

Olivier Hersent Jean-Pierre Petit David Gurle



BEYOND VOIP PROTOCOLS

UNDERSTANDING VOICE
TECHNOLOGY AND NETWORKING
TECHNIQUES FOR IP TELEPHONY

 WILEY

Beyond VoIP Protocols

Understanding Voice Technology and Networking Techniques for IP Telephony

Beyond VoIP Protocols: Understanding Voice Technology and Networking Techniques for IP Telephony
by O. Hersent, J.P. Petit, D. Gurle

Copyright ©2005 John Wiley & Sons, Ltd. ISBN: 0-470-02362-7

IP Telephony: two-book reference set

Beyond VoIP Protocols: Understanding Voice Technology and Networking Techniques for IP Telephony is a companion reference to *IP Telephony: Deploying Voice-over-IP Protocols*. More details of this companion text may be found on the last page of this book.

Beyond VoIP Protocols

Understanding Voice Technology
and Networking Techniques for IP Telephony

Olivier Hersent

Founder and CTO

NetCentrex

Jean-Pierre Petit

Head of France Telecom R&D Human Interaction

Department (DIS/IPS)

France Telecom

and

David Gurle

WW VP Collaboration Services

Reuters



John Wiley & Sons, Ltd

Copyright © 2005

John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester,
West Sussex PO19 8SQ, England

Telephone (+44) 1243 779777

Email (for orders and customer service enquiries): cs-books@wiley.co.uk

Visit our Home Page on www.wileyeurope.com or www.wiley.com

All Rights Reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except under the terms of the Copyright, Designs and Patents Act 1988 or under the terms of a licence issued by the Copyright Licensing Agency Ltd, 90 Tottenham Court Road, London W1T 4LP, UK, without the permission in writing of the Publisher. Requests to the Publisher should be addressed to the Permissions Department, John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England, or emailed to permreq@wiley.co.uk, or faxed to (+44) 1243 770620.

This publication is designed to provide accurate and authoritative information in regard to the subject matter covered. It is sold on the understanding that the Publisher is not engaged in rendering professional services. If professional advice or other expert assistance is required, the services of a competent professional should be sought.

Other Wiley Editorial Offices

John Wiley & Sons Inc., 111 River Street, Hoboken, NJ 07030, USA

Jossey-Bass, 989 Market Street, San Francisco, CA 94103-1741, USA

Wiley-VCH Verlag GmbH, Boschstr. 12, D-69469 Weinheim, Germany

John Wiley & Sons Australia Ltd, 33 Park Road, Milton, Queensland 4064, Australia

John Wiley & Sons (Asia) Pte Ltd, 2 Clementi Loop #02-01, Jin Xing Distripark, Singapore 129809

John Wiley & Sons Canada Ltd, 22 Worcester Road, Etobicoke, Ontario, Canada M9W 1L1

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

British Library Cataloguing in Publication Data

A catalogue record for this book is available from the British Library

ISBN 0-470-02362-7

Typeset in 10/12pt Times by Laserwords Private Limited, Chennai, India

Printed and bound in Great Britain by TJ International, Padstow, Cornwall

This book is printed on acid-free paper responsibly manufactured from sustainable forestry in which at least two trees are planted for each one used for paper production.

Contents

Glossary	xi
List of Abbreviations	xv
1 Introduction	1
1.1 The rebirth of VoIP	1
1.2 Why <i>beyond VoIP protocols</i> ?	2
1.2.1 Selecting a voice coder	2
1.2.2 Providing ‘toll quality’ ... and more	2
1.2.3 Controlling IP quality of service	3
1.2.4 Dimensioning the network	4
1.2.5 Unleashing the potential of multicast	5
1.3 Scope of this book	5
1.4 Intended audience	6
1.5 Conclusion	7
1.6 References	7
2 Introduction to Speech-coding Techniques	9
2.1 A primer on digital signal processing	9
2.1.1 Introduction	9
2.1.2 Sampling and quantization	10
2.1.3 The sampling theorem	12
2.1.4 Quantization	14
2.1.5 ITU G.711 A-law or μ -law, a basic coder at 64 kbit/s	16
2.2 The basic tools of digital signal processing	20
2.2.1 Why digital technology simplifies signal processing	20
2.2.2 The Z transform and the transfer function	22
2.2.3 Linear prediction for speech-coding schemes	30

