required, but before confirming the new bit rate, it must ask its gatekeeper for permission. Therefore, the remote endpoint also sends a BRQ and receives a BCF. Only upon receipt of the BCF from its gatekeeper does the remote endpoint confirm the opening of the new logical channel. Note that Figure 4-9 shows that the two endpoints are controlled by the same gatekeeper. They could just as easily be controlled by different gatekeepers. H.245 signaling is described in greater detail later in this chapter.

Status

It is desirable for a gatekeeper to be informed of the status of an endpoint, simply to know whether the endpoint is still functioning and to obtain information regarding any call that the endpoint may currently have active.

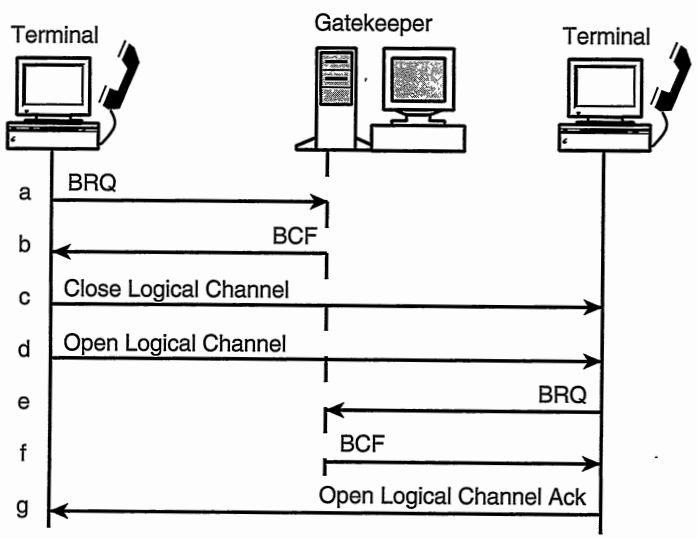


Figure 4-9 *Interaction between RAS bandwidth management and H.245 signaling*

Information to be provided by an endpoint to a gatekeeper is carried in an *InformationRequestResponse* (IRR) message. This message carries information about the endpoint itself, plus information about one or more of the calls that the endpoint might currently have active. The per-call information includes such items as the call identifier, the call reference value, the call type (two-party or multiparty), the bandwidth in use, and RTP session information (the *canonical name* [CNAME], RTP address, RTCP address, and so on).

The gatekeeper can stimulate an endpoint to send an IRR in two ways. First, the gatekeeper can poll the endpoint by sending an *InformationRequest* (IRQ) message to the endpoint. The other way is for the gatekeeper to indicate within the ACF message that the endpoint should periodically send the information. This is done by including the optional parameter, irrFrequency, in an ACF message. This parameter indicates the frequency (in seconds) at which an endpoint in a call should send an IRR message to the gatekeeper. If the gatekeeper has pregranted admission in an RCF message, then the preGrantedACF information element will contain the irrFrequency parameter.

An IRR might or might not receive an acknowledgment. The gatekeeper and endpoint jointly determine whether an acknowledgement is to be sent. First, the gatekeeper indicates whether or not it is willing to send acknowledgments to IRR messages and does so by setting the BOOLEAN willRespondToIRR parameter in the ACF or RCF message. If that parameter has been set, then an endpoint can indicate that a particular IRR should receive a response, which it does by setting the BOOLEAN needResponse parameter in the IRR.

The response to an IRR can be positive or negative. A positive response takes the form of an *InfoRequestAck* (IACK) message and a negative response takes the form of an *InfoRequestNak* (INAK) message. The negative response should be returned only in cases when a gatekeeper has received an IRR message in error, such as from an unregistered endpoint.

Disengage

At the end of a call, the various parties involved stop transmitting media to each other and terminate the session. At this stage, each endpoint should send a *DisengageRequest* (DRQ) to its gatekeeper. The message contains, among other parameters, the call identifier, the call reference value, and a disengage reason. In the case that this is a normal call termination, the value of the reason will be normalDrop.

A gatekeeper that receives the DRQ request normally responds with a *DisengageConfirm* (DCF) message. The gatekeeper might respond with a *DisengageReject* (DRJ) message under some unusual circumstances. Such scenarios could include an endpoint requesting to disengage when the endpoint is not registered with that gatekeeper or if the endpoint wants to terminate a call that is handled by a different endpoint.

Situations might exist in which a gatekeeper, rather than an endpoint, decides to terminate a call. In such a case, the gatekeeper sends a DRQ message to the endpoint. Upon receiving such a request, the endpoint must stop transmitting media and must bring the session to a close using the appropriate H.245 control messages and Q.931 call-signaling messages. Once the session is terminated, the endpoint responds to the gatekeeper with a DCF message.

Resource Availability

H.225.0 defines two messages related to resource availability: the *ResourceAvailableIndicate* (RAI) message and the *ResourceAvailableConfirm* (RAC) message. The first message is sent from a gateway to a gatekeeper to inform the gatekeeper of the currently available call capacity and bandwidth for each protocol supported by the gateway. Of particular importance is the mandatory Boolean parameter, almostOutOfResources, that, when set, indicates that the gateway is almost out of resources. It is the manufacturer's choice as to how close to capacity the gateway needs to be before this parameter is set. Equally, any action taken as a result of the parameter being set is also at the discretion of the system manufacturer. The gatekeeper acknowledges the RAI message with the RAC message.

Service Control

H.323 version 4 includes the message *ServiceControlIndication* (SCI) and *ServiceControlResponse* (SCR). These two messages are used to initiate a session between H.323 entities for the purpose of enabling some advanced features. H.323 does not specify what those features might be; it simply provides the SCI and SCR as a mechanism to enable advanced features. A particular system vendor might exploit the SCI and SCR to implement a vendor-specific capability.

Request in Progress

Finally, H.225.0 defines the *RequestInProgress* (RIP) message. It is a means of indicating that a given request may take longer than expected. H.225.0 specifies recommended timeout periods for various messages. If an entity cannot respond to a request within the applicable timeout period, then it should send an RIP message indicating the expected delay before it expects to send a response to the original request. If the entity can subsequently respond before the delay period expires, then it should do so.

If an entity expecting a response to a RAS message receives an RIP message instead, then it should wait for a response or for the RIP delay time to expire before sending the original message again. Note that it is perfectly legal and sometimes necessary to retransmit a RAS message. After all, RAS messages are sent using unreliable transport, as indicated in Figure 4-4. Therefore, the possibility exists that RAS messages can become lost.

Call Signaling

We have already mentioned call signaling and the fact that it is the signaling used between endpoints to enable the establishment and teardown of calls. The messages used are Q.931 messages, as modified by recommendation H.225.0.

At first reading, such a statement can be somewhat confusing. After all, Q.931 is the layer 3 signaling protocol for an ISDN user-network interface, and the various messages are defined in that recommendation. The fact is that H.225.0 takes advantage of the protocol defined in Q.931 and simply reuses the messages, with some modifications necessary for use in the overall H.323 architecture. H.225.0 also uses one Q.932 message. This approach is a pretty smart way of doing things, since it has avoided a great deal of protocol development effort. H.225.0 does not use all the messages defined in Q.931, just those that are necessary for the support of call-signaling functions in the H.323 architecture. The messages used are specified in Table 4-2.

H.225.0 specifies the modifications that are applicable to Q.931 messages when used for call signaling in an H.323 network. In general, relatively few changes are made to the body of the Q.931 messages. For

Table 4-2

Messages used by H.225.0 for call-signaling functions in the H.323 architecture

Message	Function	Comment
Alerting	Sent by called endpoint to indicate that the called user is being alerted.	Must be supported.
Call-Proceeding	Optional interim response sent by called endpoint or gatekeeper prior to sending of connect message.	Should be sent if a called endpoint uses a gatekeeper.
Connect	Indication that the called user has accepted the call.	Must be supported.
Progress	Optional message sent by called endpoint prior to connect.	Can be used by a called gateway in the case of PSTN interworking.
Setup	Initial message used to begin call establishment.	Must be supported.
Setup Acknowledge	Optional response to setup message.	Can be forwarded from a gateway in the case of PSTN interworking.
Release Complete	Used to bring a call to an end.	Q.931 Release message is not used.
Information	An optional message that is used to send additional call-establishment information.	Can be used in overlap signaling or to enable proprietary features.
Notify	Optional message that may be used to provide information for presentation to the user.	Can be used by calling or called endpoint.
Status	Sent in response to status inquiry or in response to an unknown message.	An optional message that can be sent in either direction.
Status Inquiry	A message used to query the remote end as to the current call status from its perspective.	Can be sent in conjunction with RAS status procedures.
Facility (Q.932)	Used to redirect a call or to invoke a supplementary service.	Can be sent by either end at any time during a call and is useful for conveying information when no other message can be sent.

example, H.225.0 does not define any exotic new information elements to be included in particular messages. On the other hand, H.225.0 does not take each Q.931 message as is either. Instead, H.225.0 specifies a number of rules regarding the usage of information elements defined in Q.931. For example, H.225.0 does not permit the use of the Transit Network Selection information element, nor does it currently support the use of High-Layer Compatibility and Low-Layer Compatibility information elements. In fact, the changes specified by H.225.0 generally involve specifying that certain Q.931 mandatory information elements are either forbidden or optional when used in an H.323 network.

The big question, however, is how H.225.0's use of Q.931 messages enables the transfer of H.323-specific information. For example, a clear need exists to transport information regarding gatekeepers and H.245 addresses to be used for logical channels. These types of items are foreign to the usual ISDN environment of Q.931 signaling, which means that information must be added to the Q.931 messages. This task is performed through a rather clever use of the User-to-User information element. This information element in Q.931 ISDN messages is used to pass information transparently from one user to another using the D channel. The network does not interpret the information. In H.225.0, the User-to-User information element is used to convey all the extra information needed in an H.323 environment. Such information includes parameters such as a mandatory protocol identifier, endpoint aliases, H.245 addresses, and so on. The ASN.1 syntax specified in H.225.0 provides the exact syntax to be used in the User-to-User information element of the various messages.

The following sections of this chapter provide a brief overview of some of the most important call-signaling messages. For complete details, the user should refer to the Q.931 and H.225.0 recommendations.

Setup

The Setup message is the first call-signaling message sent from one endpoint to another to establish a call. If an endpoint uses the services of a gatekeeper, then this message will be sent after the endpoint has received admission from the gatekeeper providing permission for the endpoint to establish the call. The message must contain the Q.931 Protocol Discriminator, a Call ReferenceSetup, a Bearer Capability, and the User-to-User information element.

Although the Bearer Capability information element is mandatory, the concept of a bearer as used in the circuit-switched world does not map very well to an IP network. For example, no separate B-channel exists in IP, and the actual agreement between endpoints regarding the bandwidth requirements is done as part of H.245 signaling, where RTP information such as the payload type is exchanged. Consequently, many of the fields in the Bearer information element, as defined in Q.931, are not used in H.225.0. Of those fields that are used in H.225.0, many are used only when the call has originated from outside the H.323 network and has been received at a gateway, where the gateway performs a mapping from the signaling received to the appropriate H.225.0 messages.

A number of parameters are included within the mandatory User-to-User information element. These include the call identifier, the call type, a conference identifier, and information about the originating endpoint. Among the optional parameters, we might find a source alias, a destination alias, an H.245 address for subsequent H.245 messages, and a destination call-signaling address. Note that these parameters may supplement information already contained within the body of the Q.931 message. For example, the optional Calling Party Number information element in the body of the message will contain the E.164 number of the source, which the source might also use as an alias.

Call-Proceeding

Call-Proceeding is an optional message that can be sent by the recipient of a Setup message to indicate that the Setup message has been received and that call establishment procedures are under way.

As is the case for all call-signaling messages, the Protocol Discriminator, Call Reference, and Message Type elements are mandatory. The only other mandatory information element is User-to-User. Within that information element, the mandatory pieces of information, besides the Protocol Identifier, are the destination information (which indicates the type of endpoint) and the call identifier. Among the optional parameters is the H.245 address of the called end, which indicates where it would like H.245 signaling messages to be sent.

Alerting

The called endpoint sends this message to indicate that the called user is being alerted. Besides the Protocol Discriminator, Call Reference, and Mes-

sage Type, only the User-to-User information element is mandatory. The optional Signal information element can be returned if the called endpoint wants to indicate a specific alerting tone to the calling party. The mandatory User-to-User information element in the Alerting message contains mostly the same parameters as those defined for the Call-Proceeding message.

Progress

The Progress message can be sent by a called gateway to indicate call progress, particularly in the case of interworking with a circuit-switched network. The Cause information element, though optional, is used to convey information to supplement any in-band tones or announcements that might be provided. The User-to-User information element contains the same set of parameters as defined for the Call-Proceeding message.

Connect

The Connect message is sent from the called entity to the calling entity to indicate that the called party has accepted the call. Although some of the messages from called party to calling party (such as Call-Proceeding and Alerting) are optional, the Connect message must be sent if the call is to be completed. The User-to-User information element contains the same set of mandatory parameters as defined for the Alerting message, with the addition of the Conference Identifier. This parameter is also used in a Setup message, and its use in the Connect message is to correlate this conference with that indicated in a Setup. Any H.245 address sent in a Connect message should match that sent in any earlier Call-Proceeding, Alerting, or Progress message.

Release Complete

The Release Complete message is used to conclude a call. Unlike ISDN, Release Complete is not sent as a response to a Release message. In fact, H.323 endpoints should never send a Release message. Instead, the Release Complete is all that is needed to bring the call to a close.

The Release Complete message contains the optional Cause information element. If the Cause element is not included, then the User-to-User information element will contain a Release reason. On the other hand, if no Release reason is specified in the User-to-User information element, then

the Cause information element should be included. One way or the other, the party who releases the call should provide the other party with a reason for the release.

Facility

The Facility message is actually defined in ITU-T recommendation Q.932. In H.225.0, the message is used in situations where a call should be redirected. The Facility message can also be used to request certain supplementary services as described in recommendation H.450.1. Within the User-to-User information element, one important item is the mandatory reason parameter, which provides extra information to the recipient of the Facility message. An important use of the parameter is when a Setup message has been sent to an endpoint, and that endpoint's gatekeeper indicates to the endpoint that it wishes to be involved in the call signaling. In such a case, the called endpoint would return a Facility message, with the reason set to routeCallToGatekeeper. This would cause the recipient of the Facility message to release the call and attempt to set up the call again via the called endpoint's gatekeeper.

Interaction Between Call Signaling and H.245 Control Signaling

Before delving into some examples of call signaling, we should note a few points regarding the interplay between call signaling and H.245 control signaling. As mentioned previously, Q.931 call-signaling messages are used for the establishment and teardown of calls between endpoints, and H.245 messages are used for the negotiation and establishment of media streams between call participants. Obviously, the two are closely tied together.

Using Q.931, call establishment begins with a Setup message, and the call is considered established upon return of a Connect message from the called entity. In the interim, between the receipt of the Setup message and the sending of the Connect message, the called party might issue one or more progress-related messages (such as Call-Proceeding or Alerting). Given that H.245 messaging must also take place, at which point during call signaling should it occur? In other words, which of the call-signaling messages should be used as a trigger to begin the exchange of H.245 messages?

The answer to this issue is equipment dependent. H.323 requires that H.245 messages be exchanged, but it does not mandate that the exchange of these messages must occur at any specific point within the call-signaling process. Obviously, since the Setup message is the first call-signaling message to be sent, H.245 messages cannot be exchanged before the called endpoint has received a Setup message. H.245 messages can be exchanged at any point after that, however. The called endpoint can initiate the exchange of H.245 messages upon receipt of a Setup message or the calling endpoint can initiate the exchange upon receipt of a Call-Proceeding, Progress, or Alerting message. If an H.245 message exchange has not begun by the time a Connect message is sent, then it should begin immediately thereafter.

Call Scenarios

Now that we have described RAS signaling and call signaling, we will present a number of call scenarios to show how the different types of messages are used in the establishment and teardown of calls. Granted, H.245 messaging has not yet been described, and H.245 messaging is a prerequisite for media exchange between the endpoints. We can assume in the following scenarios, however, that a successful H.245 message exchange takes place immediately after a Connect message is sent.

Basic Call Without Gatekeepers

A gatekeeper is an optional entity in H.323. In the absence of gatekeepers, call signaling happens directly between endpoints. Figure 4-10 shows a typical call establishment and teardown in the absence of gatekeepers. The call is initiated with a Setup message, answered by a Connect message, and released with a Release Complete message. Either party can send the Release Complete message. Note that no acknowledgement is necessary for the Release Complete message.

A Basic Call with Gatekeepers and Direct Endpoint Call Signaling

In the event that the endpoints are registered with a gatekeeper, then prior to establishment of call signaling, an endpoint must first gain permission

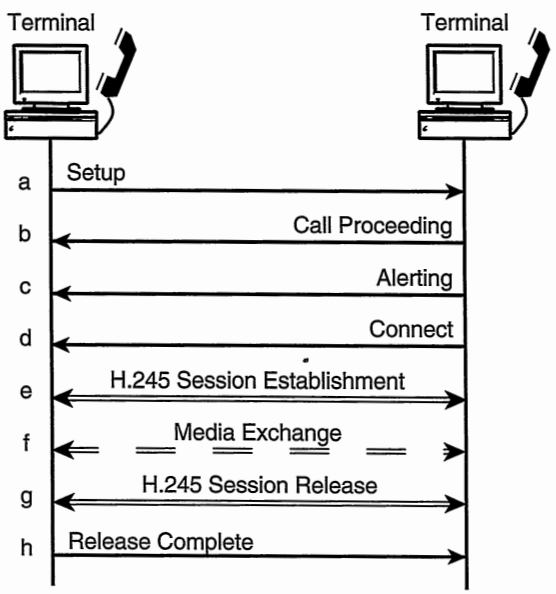


Figure 4-10
A basic call without gatekeepers

from its gatekeeper. The permission may be granted through the ARQ/ACF mechanism described earlier or may have been pregranted by the gatekeeper. Furthermore, at the end of the call, an endpoint must notify the gatekeeper that the call has been disconnected.

When gatekeepers are used in the network, a given endpoint may send call signaling directly to the other endpoint or it may send call signaling via its gatekeeper. Although an endpoint can indicate a preference, the gatekeeper chooses the path that the call signaling will take.

Figure 4-11 shows a scenario where two endpoints are controlled by different gatekeepers. Both gatekeepers have chosen that call signaling should be sent directly from endpoint to endpoint. We also assume that H.245 signaling goes from endpoint to endpoint. In the example of Figure 4-11, we assume that the gatekeeper has not pregranted access, which means that the originating endpoint must request access permission from its gatekeeper. It is only when permission is granted that the endpoint commences call signaling to the remote endpoint. Note that the remote endpoint immediately returns a Call-Proceeding message prior to the endpoint, asking its gatekeeper for admission to handle the call. Again, our example assumes that the gatekeeper has not pregranted admission as part of the registration of the endpoint with the gatekeeper. By sending back the Call-Proceeding message immediately, the endpoint signals to the call originator

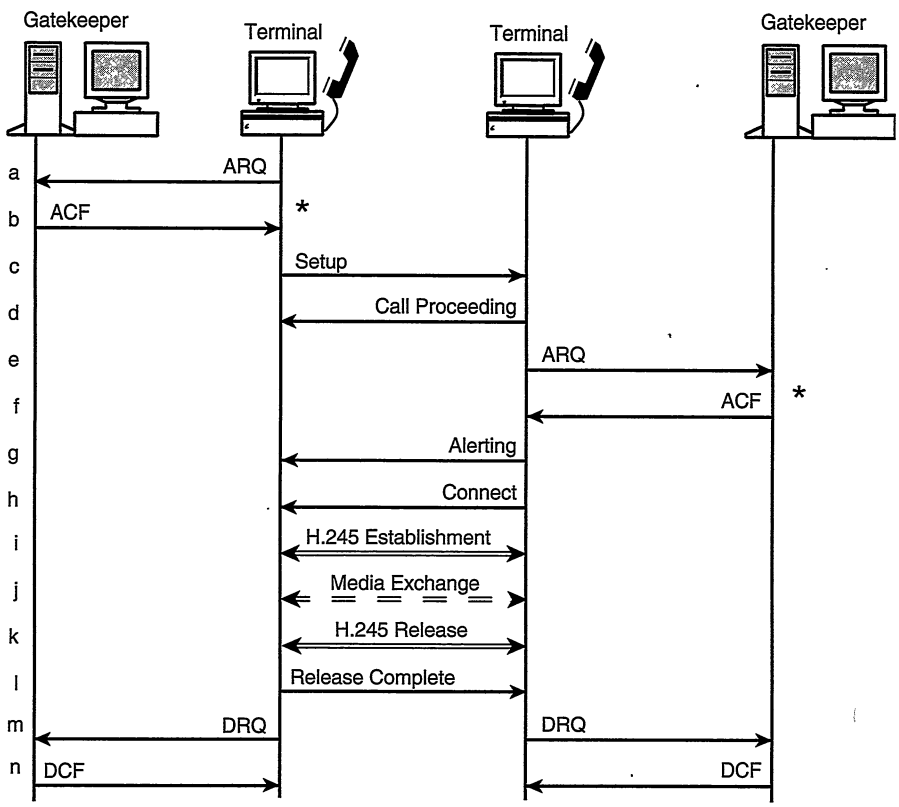


Figure 4-11
Direct endpoint call signaling

* ARQ/ACF message exchange is required only if admission has not been pregranted.

that the Setup has been received successfully. Thus, the calling endpoint is not left in the dark while the called endpoint requests permission from its gatekeeper to handle the call.

In the example, it is assumed that H.245 message exchange begins immediately after the Connect message. These messages could optionally occur earlier.

A Basic Call with Gatekeeper/ Direct Routed Call Signaling

A gatekeeper might choose that call signaling be routed via the gatekeeper rather than directly from endpoint to endpoint. In the case that the two

endpoints are connected to different gatekeepers, then it is very possible that one gatekeeper will choose to route the call signaling itself, while the other gatekeeper will not. Such a scenario is depicted in Figure 4-12, where the gatekeeper of the calling endpoint chooses to route the call signaling itself, while the gatekeeper of the called endpoint does not want to be in the path of call-signaling messages.