2.3 Overview of speech signals	32
2.3.1 Narrow-band and wide-band encoding of audio signals	32
2.3.2 Speech production: voiced, unvoiced, and plosive sounds	32
2.3.3 A basic LPC vocoder: DOD LPC 10	36
2.3.4 Auditory perception used for speech and audio bitrate reduction	37
2.4 Advanced voice coder algorithms	39
2.4.1 Adaptive quantizers. NICAM and ADPCM coders	39
2.4.2 Differential predictive quantization	42
2.4.3 Long-term prediction for speech signal	45
2.4.4 Vector quantization	46
2.4.5 Entropy coding	47
2.5 Waveform coders. ADPCM ITU-T G.726	47
2.5.1 Coder specification . . . from digital test sequences to C code	50
2.5.2 Embedded version of the G.726 ADPCM coder G.727	51
2.5.3 Wide-band speech coding using a waveform-type coder	52
2.6 Hybrids and analysis by synthesis (ABS) speech coders	56
2.6.1 Principle	56
2.6.2 The GSM full-rate RPE-LTP speech coder (GSM 06.10)	58
2.7 Codebook-excited linear predictive (CELP) coders	61
2.7.1 ITU-T 8-kbit/s CS-ACELP G.729	64
2.7.2 ITU-T G.723.1: dual-rate speech coder for multimedia communications transmitting at 5.3 kbit/s and 6.3 kbit/s	66
2.7.3 The low-delay CELP coding scheme: ITU-T G.728	69
2.7.4 The AMR and AMR-WB coders	71
2.8 Quality of speech coders	74
2.8.1 Speech quality assessment	75
2.8.2 ACR subjective test, mean opinion score (MOS)	77
2.8.3 Other methods of assessing speech quality	80
2.8.4 Usage of MOS	81
2.9 Conclusion on speech-coding techniques and their near future	81
2.9.1 The race for low-bitrate coders	81
2.9.2 Optimization of source encoding and channel encoding	82
2.9.3 The future	83
2.10 References	84
2.10.1 Articles	84
2.10.2 Books	85
2.11 Annexes	86
2.11.1 Main characteristics of ITU-T standardized speech coders	86
2.11.2 Main characteristics of cellular mobile standardized speech coders	88
3 Voice Quality	89
3.1 Introduction	89
3.2 Reference VoIP media path	90

3.3 Echo in a telephone network	91
3.3.1 Talker echo, listener echo	91
3.3.2 Electric echo	92
3.3.3 Acoustic echo	94
3.3.4 How to limit echo	96
3.4 Delay	101
3.4.1 Influence of the operating system	101
3.4.2 The influence of the jitter buffer policy on delay	102
3.4.3 The influence of the codec, frame grouping, and redundancy	103
3.4.4 Measuring end-to-end delay	106
3.5 Acceptability of a phone call with echo and delay	107
3.5.1 The G.131 curve	107
3.5.2 Evaluation of echo attenuation (TELR)	108
3.5.3 Interactivity	111
3.5.4 Other requirements	112
3.5.5 Example of a speech quality prediction tool: the E-model	113
3.6 Conclusion	114
3.7 Standards	115
4 Quality of Service	117
4.1 Introduction: What is QoS?	117
4.2 Describing a data stream	118
4.3 Queuing techniques for QoS	120
4.3.1 Class-based queuing	120
4.3.2 Simple fair queuing: bitwise round robin fair queuing algorithm	121
4.3.3 GPS policy in a node	122
4.4 Signaling QoS requirements	130
4.4.1 The IP TOS octet	130
4.4.2 RSVP	139
4.4.3 Scaling issues with RSVP	145
4.4.4 Scaling RSVP with a layered architecture	148
4.5 The CableLabs® PacketCable™ quality-of-service specification: DQoS	154
4.5.1 What is DQoS?	154
4.5.2 Session-per-session QoS reservation	155
4.5.3 Two-phase reservation mechanism	157
4.5.4 CMS to CMTS communications	160
4.6 Improving QoS in the best effort class	170
4.6.1 Issues with UDP traffic	171
4.6.2 Issues with TCP traffic	171
4.6.3 Using ‘intelligent’ packet discard	173
4.7 Issues with slow links	174
4.7.1 Queuing	174
4.7.2 Overhead	174

4.7.3 Overhead compression	175
4.7.4 Packet fragmentation, prioritization over serial links	177
4.8 Conclusion	179
4.9 References	181
5 Network Dimensioning	183
5.1 Simple compressed voice flow model	183
5.1.1 Model of popular voice coders	183
5.1.2 Model for N simultaneous conversations using the same coder	186
5.1.3 Loss rate and dimensioning	189
5.1.4 Packet or frame loss?	197
5.1.5 Multiple coders	198
5.2 Building a network dedicated to IP telephony	199
5.2.1 Is it necessary?	199
5.2.2 Network dimensioning	199
5.3 Merging data communications and voice communications on one common IP backbone	203
5.3.1 Prioritization of voice flows	203
5.3.2 Impact on end-to-end delay	205
5.4 Multipoint communications	206
5.4.1 Audio multipoint conferences	206
5.4.2 Multipoint videoconferencing	211
5.5 Modeling call seizures	212
5.5.1 Model of call arrivals: the Poisson process	212
5.5.2 Model of a call server	213
5.5.3 Dimensioning call servers in small networks	216
5.5.4 Dimensioning call servers in large networks	220
5.6 Conclusion	224
5.7 References	225
6 IP Multicast Routing	227
6.1 Introduction	227
6.2 When to use multicast routing	227
6.2.1 A real-time technology	227
6.2.2 Network efficiency	229
6.2.3 Resource discovery	230
6.3 The multicast framework	230
6.3.1 Multicast address, multicast group	230
6.3.2 Multicast on ethernet	232
6.3.3 Group membership protocol	233
6.4 Controlling scope in multicast applications	236
6.4.1 Scope versus initial TTL	236
6.4.2 TTL threshold	237

6.5 Building the multicast delivery tree	238
6.5.1 Flooding and spanning tree	238
6.5.2 Shared trees	238
6.5.3 Source-based trees	239
6.6 Multicast-routing protocols	241
6.6.1 Dense- and sparse-mode protocols	241
6.6.2 Other protocols	244
6.7 The mBone	249
6.7.1 An experimental network that triggered the deployment of commercial multicast networks	249
6.7.2 Routing protocols and topology	249
6.7.3 mBone applications	249
6.8 MULTICAST issues on non-broadcast media	253
6.8.1 Bridged LANs	253
6.8.2 IGMP snooping	253
6.8.3 Cisco group management protocol (CGMP)	254
6.8.4 IEEE GMRP	254
6.9 Conclusion	254
6.10 References	255
Index	257

Abbreviations

%GoB	Percent good or bad
%PoW	Percent poor or worse
3GPP	3rd Generation Partnership Project
A/D	Analog to digital
AAC	Advanced audio coding
AAL5	ATM Adaptation Layer 5
AAP	Multicast Address Allocation Protocol (draft-handley-aap-00.txt)
ABR	Available bitrate; area border router
ABS	Analysis by synthesis
ACELP	Algebraic code excited linear prediction
ACR	Absolute category rating; subjective test (MOS)
ADM	Adaptive delta modulation (modulation delta)
ADPCM	Adaptive differential pulse code modulation (MICDA)
ADSL	Asymmetric Digital Subscriber Line
ADSPEC	ADvertisement Specification (RSVP)
AEC	Acoustic echo cancelation
AMPS	Analog mobile phone standard
AMR	Adaptive multi-rate
AMR-WB	Adaptive multi-rate (wide band)
ANSI	American National Standard Institute
API	Application Programming Interface
ARIB	Association of Radio Industries and Businesses
ASBR	AS boundary router
ASVD	Analog simultaneous voice and data
ATC	Adaptive transform coding
ATM	Asynchronous Transfer Mode
BA	Behavior aggregate

BDR	Backup designated router
BER	Bit error rate
BFI	Bad frame indicator
BGMP	Border Gateway Multicast-routing Protocol.
C/D	Continuous to discrete
CAPEX	Capital Expenditure
CAT	Client accept
CBQ	Class-based queuing
CBR	Constant bitrate
CBT	Core-based tree
CC	Client close
CCR	Comparison category rating (subjective tests)
CDMA	Code division multiplex access
CDSL	Consumer Digital Subscriber Line
CELP	Code-excited linear prediction (vector quantization of excitation)
CGMP	Cisco Group Management Protocol
CID	Context identifier
CIDR	Classless Interdomain Routing
CLP	Cell loss priority
CLR	Circuit loudness rating
CM	Cable modem
CMOS	Comparison mean opinion score
CMS	Call management server
CMTS	Cable modem termination system
CNAME	Canonical Name
CNG	Comfort noise generation
COPS	Common Open Policy Service
CoS	Class of service
CPE	Customer Premises Equipment
CPU	Central processing unit
CRC	Cyclic redundancy check
CRCX	Create Connection
C RTP	Compressed Real Transport Protocol
CS-ACELP	Conjugate-structure ACELP
CT	Cordless telecommunications
CT2	CT system two
CU	Currently unused
CWTS	China Wireless Telecommunication Standard group
D/A	Digital to analog
D/C	Discrete to continuous
DAMPS	Digital AMPS (Analog Mobile Phone Service)
DCME	Digital circuit multiplication equipment
DCR	Degradation category rating test (subjective test: DMOS)

DCS	Distributed call signaling
DCT	Discrete cosine transform
DE	Discard eligible bit
DEC	Decision
DECT	Digital Enhanced cordless telephone
DMOS	Degradation mean opinion score
DNS	Domain name system
DoD	Department of Defense (US)
DQoS	Dynamic quality of service
DR	Designated router
DRQ	Delete request state
DS	Differentiated service(s)
DSCP	Differentiated services codepoint
DSL	Digital Subscriber Line
DSLAM	DSL Multiplexer
DSP	Digital signal processor (fixed point or floating point)
DSVD	Digital simultaneous voice and data
DTMF	Digital tone multi-frequency
DTX	Discontinuous transmission
DVMRP	Distance Vector Multicast-routing Protocol
EDGE	Enhanced Data Rates for GSM Evolution
EEC	Electric echo canceler
EFMA	Ethernet First Mile Association
EFR	Enhanced full rate
EGP	Exterior Gateway Protocol
EMTA	Embedded multimedia terminal adapter
ERLE	Echo return loss enhancement
ETSI	European Telecommunications Standardization Institute
FCC	Federal Communication Commission
FCFS	First come first served
FEC	Forward error correction
FER	Frame error rate
FIB	Forwarding information base, or forwarding table
FIFO	First in first out
FIR	Finite impulse response
FPLMTS	Future public land mobile telecommunication system
FR	Full rate
FS1015	Federal Standard 1015: 2.4-kbit/s LPC speech coder (NATO)
FS1016	Federal Standard 1016: 4.8-kbit/s CELP speech coder (NATO)
GARP	Generic Attribute Registration Protocol (formerly Group Address Resolution Protocol)
GC	Gate controller
GMRP	GARP Multicast Registration Protocol
GPS	Generalized processor sharing
GSM	Global System for Mobile communications