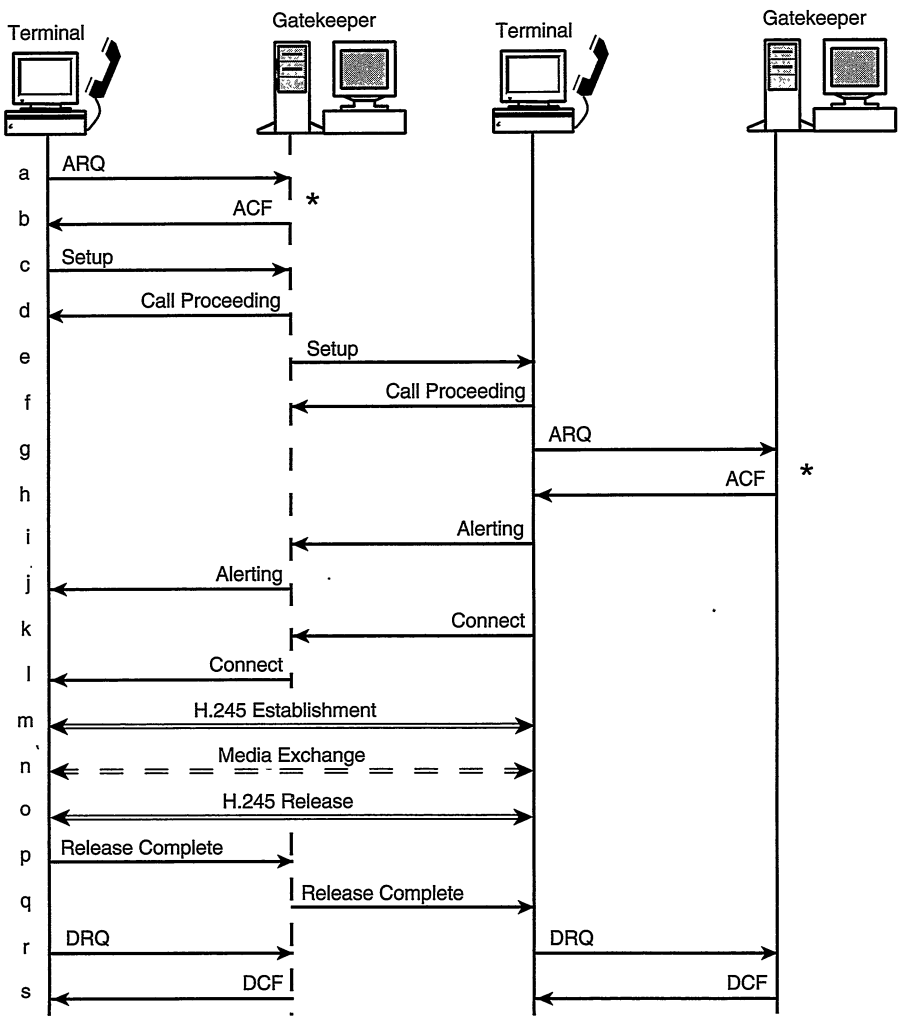


Figure 4-12 Gatekeeper routed/direct endpoint call signaling

* ARQ/ACF message exchange is required only if admission has not been pregranted.

In the first ACF message, the gatekeeper of the calling endpoint indicates that call signaling should be sent to the gatekeeper. The gatekeeper includes a call-signaling address of itself in the ACF. Note that the endpoint might have received pregranted admission from the gatekeeper. In such a case, and if the gatekeeper wants to be in the path of call signaling, then the pregranted admission would specify that call signaling should be routed via the gatekeeper.

The endpoint sends a Setup message to the gatekeeper and the gatekeeper responds immediately with Call Proceeding. The gatekeeper then sends a Setup message to the destination endpoint using the call-signaling address of that endpoint.

If the called endpoint wants to handle the call, it first responds with a Call-Proceeding message and then sends an ARQ to its gatekeeper. In this case, the gatekeeper responds with an ACF indicating direct call signaling as opposed to gatekeeper-routed call signaling. Again, the ARQ/ACF procedure is not necessary if the endpoint has pregranted admission.

The called endpoint alerts the user and sends an Alert message to the gatekeeper of the calling endpoint. That gatekeeper forwards the Alerting message to the calling endpoint. Once the called party accepts the call, the called endpoint sends a Connect message, which follows the same path as the Alert message.

At this point in the example, H.245 messages are exchanged in order to set up the media streams between the endpoints. Note that an H.245 message exchange can occur earlier depending on the options implemented in the two endpoints.

In the example, the H.245 messages are routed directly from endpoint to endpoint. However, they could, like call-signaling messages, be sent via one or both gatekeepers, depending on whether each gatekeeper wants to be in the path of the H.245 messages. It is important to remember that the actual RTP streams will pass directly from endpoint to endpoint. This is regardless of whether a gatekeeper is in the path of call signaling and/or H.245 control signaling. It is not required that the media streams follow the same path as the H.245 control messages.

A Basic Call with Gatekeeper-Routed Call Signaling

The signaling process is slightly more complex when the gatekeeper controlling the called endpoint wants to be in the path of call-signaling messages. Such a scenario is shown in Figure 4-13, where both gatekeepers are

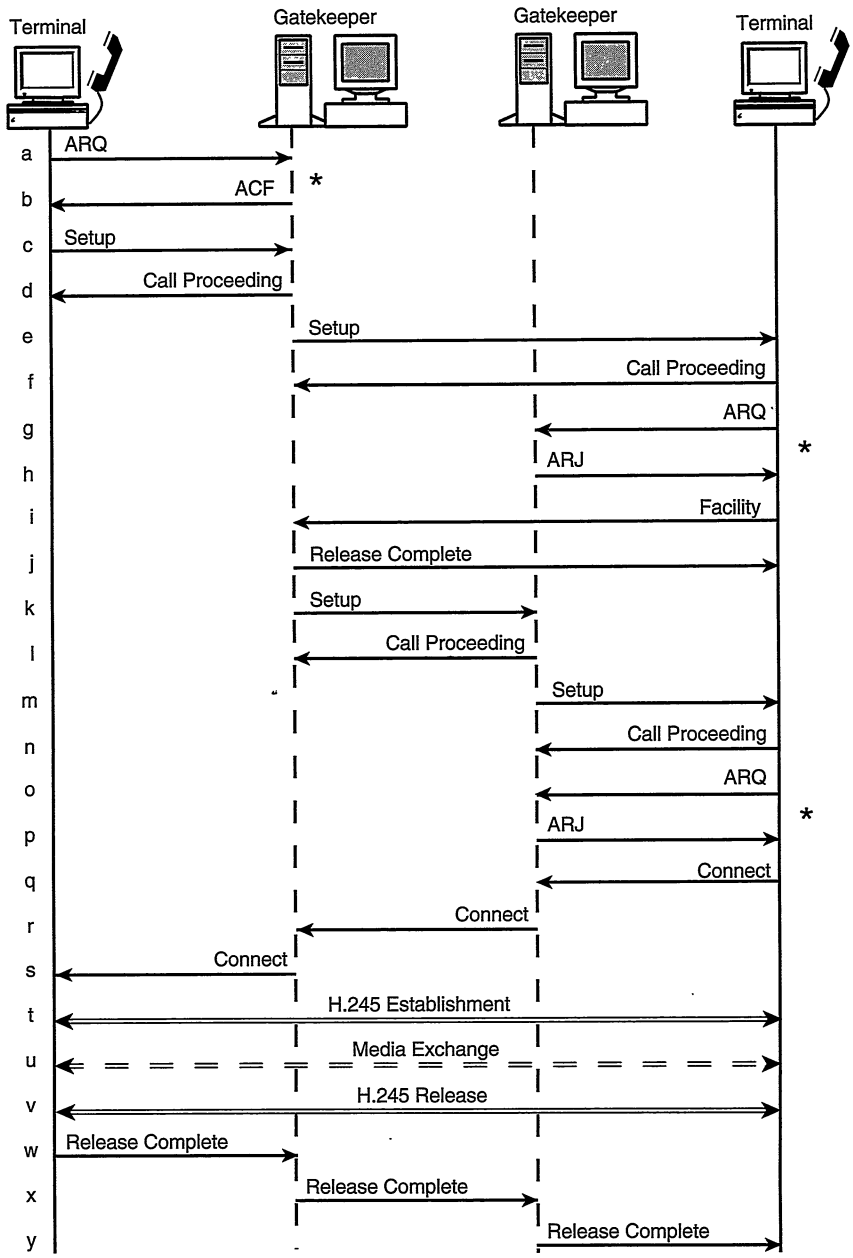


Figure 4-13
Gatekeeper-routed call signaling

* ARQ/ACF message exchange is required only if admission has not been pregranted.

involved in call signaling. Again, in the example, it is assumed that H.245 control messages are passed directly from endpoint to endpoint rather than via one or both gatekeepers.

Upon receipt of the Setup message from the calling endpoint's gatekeeper, the called endpoint requests permission from its own gatekeeper to handle the call (assuming that admission has not been pregranted). Since the gatekeeper wants to handle the call signaling, it returns an ARJ, with cause code of routeCallToGatekeeper. This action causes the called endpoint to send a Facility message to the gatekeeper of the calling endpoint. The Facility reason indicates that the call should be routed to the gatekeeper, and the call-signaling address of the gatekeeper is also included so that the calling gatekeeper knows where to send the call signaling. Note that a pregranted admission can include an indication that call signaling should be routed via a gatekeeper. If the called endpoint has pregranted admission with the requirement to route the call to the gatekeeper, then the endpoint will send the Facility message directly, without the need for the ARQ/ACF message exchange.

The gatekeeper on the originating side releases the call to the endpoint and attempts to establish the call to the gatekeeper on the terminating side instead. Upon receipt of the Setup message, the gatekeeper on the terminating side returns a Call-Proceeding message and forwards the Setup message to the called endpoint. The called endpoint immediately returns a Call-Proceeding message. If admission has not been pregranted, the called endpoint will again request permission to handle the call (ARQ).

Once the user is alerted, the Alerting message (not shown) is passed from the called endpoint, via the two gatekeepers to the calling endpoint. The same path is followed by the Connect message when the called user answers.

Once media have been exchanged, the Release Complete message is passed from one endpoint to the other via the two gatekeepers. Each endpoint then disengages using the DRQ/DCF message exchange with its gatekeeper (not shown).

Optional Called-Endpoint Signaling

The previous gatekeeper-routed example assumes that the gatekeeper of the calling endpoint knows the call-signaling address of the called endpoint.

As you can see in Figure 4-13, such a situation can lead to a rather cumbersome signaling exchange if the gatekeeper of the called endpoint wants to handle call signaling. Things do not need to be so complex, however.

Given that the two endpoints in our examples are controlled by different gatekeepers, it is very possible that the gatekeeper of the calling endpoint does not know the call-signaling address of the endpoint to be called. It may have only an alias for that endpoint, in which case it will have to issue an LRQ to the gatekeeper of the endpoint to be called, perhaps by multicast, in order to determine the call-signaling address of the endpoint.

If the gatekeeper of the terminating endpoint does not want to handle call signaling, then it will return the call-signaling address of the endpoint in the LCF. At that point, the call will proceed largely according to the example of Figure 4-12. On the other hand, if the gatekeeper wants to handle call signaling, it will return its own call-signaling address in the LRQ. In such an event, the sequence of events would be as depicted in Figure 4-14. Again, the Alerting message and the RAS Disengage sequence are not shown. As you can see, the call establishment from the beginning is in accordance with the desires of the remote gatekeeper. Therefore, the call establishment is much simpler and call redirection using the Facility message is not needed.

H.245 Control Signaling

We have mentioned the H.245 control protocol on several occasions in this chapter already. As described, H.245 is the protocol used between session participants to establish and control media streams. For a straightforward two-party voice call, this protocol ensures that participants agree on the media formats to be sent and received as well as the bandwidth requirements. For more complex multimedia calls, H.245 takes care of multiplexing multiple media streams for functions such as lip synchronization between audio and video.

You should note that H.245 is not responsible for carrying the actual media. For example, there is no such thing as an H.245 packet containing a sample of coded voice. That is the job of RTP. Instead, H.245 is a control protocol that manages the media sessions.

H.245 is not dedicated to use for VoIP. Rather it is a more generic protocol for the control of media streams and is designed to be used with a large number of applications. Consequently, the specification is a very large document. The intent in this book is not to describe H.245 in great

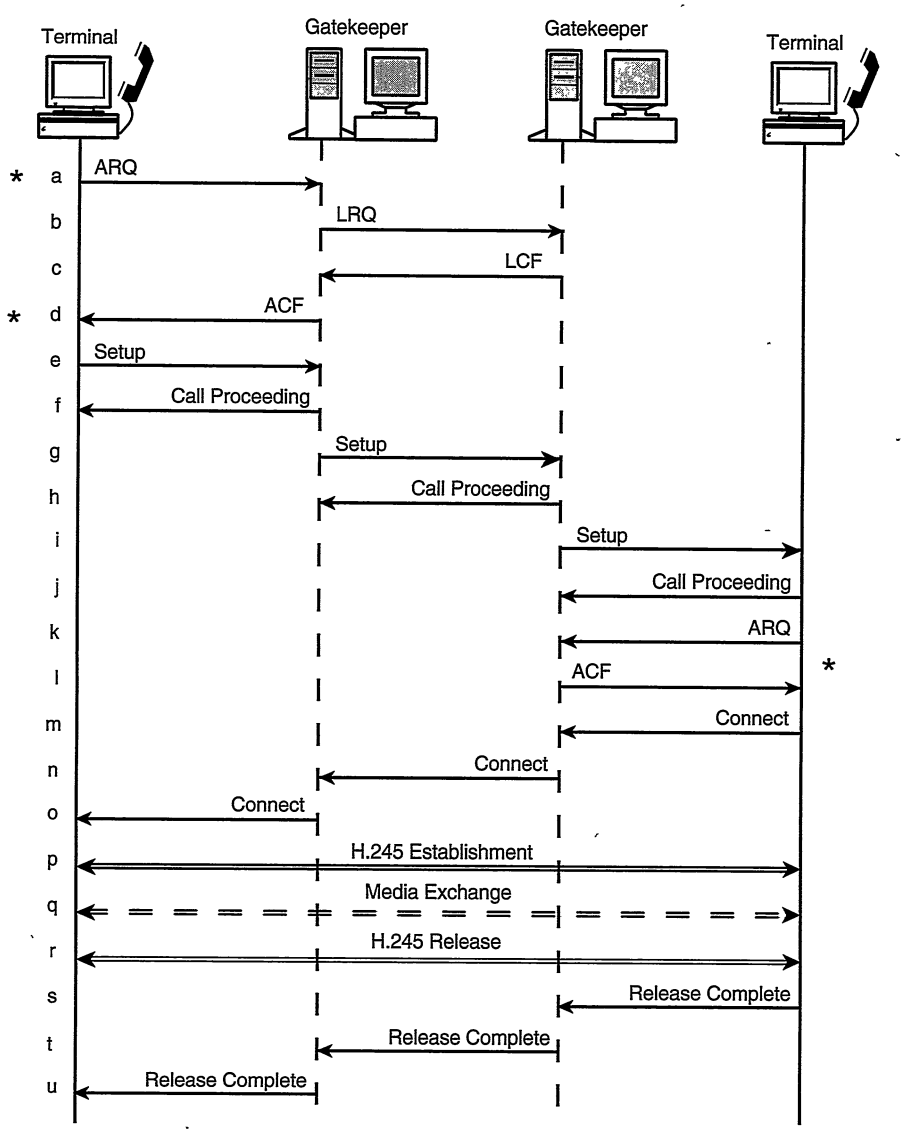


Figure 4-14
Optional called-endpoint signaling

* ARQ/ACF message exchange is required only if admission has not been pregranted.

detail, but rather to provide a general understanding of how it works in an H.323 environment and to provide descriptions of the most common messages and procedures.

H.245 Message Groupings

H.245 involves the sending of messages from one endpoint to another. These messages fall into four groups:[1]

- **Requests** These are messages that require the recipient to perform some action and to send an immediate response.
- **Responses** These messages are sent in reply to requests.
- **Commands** These are messages that require the recipient to perform some action. No explicit response is required.
- **Indications** These are messages that are of an informational nature only. The recipient of the message is not required to perform any specific action and no response is expected.

Only a selection of the most common H.245 messages is described in this chapter.

The Concept of Logical Channels

H.245 handles media streams through the use of logical channels. A logical channel is a unidirectional media path between two endpoints. Generally, in an IP environment, such a logical channel can be viewed as simply an IP address and port number supporting a particular type of media (such as G.729B-encoded voice). Each logical channel has a number that is specified by the sending entity.

Logical channels are unidirectional, from sender to receiver. Therefore, in a two-party conversation, two logical channels exist. This separation allows one terminal to potentially send voice in one format and receive voice in another format. Though logical channels are unidirectional, a perusal of the H.245 recommendation will find mention of a bidirectional channel. Such an entity is still two logical channels, though they are associated with each other.

To establish a media stream from one endpoint to another, the endpoint wanting to transmit information opens a logical channel. The endpoint does so by sending a message to the far end, indicating the logical channel num-

[1]The choices of names for these groups is a little unfortunate. At first glance, they have an uncanny resemblance to the service primitives request, indication, response, and confirm, as used in some SS7 applications. They are not the same thing.

ber and information about the media to be sent, such as RTP payload type. The message is called Open Logical Channel. If the far endpoint wants to receive the media, then it responds with a positive acknowledgement (Open Logical Channel Ack). Included in the acknowledgement will be an RTP port to which the media stream should be sent.

The H.245 messages are carried on the H.245 control channel. Each endpoint or gatekeeper shall establish one H.245 control channel for each call in which it is participating. The H.245 control channel is carried on a special logical channel, channel number 0. This is a special logical channel in that it is not opened and closed like other logical channels. Instead, logical channel 0 is considered permanently open as long as the endpoint is involved in a call.

H.245 Procedures

H.245 includes a number of procedures. For example, we have already briefly discussed the concept of opening logical channels. However, several other procedures are included in H.245, and some of them are briefly discussed here.

Before a logical channel can be opened, the sending endpoint should understand the capabilities of the receiving endpoint. There is no point in trying to send voice in a format that the receiving entity cannot handle. Consequently, H.245 provides a procedure known as Capabilities Exchange, which involves a set of messages that can be exchanged between endpoints to inform each other of their respective capabilities. Such capabilities include whether media can be sent or received (e.g. an endpoint might be able to receive video, but not send video), the media formats supported, and which formats can be supported simultaneously (particular combinations of voice and video).

In conference calls where two or more endpoints also have an MC, a potential problem can arise as to who is in control of the conference. Equally, imagine a situation where two endpoints have an urgent need to communicate with each other and both try to start a session with the other at the same time. To resolve these situations, H.245 has a procedure for master-slave determination.

Just because an endpoint can handle media in a particular format, it may have a preference for one format over another. Although capability exchange enables one endpoint to indicate to another which types of media it is able to handle, the Request Mode procedure enables an endpoint to rank formats in order of preference.

Capabilities Exchange Capabilities Exchange enables endpoints to share information regarding their receive and transmit capabilities. When a session is to be established between endpoints, the Capabilities Exchange procedure is started first. After all, there is no point in trying to open logical channels between two endpoints if they do not even speak the same language. Each endpoint should first determine what the other endpoint is able to support.

Receive capabilities indicate which media formats an endpoint is able to receive and process. Other endpoints should limit what they transmit to what the endpoint has indicated it can handle. Transmit capabilities indicate which media formats an endpoint is capable of sending. A remote endpoint can treat this information like a menu of possible media formats from which it can choose. The endpoint can subsequently indicate a preference for one format over another. If an endpoint does not indicate any transmit capabilities, then it must be assumed that the endpoint cannot and will not transmit any media.

An endpoint indicates its capabilities in a TerminalCapabilitySet message, which is a request message (and therefore requires a response). The message indicates a sequence number plus the types of audio and video formats that the endpoint can send and receive. The send and receive formats are indicated separately. Furthermore, the capabilities indicate what media can be handled simultaneously, such as a particular audio and video combination.

The TerminalCapabilitySet message is acknowledged by the TerminalCapabilitySetAck message. This message is empty except for a sequence number that matches the sequence number received in the original request. If the endpoint receiving the request finds a problem with the request, then it will respond with a TerminalCapabilitySetReject message indicating the reason for rejection. If no response is made within a timeout period, then the TerminalCapabilitySetRelease message is sent. This is an indication message and does not require a response.

As well as being able to unilaterally let another endpoint know about its capabilities, an endpoint can also request that another endpoint send information about its capabilities. This is done using the message SendTerminalCapabilitySet. Strictly speaking, this is a command message, and according to the definition of a command message, it does not require a specific response. However, the endpoint that receives the message should subsequently send a TerminalCapabilitySet message.

The SendTerminalCapabilitySet message contains two options. The sender can request that the far endpoint indicate all its capabilities. Alter-

natively, the sending endpoint can request confirmation about specific capabilities that the far endpoint has already described. This could be used, for example, in a situation where there has been a break in communication and one endpoint wants to make sure that it still has the latest information about the other.

Master-Slave Determination Situations can arise where two entities in a session want to initiate conflicting procedures. Consider, for example, a multiparty conference where several of the entities involved contain an MC and therefore could control the conference. For such situations, we need a method of deciding who is to be the master, which is done through the master-slave determination procedure.

The master-slave determination procedure involves two pieces of information at each entity. The first is a terminal type value, and the second is a random number between 1 and 16,777,215. The choice of the master is first made by comparing the values of terminal type. Whichever endpoint has the largest value of terminal type is automatically the master. H.323 specifies the values of terminal type that should be applied to different types of endpoints. For example, a terminal that does not have an MC has a terminal type value of 50. A gateway that does not contain an MC has a terminal type value of 60. An MCU that supports audio, video, and data conferencing has a terminal type value of 190 and an MCU that is currently managing a conference has the highest possible value, 240. If two entities have the same value of terminal type, then the random number is considered, and the terminal that chooses the highest random number value is the master.

In order to determine who is master, it is necessary that endpoints share their values of terminal type and random number. Such sharing is performed through the use of the Master-Slave Determination message and the Master-Slave Determination Ack message. Either entity may begin the master-slave determination procedure and does so by sending the Master-Slave Determination message containing a terminal type value and a random number. The receiving entity compares the value of terminal type with its own value and, if necessary, the received random number with another random number that the receiving entity chose itself. Based on these comparisons, the receiving entity makes the determination of who is master and returns a Master-Slave Determination Ack message.

The Master-Slave Determination Ack message indicates the sender's view of the other party's ranking. If it contains a master indication, it means that the receiver of the message is the master. If it contains a slave indication, it means that the receiver of the message is the slave.

Establishing and Releasing Media Streams Media streams between participants are carried on one or more logical channels. These channels need to be established before media can be exchanged and need to be closed at the end of a call.

Opening Unidirectional Logical Channels A logical channel is opened by sending an Open Logical Channel request message. This message contains a mandatory parameter called forwardLogicalChannelParameters, which relates to the media to be sent in the forward direction (from the endpoint issuing this request). The parameter contains information regarding the type of data to be sent (e.g., G.728 audio), an RTP session ID, an RTP payload type, an indication as to whether silence suppression is to be used, a description of redundancy encoding (if used), and so on.

If the recipient of the message wants to accept the media to be sent, then it will return an OpenLogicalChannelAck message containing at least the same logical channel number as received in the request and a transport address to which the media stream should be sent.

The process is shown in Figure 4-15, where a single logical channel is opened. The initiator of the logical channel has not indicated that it wants to receive media. We are not saying, however, that the initiator is unwilling to receive media. If the called endpoint also wants to send media, then after the first logical channel is opened, it can choose to open a logical channel in the reverse direction.

If an endpoint does not want to accept a request to open a channel, then it can reply with the OpenLogicalChannelReject message. This message contains the logical channel number received in the request and a reason for the rejection, such as an incapability to handle the proposed media format.

Opening Bidirectional Logical Channels The foregoing discussion regarding the opening of unidirectional logical channels mentions the fact that two endpoints can send media to each other by opening a logical channel in each direction. This requires four messages: two requests and two responses. If, however, an endpoint wants to send media and is expecting to receive media in return, as would be the case for a typical voice conversation, then it can attempt to establish a bidirectional logical channel. This procedure is a means of establishing two logical channels, one in each direction, in a slightly more efficient manner.