GSM-EFR	GSM-enhanced full-rate speech coder 13 kbit/s-NPAG
GSM-FR	GSM full-rate RPE-LTP 13 kbit/s
GSM-HR	GSM half-rate VSELP 5.6 kbit/s
GSTN	General switch telephone network (deregulated PSTN!)
HDLc	High-Level Data Link Control
HDVMP	Hierarchical DVMP
HTTP	Hypertext Transfer Protocol
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
IGMP	Internet Group Membership Protocol
iif	Incoming interface
IIR	Infinite impulse response
IP	Internet Protocol
IRC	Internet Relay Chat (protocol)
IS-xx	Intermediate standard xx (cf. IS-54 VSELP)
ISDN	Integrated service digital network (RNIS, NUMERIS)
ISO	International Standardization Organization
ISP	Internet service provider
ITU	International Telecommunications Union
JB	Jitter buffer
JND	Just noticeable distortion
KA	Keep alive
L2TP	Layer 2 Tunneling Protocol
LAN	Local area network
LARs	Logarithmic area ratios
LD-CELP	Low-delay CELP (ITU-T G.7728)
LFI	Link fragmentation and interleaving
LMS	Least mean squares
log-PCM	Logarithmic pulse code modulation (G.711 A-law or μ -law)
LPC	Linear predictive coding; linear prediction coefficient
LR	Loudness rating
LSA	Link-state advertisement
LSP	Line spectral pair
LTP	Long-term prediction
MA	Moving average
MAAS	Multicast address allocation server
MAC	Medium Access Control (Layer)
MARS	Multicast Address Resolution Server
MASC	Multicast Address Set Claim protocol (draft-ietf-idmr-masc-00.txt)
MBE	Multi-band excitation
MBR	Multicast border router
MBZ	Must be zero
MCML	Multi-class extension to multilink PPP
MCU	Multipoint control unit

MDCX	Modify Connection
MDHCP	Multicast Dynamic Host Configuration Protocol (address allocation extensions)
MEF	Metro Ethernet Forum
MELP	Mixed excitation linear predictor
MF	Multi-field
MGCP	Media Gateway Control Protocol
MIPS	Millions of instructions per second
ML-PPP	Multilink PPP
MLT	Modulated lapped transform
MNRU	Modulated noise reference unit
MOS	Mean opinion score (subjective tests)
MOS _{CQE}	Mean opinion score, conversational quality evaluation
MOSPF	Multicast extension to OSPF
MP-MLQ	Multipulse maximum likelihood quantization (G.723.1 at 6.3 kbit/s)
MPEG	Moving Picture Expert Group
MPLS	Multiprotocol label switching
MSB	Most Significant Bit
MSC	Mobile switching center
MSE	Mean Square Error
MTA	Multimedia terminal adapter
MTU	Maximum transmission unit
NBMA	Non-broadcast multiple access
NCS	Network-based call signaling
NICAM	Near-instantaneous companding and multiplexing
NLRI	Network-layer reachability information
NMR	Noise-to-mask ratio
NNTP	Network News Transfer Protocol
NPAG	North American PCS 1900 Action Group
NTP	Network Time Protocol
OLR	Overall loudness rating
OPEX	Operational Expenditure
OPN	Client open
OSPF	Open Short Path First (Protocol)
PAL	Phase Alternation by Line (TV standard)
PAM	Pulse amplitude modulation
PARCORS	PARTial CORrelatorS
PASTA	Poisson Arrivals See Time Averages
PBX	Private branch exchange
PCM	Pulse code modulation (MIC)
PCME	Packet circuit multiplication equipment
PCN	Personal communication network
PCS	Personal communication system
PDC	Personal digital communication

PDCP	Packet Data Convergence Protocol
PDF	Probability density function
PDP	Policy decision point
PEP	Policy enforcement point
PGPS	Packet-generalized processor sharing
PHB	Per-hop behavior
PHS	Personal handy-phone system
PID	Protocol ID
PIM	Protocol-independent multicast
PIM-SM	PIM (sparse mode)
PIM-DM	PIM (dense mode)
POP	Point of Presence
POTS	Plain Old Telephone Service
PPP	Point to Point control Protocol
PPP-ML	Multilink PPP
PSI-CELP	Pitch Synchronous Innovation CELP
PSTN	Public-switched telephone network
PT	Payload Type
QDU	Quantization distortion unit
QMF	Quadratic mirror filter
QoS	Quality of service
RADSL	Rate Adaptative DSL
RAM	Random access memory
RED	Random early detection
RELp	Residual excited linear prediction
REQ	Request
RFC	Request for comments
RIB	Routing information base
RIP	Routing Information Protocol
RLC	Running length code
RLR	Receive loudness rating
ROHC	Robust header compression
ROM	Read-only memory
RP	Rendezvous point
RPB	Reverse path broadcasting
RPE-LTP	Regular pulse-excited LPC with long-term prediction
RPF	Reverse path forwarding
RPM	Reverse path multicasting
RPT	Report state
RQNT	Request Notification
RSP	Route switch processor (Cisco)
RSRR	Routing support for resource reservation interface
RSVP	Resource ReserVation Protocol
RTCP	Real-time Transport Control Protocol
RTP	Real time Transport Protocol

RTP-Mux	Multiplex RTP
SAP	Session Announcement Protocol
SB-ADPCM	Subband ADPCM (ITU-T G.722)
SBC	Subband coding
SCFQ	Self-clocked fair queuing
SDH	Synchronous Digital Hierarchy
SDP	Session Description Protocol
SDR	Session directory
SDSL	Symmetric DSL
SECAM	Système Electronique Couleur avec Mémoire (TV Standard)
SHDSL	Single-Pair High-Speed Digital Subscriber Line
SID	Silence description
SIP	Session Initiation Protocol
SLA	Service-level agreement
SLR	Send loudness rating
SMR	Signal-to-mask ratio
SNA	System Network Architecture
SNR	Signal-to-noise ratio
SPL	Sound pressure level
SSC	Synchronize complete
SSQ	Synchronize state request
SSRC	Synchronisation Source (Identifier)
STC	Sinusoidal transform coder
TCA	Traffic-conditioning agreement
TCL	Terminal coupling loss
TCLwdt	Weighted terminal coupling loss—double talk
TCLwst	Weighted terminal coupling loss—single talk
TCP	Transport Control Protocol
TCRTP	Tunneling-multiplexed Compressed RTP
TDM	Time division multiplexing
TDMA	Time division multiplex access
TELRL	Talker echo loudness rating
TIA	Telecommunications Industry Association (USA)
TOS	Type of service
TRAU	Transcoder/Rate Adaptation Unit
TRPB	Truncated reverse path broadcasting
TSPEC	Traffic Specification
TTA	Telecommunications Technologies Association
TTC	Telecommunication Technologies Committee
TTL	Time to live
UADSL	Universal Asymmetric Digital Subscriber Line
UBR	Unspecified Bitrate
UDP	User Datagram Protocol
UDSL	Universal Digital Subscriber Line