The process is depicted in Figure 4-16 and it begins with the sending of an Open Logical Channel request message. In addition to forward logical channel parameters, however, the message also contains reverse logical

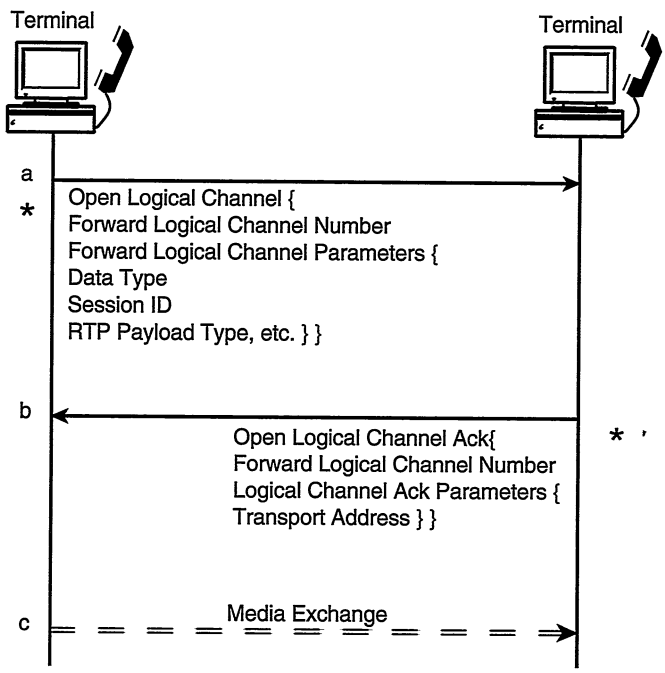


Figure 4-15
Opening a unidirectional logical channel

* Not all parameters are shown.

channel parameters that describe the type of media that the endpoint is willing to receive and where that media should be sent.

Upon receiving the request, the far endpoint can send an Open Logical Channel Ack message containing the same logical channel number for the forward logical channel, a logical channel number for the reverse logical channel, and descriptions related to the media formats that it is willing to send. These media formats should be chosen from the options originally received in the request, thereby ensuring that the called end will only send media that the calling end supports.

Upon receipt of the Open Logical Channel Ack, the originating endpoint responds with an Open Logical Channel Confirm message to indicate that all is well. Media flow can now begin in both directions.

Closing Logical Channels and Ending a Session Closing a logical channel involves the sending of a CloseLogicalChannel message. In the case of a successful closure, the far end should send the response message Close LogicalChannelAck.

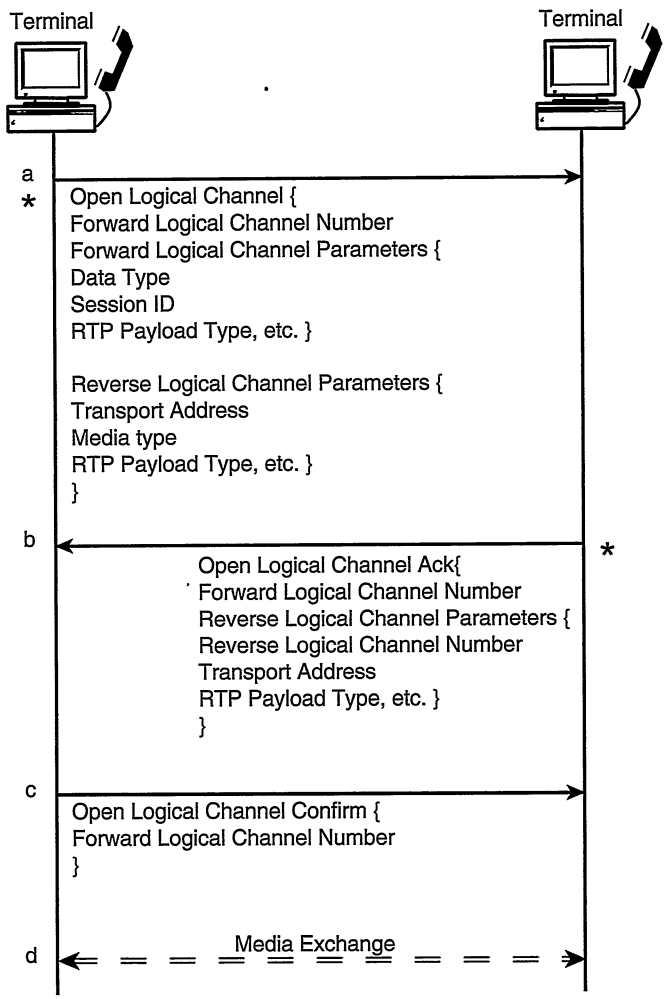


Figure 4-16
Opening bidirectional logical channels

* Not all parameters are shown.

In general, a logical channel can be closed only by the entity that created it in the first place. For example, in the case of a unidirectional channel, only the sending entity can close the channel. The receiving endpoint in a unidirectional channel can, however, request the sending endpoint to close the channel. It does so by sending the RequestChannelClose message, indicating the channel that the endpoint would like to have closed. If the sending entity is willing to grant the request, then it responds with a positive acknowledgment and then proceeds to close the channel.

One exception should be noted regarding the closure of a bidirectional channel. When an endpoint closes a forward logical channel, it also closes the reverse logical channel when the two are part of a bidirectional channel.

Once all logical channels in a session are closed, then the session itself is terminated when an endpoint sends an EndSession command message. The receiving endpoint responds with an EndSession command message. Once an entity has sent this message, it must not send any more H.245 messages related to the session.

Figure 4-17 provides an example of channel closure where one entity requests that the other close the channel, followed by the channel closure.

Fast Connect Procedure

Having reviewed RAS, Q.931, and H.245 signaling, it is possible to construct a call flow that incorporates more of the signaling required for call

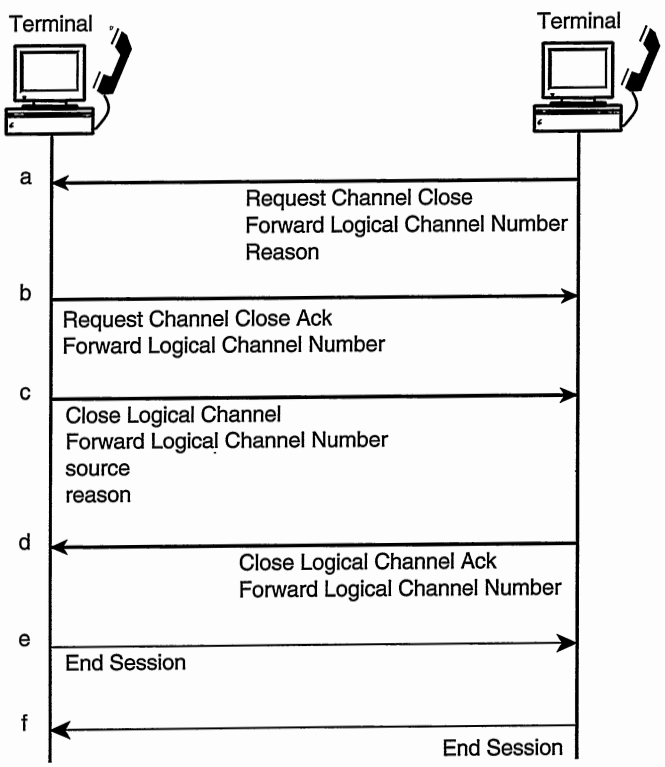


Figure 4-17
Closing a logical channel

establishment between two endpoints. Figure 4-18 shows an example of such a call establishment and can be considered a more complete version of the example depicted in Figure 4-11. One can see that many messages are required even for such a simple call (hence the term "slow start" in the caption). Furthermore, when we consider the fact that the figure does not show a Capabilities Exchange or possible master-slave determination, it becomes clear that even more messaging can be required.

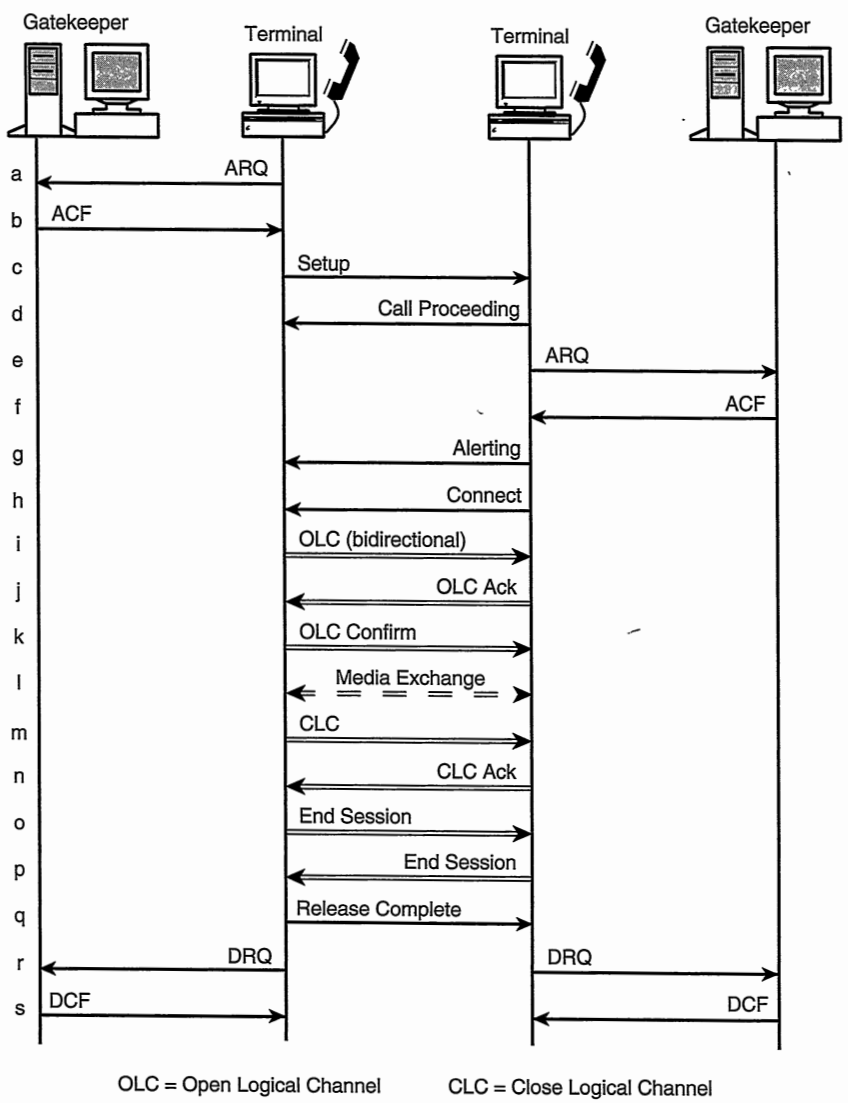


Figure 4-18
Combining RAS, Q.931, and H.245 signaling (slow start)

If one imagines a scenario with gatekeeper-routed call signaling and the need for the Facility message, as shown in Figure 4-13, plus the possibility of gatekeeper-routed H.245 control signaling, it is clear that the number of messages can be greater still. Call establishment becomes cumbersome and is certainly not conducive to fast call setup. Considering the fact that fast call setup is a requirement for carrier-grade operation, the situation is lamentable. Thankfully, the designers of H.323 recognized these issues and defined a procedure that reduces signaling overhead and speeds things up considerably. Appropriately, the procedure is known as the Fast-Connect procedure.

The Fast-Connect procedure involves setting up media streams as quickly as possible. To achieve this goal, the Setup message can contain a faststart element within the User-to-User information element. The faststart element is actually one or more OpenLogicalChannel request messages containing all the information that would normally be contained in such a request. The faststart element also includes reverse logical channel parameters if the calling endpoint expects to receive media from the called endpoint.

If the called endpoint also supports the procedure, then it can return a faststart element in one of the Call-Proceeding, Alerting, Progress, or Connect messages. That faststart element is basically another OpenLogicalChannel message, which appears like a request to open a bidirectional logical channel. The included choices of media formats to send and receive are chosen from those offered in the faststart element of the incoming Setup message. The calling endpoint has effectively offered the called endpoint a number of choices for forward and reverse logical channels, and the called endpoint has indicated those choices that it prefers. The logical channels are now considered open as if they had been opened according to the procedures of H.245. The use of faststart is depicted in Figure 4-19.

Note that the faststart element from the called party to the calling party may be sent in any message up to and including the Connect message. If the element has not been included in any of the messages, then the calling endpoint shall assume that the called endpoint either cannot or does not want to support Fast-Connect. In such a case, the standard H.245 methods must be used.

The use of the Fast-Connect procedure means that H.245 information is carried within the Q.931 messages and there is no separate H.245 control channel. Therefore, bringing a call to a conclusion is also faster. The call is released simply by the sending of the Q.931 Release Complete message. When used with the Fast-Connect procedure, this has the effect of closing

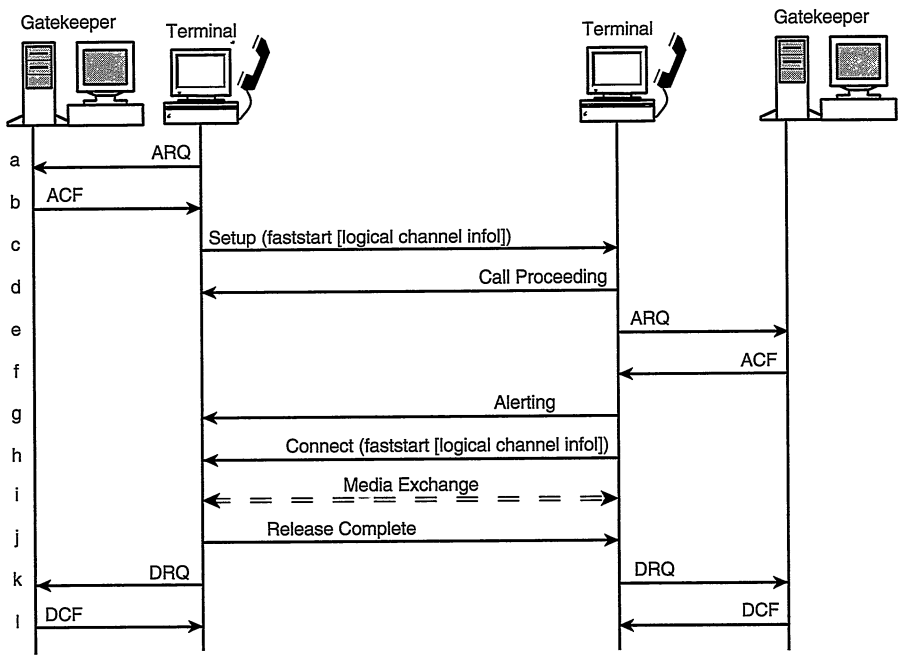


Figure 4-19
Fast connect procedure

all the logical channels associated with the call and is equivalent to using the procedures of H.245 to close the logical channels.

H.245 Message Encapsulation

In addition to the Fast-Connect procedure, H.323 also enables H.245 messages to be encapsulated with Q.931 messages as octet strings. An endpoint that wants to encapsulate such messages with Q.931 messages sets the element h245Tunneling to true in the first Q.931 message that it sends and keeps the element set to true in every other message that it sends as long as it wants to support such encapsulation. For a calling endpoint, this means that the h245Tunneling element should be set to true in the Setup message. Note that a called endpoint can use encapsulation only if the calling endpoint indicates support for encapsulation in the Setup message. The encapsulated data is actually contained within the h245Control element, which, like all H.323 information, is included in the User-to-User information element of the Q.931 message.

Note that a conflict exists between encapsulation and faststart. After all, the faststart element itself is a type of encapsulated H.245 message. Con-

sequently, a calling endpoint should not include a faststart element and an encapsulated H.245 message in a Q.931 Setup message. If, however, the endpoint wants to support encapsulation with later commands, then it should still set the h245Tunneling element to true such that the called endpoint knows that the calling endpoint can support encapsulation.

As is the case for the fast-connect procedure, the use of encapsulation means that there is no separate H.245 control channel, since H.245 messages are carried on the call-signaling channel. What happens if an endpoint needs to send an H.245 message at a time when it does not have a particular need to send a call-signaling message? This situation can be handled in one of two ways. First, an endpoint can send a Facility message with an encapsulated H.245 message. Alternatively, either endpoint can choose to change to a separate H.245 control channel.

An endpoint indicates a desire to switch to a separate H.245 control channel by populating the h245address element in a Q.931 message. In such a case, any existing logical channels established by Fast-Connect or through encapsulation shall be inherited by the H.245 control channel as if they had been established through normal H.245 procedures.

If the endpoint wants to change to a separate H.245 control channel and has no need to send a Q.931 message, then it has no means of indicating an H.245 address to the other endpoint. Again, the Facility message comes to the rescue, because that message can be sent at any time. All that is necessary is to send a Facility message with a FacilityReason of startH245 and with the h245address element populated with the endpoint's own H.245 address. This effectively establishes a separate H.245 control channel. Any subsequent Q.931 messages must have the h245Tunneling element set to false as anything else would be inconsistent with a separate H.245 control channel.

Conference Calls

The previous descriptions and examples have focused on the signaling necessary for the establishment and release of two-party calls. Situations will occur, however, when several parties are involved in a multipoint conference. In fact, H.323 defines the MC for the purpose of managing multipoint conferences. A conference call may be established in two ways. One is a prearranged conference, where participants call in to a separate MCU, which controls the conference. The other is when a two-party call needs to be expanded to include other parties. This is known as an ad hoc conference.

Pre-arranged Conference

In a pre-arranged conference, the individual participants are connected to the conference through the establishment of a call with the MCU. In other words, connections to the MCU are established through Q.931 call signaling. The sharing of media streams can be achieved in a centralized manner via the MCU or in a decentralized multicast manner. In the centralized scenario, both signaling and media pass via the MCU. In the decentralized scenario, only the signaling passes via the MCU, while the media streams are multicast from each endpoint to each other endpoint. The MCU determines the mode of the conference.

The MCU specifies the conference mode through the use of the H.245 command message Communication Mode command. This command specifies all the sessions in the conference (and for each session, various session data plus a unicast or multicast address). Note that the command only specifies the transmit requirements of each endpoint, not the receive requirements. The reason is because the receive requirements will be specified in separate Open Logical Channel commands that will be subsequently sent from the MC to the various endpoints.

Ad Hoc Conference

Quite often, the need will arise to expand an existing two-party call to a conference involving three or more participants. However, such an occurrence might not be foreseen at the beginning of the two-party call. Given that most PBXs and even residential phones support multiparty and the expansion of a call to include other parties, the same capability must be supported in a VoIP environment.

Many H.323 endpoints or gatekeepers include an MC function. It is not a callable entity like an MCU used for a call-in conference, but it can be utilized in the event that a two-party call needs to be expanded to three or more parties. In order for a two-party call to be expanded to an ad hoc conference, one of the endpoints must contain an MC or one of the endpoints' gatekeepers must contain an MC (in which case gatekeeper-routed call signaling must be used).

If these requirements are met, then the two-party call can be established as a conference call between two entities. This requires that the Setup message from endpoint 1 to endpoint 2 contain a unique *Conference ID* (CID).

If endpoint 2 wants to accept the call, then the Connect message that is returned should contain the same value of CID. H.245 capability exchange and master-slave determination procedures are performed, and the call is established as shown in Figure 4-20.

If, subsequently, the endpoint that is the master decides to invite another party (such as endpoint 3) to join the conference, then it sends a Setup message to endpoint 3, with the CID value of the conference and the conference goal parameter set to "invite." Assuming that endpoint 3 wants to join the conference, then it includes the same CID in the Connect message. Then capability exchange and master-slave determination is performed between endpoint 1 and endpoint 3. Finally, the MC at endpoint 1 sends the H.245 indication Multipoint Conference to each of the participants.

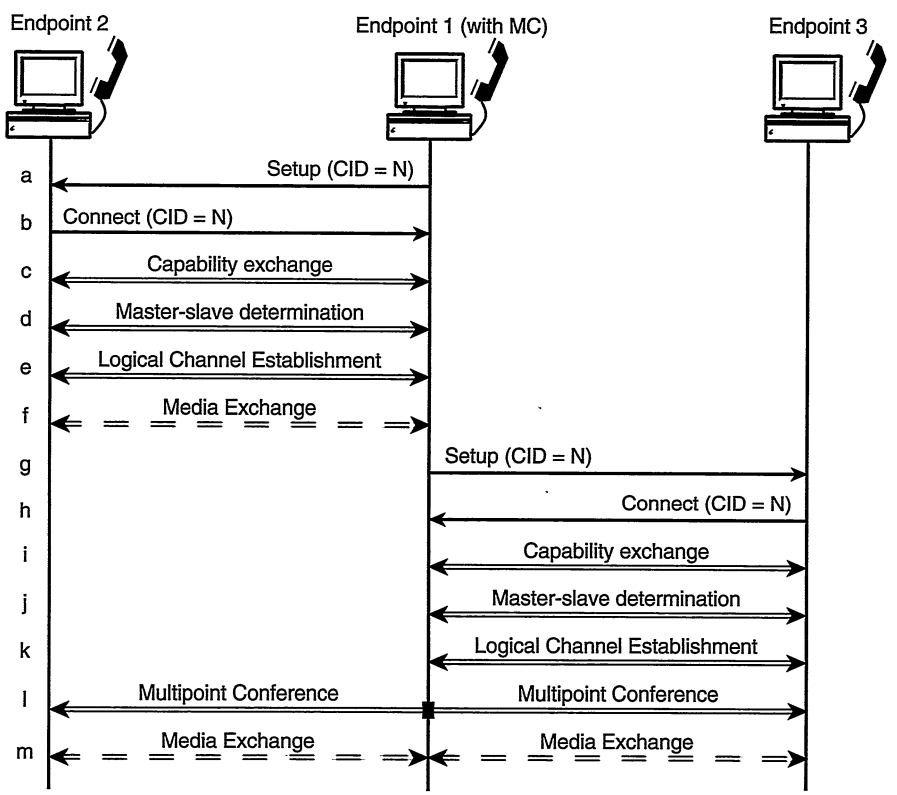


Figure 4-20
An ad hoc conference

The Decomposed Gateway

Previous descriptions have considered a gateway to be a single monolithic entity. In some cases, a gateway is a single entity. In fact, early versions of H.323 considered a gateway to be a single unit. As we shall see in Chapter 6, "Media Gateway Control and the Softswitch Architecture," however, a gateway can be comprised of several individual elements. An H.323 gateway can comprise a *media gateway* (MG) that handles media streams, a *media gateway controller* (MGC) that performs call control, and a *signaling gateway* (SG) that interfaces with external signaling networks such as the SS7 network. These components can reside on separate physical machines.

Depending on the specific application, an H.323 gateway might contain all the components in a single machine, it might contain two components in a single machine, with an interface to the third component on a separate machine, or the gateway might be completely disaggregated into three separate machines. For example, a gateway that interfaces towards PBXs will contain both a media gateway and signaling functionality (since PBXs usually employ channel-associated signaling that is carried on the same transmission facilities as the voice traffic), and it might interface with a separate MGC. A trunking gateway that interfaces with an SS7-based network might be decomposed into separate MGs, SGs, and MGC. Figure 4-21 depicts a situation where all three functions reside on separate machines.