UED/UEP	Unequal bit error detection and protection
UMTS	Universal Mobile Telecommunication System
URL	Uniform (universal) resource locator
USB	Universal Serial Bus
UTRAN	Universal Terrestrial Radio Access Network
VAD	Voice activity detector
VAT	Visual Audio Tools
VC	Virtual channel
VDSL	Very High Bitrate Digital Subscriber Line
VIC	mBone “Video Conferencing” tool
VIP	Versatile interface processor (Cisco)
VLAN	Virtual LAN
VLSI	Very large scale integration
VoIP	Voice over Internet Protocol
VQ	Vector quantization
VSELP	Vector sum-excited linear prediction
WAN	Wide-area network
WB-AMR	Wideband AMR
WFQ	Weighted fair queuing
WiFi	Wireless Fidelity
WRED	Weighted random early detection
WSNR	Weighted SNR (signal-to-noise ratio)
xDSL	Any type of DSL technology

Glossary

Aggregate *See* ‘Behavior aggregate’.

BA classifier A traffic classifier based on the DS field.

Behavior aggregate DiffServ term defined in RFC 2474 as ‘a collection of packets with the same codepoint crossing a link in a particular direction’.

Boundary link A link connecting the edge nodes of two domains (RFC 2475).

Boundary node A DS node that connects one DS domain to a node either in another DS domain or in a domain that is not DS-capable (RFC 2475).

Circuit loudness rating (CLR) Loudness loss between two electrical interfaces in a connection or circuit, each interface terminated by its nominal impedance which can be a complex value. This is 0 for a digital circuit and 0.5 for a mixed analog/digital circuit.

Class selector codepoint DiffServ term defined in RFC 2474 as ‘any of the eight codepoints in the range xxx000’ (x = 0 or 1). *See also* ‘Class selector-compliant codepoint’.

Class selector-compliant codepoint DiffServ term defined in RFC 2474 as per-hop behavior satisfying the class selector specifications as defined in RFC 2474. In short, these requirements aim at ensuring a minimal level of backward compatibility with IP precedence semantics of RFC 791 (see Chapter 4 on QoS for more details).

Codepoint Proposed name for the value of the PHB field of the DS octet, in the DiffServ framework (*see* RFC 2474 and class selector codepoint).

Controlled load service An application requesting a controlled load service for a stream of given characteristics expects the network to behave as if it was lightly loaded for that stream.

Currently unused (CU) The last two bits of the DS octet.

dBm Power level with reference to 1 mW.

dBm0 At the reference frequency (1,020 Hz), L dBm0 represents an absolute power level of L dBm measured at the transmission reference point (0-dBr point), and a level of $L + x$ dBm measured at a point having a relative level of x dBr (see G.100, annex A.4).

Differentiated service(s) (DS) The new name assigned by the IETF DiffServ group to the Ipv4 TOS field and the IPv6 traffic class field (*see* RFC 2474).

Differentiated services codepoint (DSCP) The name of the first 6 bits of the DS octet (in drafts before RFC 2474 these bits were called the PHB).

DS-compliant Behaving according to the general rules of RFC 2474 (*see* DS).

Echo Unwanted signal delayed to such a degree that it is perceived as distinct from the wanted signal.

Exterior gateway protocol (EGP) Used for unicast interdomain routing (e.g., BGP).

First come first served (FCFS) Another name for FIFO.

First in first out (FIFO) Same as FCFS.

Forwarding table For unicast routers, this is the list of the appropriate egress interface for each destination prefix. For a multicast router, this also includes the expected incoming interface (iif) and a list of outgoing interfaces (oiflist) for each destination group address (there can only be one such entry for each source for some multicast-routing protocols, like DVMRP).

Hierarchical DVMRP (HDVMP) *See* A.S. Thyagarajan and S.E. Deering. In: *Proceedings of the ACM SIGCOMM* Hierarchical distance-vector multicast routing for the MBone, pp. 60–66, October 1995.

In profile Packets part of a packet stream that were found to comply with the packet stream description (average and peak rate, maximum burst size, etc. ...).

Listener echo Echo produced by double-reflected signals and disturbing the listener.

Loudness rating (LR) As used in the G-Series Recommendations for planning, loudness rating is an objective measure of loudness loss (i.e., weighted, electro-acoustic loss between certain interfaces in the telephone network). If the circuit between the interfaces is subdivided into sections, the sum of individual section LRs is equal to the total LR. In LR contexts, the subscribers are represented from a measuring point of view by an artificial mouth and an artificial ear, respectively, both being accurately specified.