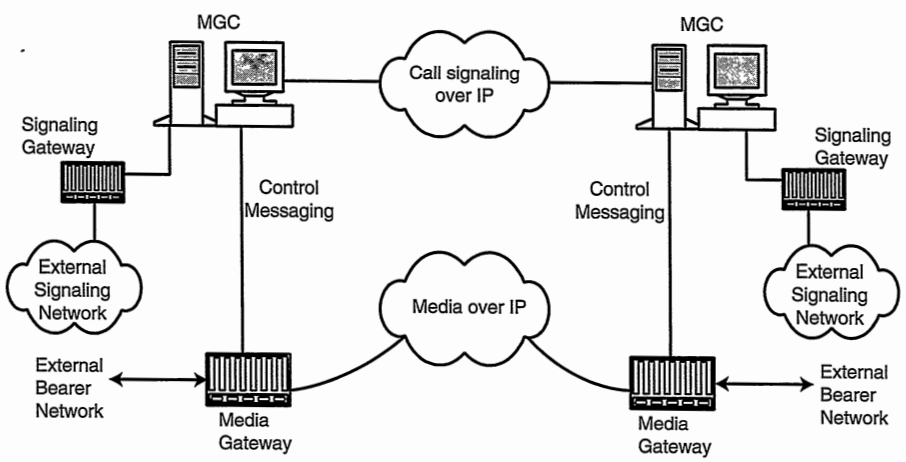


Figure 4-21
A decomposed gateway

Note that the function of the MGC is separate from that of a gatekeeper. Although a gatekeeper is an optional entity, the functions of an MGC must be performed, whether on a separate machine or as part of a monolithic gateway machine. Standard protocols exist for the MGC-MG interface and for the MGC-SG interface. Those protocols are described in later chapters.

CHAPTER 5

The Session Initiation Protocol (SIP)

Many consider the *Session Initiation Protocol* (SIP) a powerful alternative to H.323. They claim that SIP is a more flexible solution, simpler than H.323, easier to implement, better suited to the support of intelligent user devices, and better suited to the implementation of advanced features. These factors are of major importance to any equipment vendor or network operator. Simplicity means that products and advanced services can be developed faster and made available to subscribers sooner. The features themselves mean that operators are better able to attract and retain customers, and they also offer the potential for new revenue streams.

SIP is designed to be a part of the overall *Internet Engineering Task Force* (IETF) multimedia data and control architecture. As such, SIP is used in conjunction with several other IETF protocols such as the *Session Description Protocol* (SDP), the *Real-Time Streaming Protocol* (RTSP), and the *Session Announcement Protocol* (SAP).

The Popularity of SIP

Originally developed in the *Multiparty Multimedia Session Control* (MMUSIC) working group of the IETF, great interest in SIP led to the creation of a separate SIP working group. SIP was defined in *Request for Comments* (RFC) 2543, which has seen a number of revisions as the specification has proceeded from proposed standard status towards draft standard status. The number of proposed enhancements to SIP as it proceeds along the standards track is one indication of the attention it has received. Many developers are including SIP in their products without waiting for those enhancements, simply because SIP is seen as the way of the future for *Voice over IP* (VoIP) signaling. In fact, many believe that SIP, in conjunction with a *Media Gateway Control Protocol* (MGCP or MEGACO) as described in Chapter 6, "Media Gateway Control and the Softswitch Architecture," will be the dominant VoIP signaling architecture of the future.

A new SIP RFC has been produced with RFC number 3261. This specification includes a number of enhancements over the RFC 2543 specification. The descriptions provided in this chapter are based on the latest proposed enhancements to SIP as of May 2002 that are included within RFC 3261 or published as separate RFCs.

The development of SIP and its implementation by system developers has involved a number of events known as "bake-offs" or *SIP Interoperability Tests* (SIPit).[1] During these events, various vendors come together and

test their products against each other to ensure that they have implemented the specification correctly and to ensure compatibility with other implementations. To give an indication of SIP's popularity, Figure 5-1 shows how many companies have been involved in the first eight events. Although attendance at the most recent events has diminished slightly, it is clear that significant industry support exists for SIP. In fact, although the number of attending companies might have peaked, the number of attending individuals continues to increase.

The SIP Architecture

SIP is a signaling protocol that handles the setup, modification, and teardown of multimedia sessions. SIP, in combination with other protocols, is used to describe the session characteristics to potential session participants. Although strictly speaking, SIP is written such that the media to be used in a given session could use any transport protocol, the media will

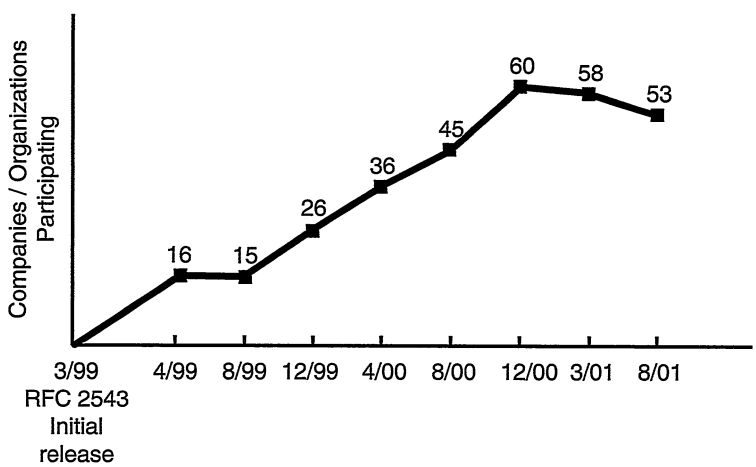


Figure 5-1
Participation in SIPit

[1]The first seven events were known as bake-offs. The Pillsbury company claimed, however, that the term "bake-off" is proprietary to that company. Therefore, the term was dropped for subsequent interoperability-testing events.

commonly be exchanged using the *Real-Time Transport Protocol* (RTP) as the transport protocol.

It is likely that SIP messages will pass through some of the same physical facilities as the media to be exchanged. SIP signaling should be considered separately from the media itself, however. Figure 5-2 shows the logical separation between signaling and session data. This separation is important, because the signaling may pass via one or more proxy or redirect servers while the media stream itself takes a more direct path. This approach can be considered somewhat analogous to the separation of signaling and media already described for H.323.

SIP Network Entities

SIP defines two basic classes of network entities: clients and servers. Strictly speaking, a client (also known as a user agent client) is an application program that sends SIP requests. A server is an entity that responds to those requests. Thus, SIP is a client-server protocol. VoIP calls using SIP are originated by a client and terminated at a server. A client may be found within a user's device, which could be a PC with a headset attachment or a SIP phone, for example. Clients may also be found within the same platform as a server. For example, SIP enables the use of proxies, which act as both clients and servers.

Four different types of servers exist: proxy server, redirect server, user-agent server, and registrar. A proxy server acts in a similar way to a proxy server used for web access from a corporate *local area network* (LAN). Clients send requests to the proxy and the proxy either handles those requests itself or forwards them on to other servers, perhaps after performing some translation. To those other servers, it appears as though the message is coming from the proxy rather than some entity hidden behind it.

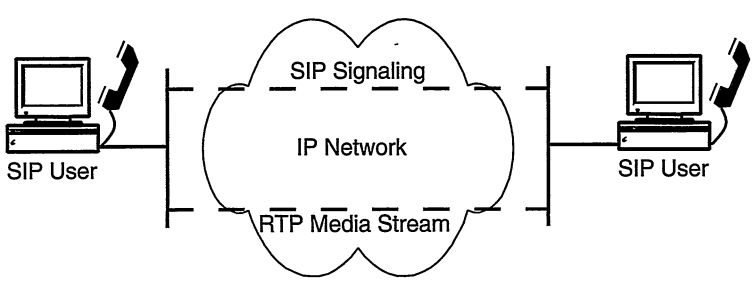


Figure 5-2
The separation of signaling and media

Given that a proxy both receives requests and sends requests, it incorporates both server and client functionality.

Figure 5-3 shows an example of the operation of a proxy server. It does not take much imagination to realize how this type of functionality can be used for call forwarding, time-of-day routing, or follow-me services. For example, if the message from the caller to Collins in Figure 5-3 is an invitation to participate in a call, the net effect is that the call is forwarded to Collins at home. Of course, it is necessary that the proxy be aware that Collins happens to be at home instead of at work.

A redirect server is a server that accepts SIP requests, maps the destination address to zero or more new addresses, and returns the new address to the originator of the request. Thereafter, the originator of the request can send requests directly to the address(es) returned by the redirect server. A redirect server does not initiate any SIP requests of its own. Figure 5-4 shows an example of the operation of a redirect server. This can be another means of providing the call-forwarding/follow-me service that can be provided by a proxy server. The difference is that, in the case of a redirect server, the originating client does the actual forwarding of the call. The redirect server simply provides the information necessary to enable the originating client to correctly route the call, after which the redirect server is no longer involved.

A user-agent server accepts SIP requests and contacts the user. A response from the user to the user-agent server results in a SIP response on behalf of the user. In reality, a SIP device (such as a SIP-enabled telephone) will function as both a user-agent client and a user-agent server. Acting as a user-agent client, it is able to initiate SIP requests. Acting as a user-agent

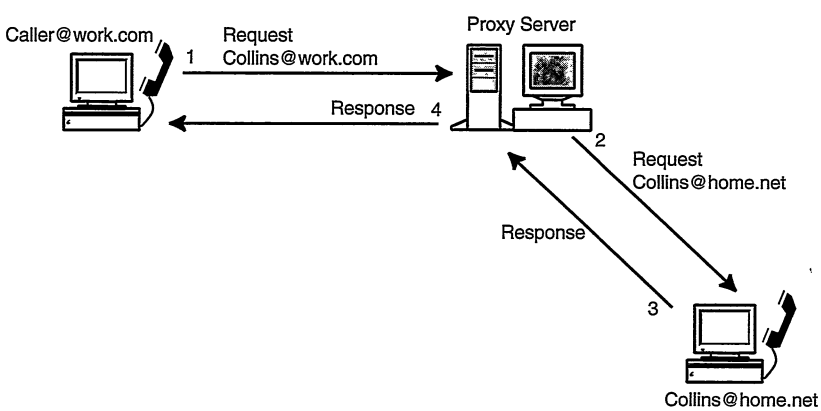


Figure 5-3
SIP proxy operation

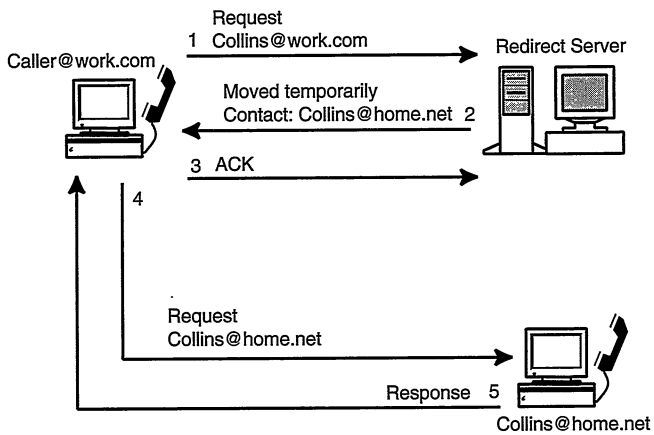


Figure 5-4
SIP redirect operation

server, it is able to receive and respond to SIP requests. In practical terms, this means that it is able to initiate and receive calls. This enables SIP (a client-server protocol) to be used for peer-to-peer communication.

A registrar is a server that accepts SIP REGISTER requests. SIP includes the concept of user registration, whereby a user indicates to the network that he or she is available at a particular address. The use of registration enables SIP to support personal mobility. For example, a user could have several SIP devices, one of which might be the user's office PC. When the user logs onto the office network, then the PC would issue a SIP REGISTER request to the appropriate registrar. Thereafter, calls can be routed to the user's office PC. When the user leaves the office, he or she might register at a different device, such as a home PC or SIP phone. A new registration would then be performed, enabling the user to be reached at the new device.

Typically, a registrar will be combined with a proxy or redirect server. Given that practical implementations will involve the combining of a user-agent client and a user-agent server as well as the combining of registrars with either proxy servers or redirection servers, a real network may well involve only user agents and redirection or proxy servers.