Meter A device that performs metering (RFC 2475).

Metering The process of measuring the temporal properties (e.g., rate) of a traffic stream selected by a classifier. The instantaneous state of this process may be used to affect the operation of a marker, shaper, and dropper, and/or may be used for accounting and measurement purposes (RFC 2475).

MF classifier A traffic classifier based on one or more IP header fields such as protocol number, source and destination IP addresses, port numbers, DS field value, etc. (*see also* BA classifier and the next entry).

MF classifier: multi-field classifier (MF) A functional block that is able to sort flows according to several fields of the IP packet (source address, destination address, source port, destination port, ...).

Microflow A single instance of an application-to-application flow of packets which is identified by source address, source port, destination address, destination port, and/or protocol ID (*see also* MF classifier) (RFC 2475).

Multicast RIB The routing information base, or routing table, used to calculate the ‘next hop’ toward a particular address for multicast traffic.

Network layer reachability information (NLRI) Conveyed by BGP4+, this information is used by BGMP to inject multicast routes in the interdomain-routing protocol.

oiflist A list of outgoing interfaces which is part of each forwarding table entry.

Out of profile Property of data packets within a flow which momentarily exceeds some envelope parameters of its profile (such as maximum burst size) (e.g., if the flow is regulated by a token bucket, packets arriving when there are no tokens and the backlog buffer is full are out of profile) (*see* Profile).

Overall loudness rating (OLR) Loudness loss between the speaking subscriber’s mouth and the listening subscriber’s ear via a connection.

PHB group A set of one or more PHBs that can only be meaningfully specified and implemented simultaneously, due to a common constraint applying to all PHBs in the set such as queue servicing or queue management policy (RFC 2475).

PHB Defined in RFC 2474 as ‘a description of the externally observable forwarding treatment applied at a differentiated services compliant node to a behavior aggregate’. PHB also referred to the first six bits of the DS octet in drafts before RFC 2474 (these bits are now called DSCP).

Policing The process of discarding packets by a dropper within a traffic stream in accordance with the state of a corresponding meter enforcing a traffic profile (RFC 2475).

Profile Properties of a data flow, usually defined as envelope parameters (such as maximum burst size) and mean values (such as average bitrate).

Promiscuous An interface set in promiscuous mode receives and forwards to upper layers (the device driver), all the packets it has access to, even if the physical destination address of such packets shows it is destined to another interface.

Prune A message sent by a downstream multicast router to an upstream router, meaning he is not interested in receiving multicast packets for a specific group and source. This marks a soft state in the upstream router, which usually expires after an hour or two.

Receive loudness rating (RLR) Loudness loss between an electric interface in the network and the listening subscriber's ear. Loudness loss is here defined as the weighted (dB) average of driving e.m.f. to measured sound pressure. The weighted mean value for G.111 and G.121 is 1–6 in the short term and 1–3 in the long term (from G.111).

Routing information base (RIB) The list of all routes (next hop and distance to each destination prefix) from the router.

Send loudness rating (SLR) Loudness loss between the speaking subscriber's mouth and an electric interface in the network. Loudness loss is here defined as the weighted (dB) average of driving sound pressure to measured voltage. The weighted mean value for G.111 and G.121 is 7–15 in the short term and 7–9 in the long term (from G.111).

Soft state Any state that times out after a certain delay if not refreshed.

Source router In this document only: any router directly connected to a subnetwork with a source station.

Stub domain A domain that has no transit traffic between its border routers (i.e., not used by other domains as a transit domain to destinations external to the domain).

Talker echo loudness rating (TELRL) Loudness loss of the sound of the speaker's voice reaching his ear as a delayed echo (*see* 4.2/G.122 and fig. 2/G.131).

Talker echo Echo produced by reflection near the listener's end of a connection and disturbing the talker.

Terminal coupling loss (TCL) Coupling loss between the receiving port and the sending port of a terminal due to acoustical coupling at the user interface, electrical coupling due to crosstalk in the handset cord or within the electrical circuits, and seismic coupling through the mechanical parts of the terminal. For a digital handset it is commonly in the order of 40 dB to 46 dB.

Traffic-conditioning agreement (TCA) The specification of all traffic-shaping parameters, discard policies, in/out-of-profile handling rules used for a particular service-level agreement (SLA).

Transit domain A domain that has transit traffic between its border routers (i.e., used by other domains to reach destinations external to the domain).

Weighted terminal coupling loss—double talk (TCLwdt) The weighted loss between R_{in} and S_{out} network interfaces when echo control is in normal operation and when the local user and the far-end user talk simultaneously.

Weighted terminal coupling loss—single talk (TCLwst) The weighted loss between R_{in} and S_{out} network interfaces when echo control is in normal operation and when there is no signal coming from the user.

1

Introduction

1.1 The rebirth of VoIP

In his famous book *Crossing the Chiasm*, Geoffrey A. Moore [B1] explains why so many innovative companies fail to turn their early successes into solid market positions and recurrent revenues. In an immature market, early adopters are eager to test new products and services, and they are willing to accept minor imperfections. When the market starts to mature, however, these ‘beta stage’ products do not sell as well, and many executives are led to believe that they do not innovate enough: instead of completing the product, they push even more ‘beta products’ to the market. This is the wrong decision: the key to massive product adoption in a mature market is *the whole product* (i.e., a product acceptable not only to technology enthusiasts and visionaries—the early market, but also to pragmatists, conservatives and even skeptics—the mainstream market). The whole product is not a marginal enhancement compared with the first prototypes, very often it requires as much effort and a lot of time to carefully analyze, understand and leverage the feedback of early users.

In many ways, the whole VoIP industry has just ‘crossed the chiasm’.

In 1999–2000, VoIP was one of the most successful buzzwords of the telecom bubble era. Every start-up company had a ‘new service’ or a ‘killer application’ that would change the landscape of the telecommunication industry for ever. The technology was evolving so fast that protocols introduced in 1998 were called ‘obsolete’ a year later ... in fact, every manufacturer claimed that his technology was so much better than that of competitors that interoperability was impossible.

From 2001 to 2003, VoIP faced a very tough reality check by pragmatist and conservative service providers. Many enthusiast ‘next-generation’ service providers that had spent billions on immature products realized that this expense did not find a mass market for them; in fact, the cost of sales of many of the ‘killer services’ exceeded any foreseeable

revenue. In a depressing climate with a start-up failing every month and even large service providers filing for the now famous ‘chapter 11’, the buzz for VoIP quickly disappeared.

Today the VoIP chiasm is behind us. Quite a few manufacturers and service providers survived, and are ready to participate in one of the most massive and disruptive technology changes ever faced by the telecom industry.

1.2 Why beyond VoIP protocols?

The new multimedia services will need to be much more than mere technology demonstrators. In order to build a demonstrator, engineers need only focus on the functional aspects: select a protocol, make sure it has the right service primitives, and combine these primitives into the desired functionality. The companion reference to this book, *IP Telephony: Deploying Voice-over-IP Protocols*, focuses on such functional aspects, presenting a high-level overview of packet media transport technologies, details on all three major VoIP protocols (H.323, SIP, and MGCP), and specific strategies to design services in the context of public networks where endpoints cannot be trusted and can be behind firewalls.

As its title implies, *Beyond VoIP Protocols: Understanding Voice Technology and Networking Techniques for IP Telephony* provides a broad overview of all the additional issues that need to be solved in order to deploy a multimedia service.

1.2.1 Selecting a voice coder

In the lab, almost any voice coder can be selected: there is plenty of bandwidth and hardly any packet loss. In a real network, however, even with the massive deployment of DSL, there is often a need to carefully select a voice coder that fits in the available bandwidth and provides the desired level of service. This is not an obvious choice, and it is necessary to have a deeper understanding on the internals of each voice coder in order to understand how each candidate coder may react to packet loss, for instance. During the bubble, the VoIP industry has generated many ‘magic coders’ which are supposed to outperform any other codec: a deeper understanding of codec technology helps separating true innovations from naive tricks. Finally, as the new generation of multi-rate adaptive coders appears for use by the 3G networks, it is important to keep in mind the fundamental differences between wireless and wired networks in order to evaluate which of the many innovations of AMR or AMR-WB coders may lead to significant improvements for Internet-based multimedia applications.

Chapter 2 “Introduction to Speech Coding Techniques” provides the necessary background to efficiently evaluate the candidate coders for a network and make the best compromise.

1.2.2 Providing ‘toll quality’ . . . and more

The first service providers who massively adopted VoIP were prepaid card vendors. Unfortunately, many of these service providers bet on the fact that most of their potential clients

would focus only on price and would have no means of complaining or asking for a refund if the voice quality was not acceptable. VoIP also had a lot of success among international transit carriers and arbitrage houses, and here as well voice quality is often not a prime concern. If you travel abroad and try to reach your voicemail, but cannot dial your DTMF access code correctly, chances are that your current service provider uses a VoIP network for international calls and never checked whether DTMF tones could get through.

Such bad experiences unfortunately backfired and created a perception among first-tier service providers that VoIP did not work. Most first-tier service providers conducted experiments between 2000 and 2002 in order to assess the elasticity of user voice quality acceptance levels relative to price. These studies aimed at designing tiered voice offers, with cheap, low-quality calls and more expensive toll-quality calls. To the surprise of everyone, these studies showed that there was only a willingness to pay more than toll quality for very high-quality calls (wideband coders); on the other hand, if the toll quality was perceived to be significantly lower than toll quality, there was no willingness to pay at all.

The consequence is that all post-bubble VoIP networks will need to provide a voice quality guaranteed to be comparable with toll quality, or better. Beyond the intrinsic quality of the voice coder, detailed in Chapter 2, Chapter 3 ‘Voice Quality’ discusses in detail how to control the most important parameters influencing end-users’ perception of voice quality: delay and echo.

1.2.3 Controlling IP quality of service

Peer-to-peer applications killed the idea that over-provisioning could solve quality-of-service problems on the Internet. Now that almost every DSL user is attempting to download his wish list of 10 or more 700-MB ‘DivX’ videos, it can be taken for granted, on the contrary, that most DSL links are permanently congested. The situation is likely to become even worse as some peer-to-peer telephony applications begin to use very aggressive redundancy techniques in order to get an unfair share of the best effort bandwidth. If everyone keeps throwing more packets to the best effort Internet, it will soon become very difficult to use this class of service for many applications that require a minimum level of responsiveness. In fact, the ‘best effort’ class of service is no longer usable for real-time applications, like telephony or videoconferencing, which cannot recover from packet loss.