SIP Call Establishment

At a high level, a SIP call establishment is simple, as shown in Figure 5-5. The process starts with a SIP INVITE message, which is used from calling party to called party. The message invites the called party to participate in

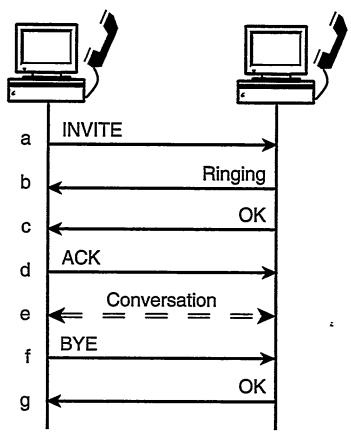


Figure 5-5
An example of a SIP call

a session, that is, a call. A number of interim responses may be made to the INVITE prior to the called party accepting the call. For example, the caller might be informed that the call is queued and/or that the called party is being alerted (the phone is ringing). Subsequently, the called party answers the calls, which generates an OK response back to the caller. The calling client acknowledges that the called party has answered by issuing an ACK message. At this point, media are exchanged. This media will most often be regular speech, but could also be other media, such as video. Finally, one of the parties hangs up, which causes a BYE message to be sent. The party receiving the BYE message sends OK to confirm receipt of the message. At that point, the call is over.

SIP Advantages over Other Signaling Protocols

Given that the call establishment and release is pretty straightforward, one might wonder what is so special about SIP. After all, any call-signaling protocol must have a means for one party to call another, a means to indicate acceptance of the call, and a means to release the call. SIP performs these actions and does a little more, but unlike many traditional telephony protocols, SIP offers a great deal of flexibility. For example, SIP does not care what type of media is to be exchanged during a session or the type of transport to be used for the media. In fact, SIP itself can be carried over several different transport protocols. In other words, SIP provides more flexibility

than is found in typical telecommunications protocols. That flexibility can be exploited to enable custom services and features.

SIP messages can include a number of optional fields, which can contain user-specified information. This approach enables users to share nonstandard user-specific information, which enables users and devices to make intelligent call-handling decisions. For example, a SIP INVITE can include a subject field. A person who receives an INVITE might decide to accept or reject the call depending on who is calling and what the subject happens to be. One can imagine a situation where an INVITE contains a text string such as "I know you are there. Please answer the phone." If the call is rejected, the response might contain a text string such as "Stop calling me!"

Imagine another scenario where a call is directed to a user who is currently not available. Obviously, the SIP response will indicate that the user is unavailable (no big surprise). The response could, however, include an indication that the user expects to be available again at 4 P.M. In such a case, the calling terminal could do two things. First, the terminal could tell the calling party that the called user expects to be available at 4 P.M. Second, at 4 P.M., it could ask the caller if he/she wants to make the call again and, if so, the terminal can automatically set up the call. Such a scenario is shown in Figure 5-6. This is a clever and simple mechanism in the IP world of providing a call-completion service, similar to the *Call Completion to Busy Subscriber* (CCBS) of the telephony world. It is also a lot more effi-

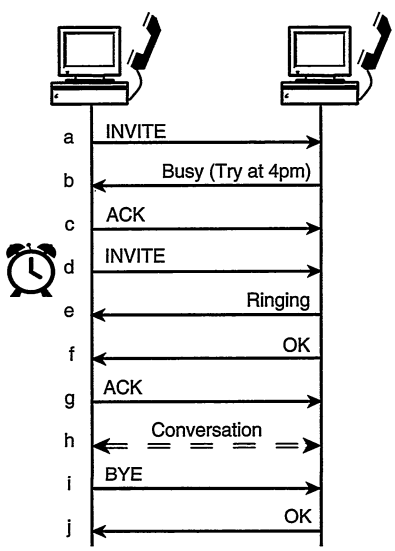


Figure 5-6
An example of a SIP-enabled service

cient than calling back every 15 minutes in the hope that the person will finally answer or paying the telephone company to keep trying to connect the call.

The foregoing are some simple examples of capabilities that SIP offers. Since SIP provides many pieces of information for inclusion in messages, and because additional, nonstandard information can also be included, the opportunity exists to offer numerous intelligent features to subscribers. Moreover, the control of those features is placed in the hands of the customer. No longer does a customer need to subscribe to a particular feature and have that feature actually controlled by the network operator.

The following sections of this chapter are devoted to describing the fundamentals of SIP. From this description, the reader should gain a good understanding of how SIP works, how it compares to other standards (such as H.323), and how it can be used to provide new and advanced features to users.

Overview of SIP Messaging Syntax

As mentioned, SIP is a signaling protocol. As such, it has a particular syntax. In the case of SIP, the syntax is text-based, using the *International Organization for Standardization* (ISO 10646) character set, and has a similar look and feel to the *Hypertext Transfer Protocol* (HTTP). One advantage of this approach is that programs designed for the parsing of HTTP can be adapted relatively easily for use with SIP. One obvious disadvantage, compared to a binary encoding, is that the messages themselves consume more bandwidth.

SIP messages are either requests from a client to a server or responses (which are also known as status messages) from a server to a client. Each message, whether a request or a response, contains a start-line, possibly followed by headers and a message body:

```
message = start-line
                *message-header
                CRLF
                [message-body]
```

Given that SIP defines only request and status messages, the start-line is

```
Start-line = request-line | status-line
```

The request-line specifies the type of request being issued, while the response line indicates the success or failure of a given request. In the case of failure, the status line indicates the type of failure or the reason for failure.

Message headers provide additional information regarding the request or response. Information that is obviously required includes the message originator and the intended message recipient. Message headers also offer the means for carrying additional information, however. For example, the Retry-after header indicates when a request should be attempted again. In the example of Figure 5-6, the Retry-after header would convey the fact that the called user expects to be available again at 4 P.M. Another example of a useful header is the Subject header, which enables a caller to indicate the reason for a call.

The message body normally describes the type of session to be established, including a description of the media to be exchanged. Thus, for a given call, the message body might indicate that the caller wants to communicate using voice, coded according to G.711 A-law. Note, however, that SIP does not define the structure or content of the message body. This structure and content is described using a different protocol. The most common structure for the message body is the *Session Description Protocol* (SDP) described later in this chapter. The message body could, however, contain information coded according to another standard. In fact, the message body could contain multiple parts, with each part coded according to a different format or structure.

This capability is used in some situations to carry an *ISDN User Part* (ISUP) message in binary format, thereby enabling SIP to carry ISUP information. Carrying an ISUP message within a SIP message body would be used, for example, in a scenario where the SIP network is interworking with the *Public Switched Telephone Network* (PSTN) using *Signaling System 7* (SS7) signaling. SIP does not greatly care about what the message body happens to say. It is concerned only with making sure that the message body is carried from one party to the other. It is at the two ends that the message body is examined. The message body can be considered to be within a sealed envelope. SIP carries it from one end to the other, but does not look inside the envelope.

SIP Requests

A SIP request begins with a request-line, which comprises a method token, a Request-URI, and an indication of the SIP version. The method token identifies the specific request being issued and the Request-URI is the

address of the entity to which the request is being sent. The three components of the request-line are separated by spaces, and a CRLF character terminates the line itself. Thus, the syntax is as follows:

```
request-line = method SP Request-URI SP SIP-Version CRLF
```

RFC 3261 defines six different methods (and hence six different types of requests): INVITE, ACK, OPTIONS, BYE, CANCEL, and REGISTER. A number of extensions to SIP specify additional methods, such as INFO, REFER, and UPDATE.

The INVITE method is used to initiate a session. For a simple call between two parties, the INVITE is used to initiate the call, with the message including information regarding the calling and called parties as well as the type of media to be exchanged. For those who are familiar with ISUP, the INVITE can be considered akin to the *Initial Address Message* (IAM). As well as a means for initiating a simple two-party call, INVITE can also be used to initiate a multiparty conference call.

Once it has received a final response to an INVITE, the client that initiated the INVITE sends an ACK. The ACK method is used as a confirmation that the final response has been received. For example, if the response to an INVITE indicates that the called user is busy and the call cannot be completed at that time, then the calling client will send an ACK. On the other hand, if the response to the INVITE indicates that the called user is being alerted or that the call is being forwarded, then the client does not send ACK, because such responses are not considered final.

The BYE method terminates a session. The method can be issued by either the calling or called party and is used when the party in question hangs up.

The OPTIONS method queries a server as to its capabilities. This method could be used, for example, to determine whether a called user agent can support a particular type of media or how a called user agent would respond if sent an INVITE. In such a case, the response might indicate that the user can support certain types of media or perhaps that the user is currently unavailable.

The CANCEL method is used to terminate a pending request. For example, CANCEL could be used to terminate a session where an INVITE has been sent, but a final response has not yet been received. As discussed later in this chapter, it is possible to initiate a parallel search to multiple destinations (if a user is registered at multiple locations). In such a case, if a final response has been received from one of the destinations, then CANCEL can be used to terminate the pending requests at the other destinations.

The REGISTER method is used by a user-agent client to log in and register its address with a SIP server, thereby letting the registrar know the

address at which the user can be reached. The user-agent client might register with a local SIP server at startup, with a known registrar server whose address is configured within the user agent, or by multicasting to the "all-SIP-servers" multicast address (224.0.1.175).

A client can register with multiple servers and a given client can have multiple registrations at a single server. Such a situation would occur in the event that a user has logged in at several terminals or devices. If a user has multiple active registrations, then calls to the user may be sent to all registered destinations. This capability can enable a "one-number" service where a user publishes just a single number, but when that number is called, the user's office phone, home office phone, and wireless phone all ring. Figure 5-7 shows an example of how this would work and also shows the usage of the CANCEL message in such a scenario.

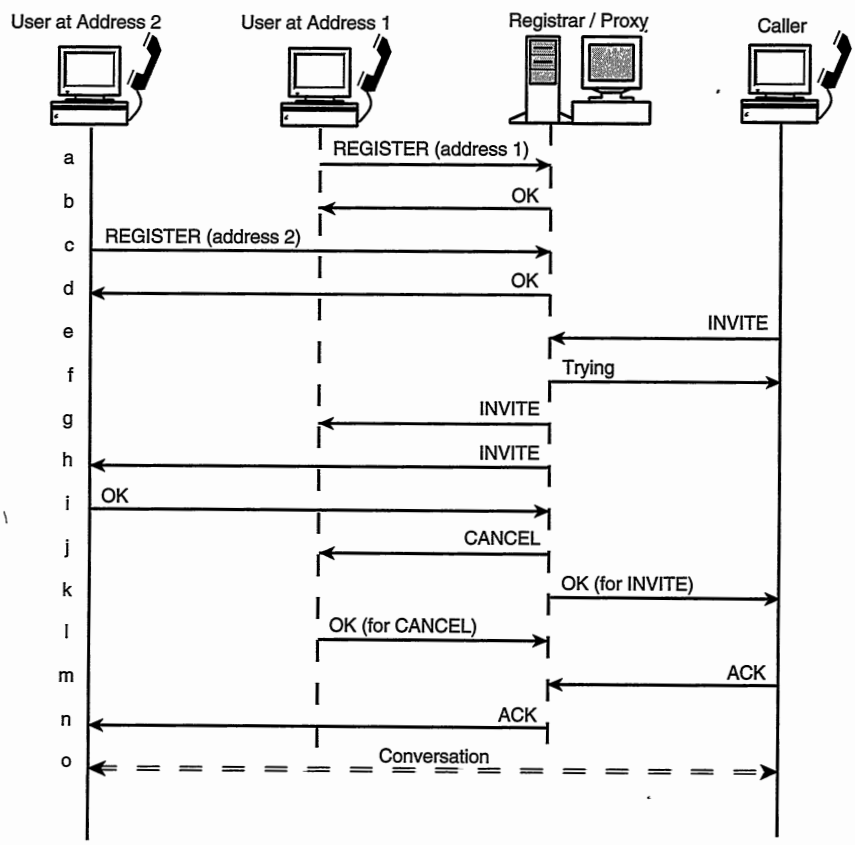


Figure 5-7
Multiple registrations enabling a one-number service