Chapter 4 ‘Quality of Service’ discusses these issues from multiple points of view. At a low level, it explains the PGPS theory that makes it possible to provide differentiated levels of quality of service over all packet networks and helps understand the old ‘IP against ATM’ battles. It then presents the ‘DiffServ’ framework that today provides a simple, yet effective, way of marking IP packets with a desired quality-of-service level, and eventually downgrades to lower quality-of-service levels packets that are outside the agreed service-level agreement. There is a lot that can be done with DiffServ, but it must be used carefully, and the chapter also gives some guidelines on which types of traffic can be aggregated within a given service level and how to improve ‘fairness’ in a given class of service (even the best effort class).

Nevertheless, Diffserv does have some limitations, and it is likely that in the long run service providers will need to implement more dynamic ways of managing the service level of packet streams generated by end-users. The IntServ framework was initially presented as a direct application of the PGPS theory, and as such it is very powerful but also difficult to scale. Chapter 4 also discusses how a mix of IntServ and Diffserv could provide a good compromise for the future, and describes the current DQoS framework for cable networks, which is probably very similar to the techniques that will be used in the future on all public IP networks.

1.2.4 Dimensioning the network

From a dimensioning point of view, packet multimedia networks based on IP are unique compared with traditional telephony networks based on time division multiplexing (TDM), but also compared with other packet-based networks (e.g., ATM). The difference with TDM is obvious: bandwidth is no longer needed during silence periods, which makes it possible, when aggregating multiple streams, to save up to 50% of the bandwidth that would have been necessary if all voice channels were transmitting continuously. Unfortunately, this gain must be mitigated by the fact that the technique used by virtually all IP applications to transport media streams, RTP, is very inefficient in itself. In most cases, discontinuous transmission gains will be compensated by the overheads of the IP transport, and in the end the average capacity required by an IP transport with simple voice coders is comparable with what would have been required on TDM. It is possible to achieve further gains by using low-bitrate coders, but this has an influence on end-to-end delay and voice quality, or in specific circumstances by optimizing the IP/UDP/RTP transport layers.

The difference from ATM networks is less obvious. As there was no ‘best effort’ traffic on ATM, networks required a very strict dimensioning in order to minimize the chances of rejecting a new connection if the admission control failed. As we have seen above, the explosive growth of best effort traffic makes it impossible to use this class for interactive streams; but once a separate class of service is created for voice or video-conferencing, the fact that most of the network capacity is used for best effort traffic makes it a lot easier to dimension the real-time class of service. Within reasonable limits, the real-time class can ‘eat’ the capacity used by best effort users who have no service-level agreement and, in theory, never complain. In fact, hybrid voice and data networks are not only simpler to dimension, but will provide lower end-to-end transmission delays for voice streams, compared with a pure voice network. Chapter 5 ‘Network Dimensioning’ presents the traditional techniques that must be used to dimension a network (e.g., the Erlang laws that evaluate the number of simultaneous active calls for a given number of users, or the Poisson laws that can be used to evaluate the processing capacity of softswitches required to handle VoIP signaling). Chapter 5 also discusses the characteristics of VoIP streams compared with TDM voice channels and explains how to extrapolate the results of traditional network dimensioning theory, based on constant bitrate streams, to ‘on-off’ streams typical of VoIP when used in combination with voice activity detection.

1.2.5 Unleashing the potential of multicast

The IP network was initially designed and optimized to provide robust point-to-point connections, using ‘unicast’ packets. Unfortunately, not all applications work well with point-to-point connections: all broadcast applications do not scale well if they need to duplicate the information stream for every listener. Today most commercial IP networks are still ‘unicast’-only, and for this reason when you connect to TV station websites, you get only a small, low-quality image even if you have a DSL connection at home. The reason is that the TV station still uses unicast and therefore needs to send a copy of the TV channel to everyone. Today MPEG2 requires about 2–3 Mbit/s for TV-quality transmission over IP: sending a single channel to 1 million ‘IP-TV’ sets would require no less than 3 Tbit/s at the TV station!

Today many service providers are enhancing their IP networks to provide support for ‘multicast’. This supports their ‘triple-play’ strategy in the residential market, which requires high-quality TV transmission over IP and enables many services in the corporate market (e.g. large videoconferences—the equivalent of webinars with a video stream).

Chapter 6 ‘Multicast’ explains how the technology can turn an IP backbone into a optimized broadcast medium, and discusses the numerous issues that have delayed the introduction of multicast on commercial networks and continue to limit the scope of feasible applications today. As multicast does have a specific behavior related to network sizing, Chapter 5 also presents the impact of various multicast distribution tree configurations on the required capacity of each link.

For the engineering department of service providers, Chapters 5 and 6 together will provide much of the material required when designing a ‘triple-play’ offer combining voice, IP-TV, and Internet access over DSL.

1.3 Scope of this book

Beyond VoIP Protocols is a companion reference to *IP Telephony: Deploying Voice-over-IP Protocols* (for more details see the last page of this book). Both books have been written with the goal of supporting those involved in the design and deployment of multimedia VoIP projects, and provide invaluable references for most of the required technology.

Our idea is that during the execution of a project, an engineer frequently needs to have a fairly accurate view of ‘the complete picture’ in order to avoid fundamental mistakes or misunderstandings, and he/she will usually only need a complete, exhaustive reference for a small fraction of the overall technology involved. In our professional lives, we all have spent an enormous amount of time compiling this ‘complete picture’, and we hope that these two books will significantly reduce the time needed to assimilate the essential information and avoid many of those errors that can be made through only being aware of half the picture. That said, our intention was to provide a complete overview and not every detail on each subject. For instance, when introducing the audio-coding techniques in Chapter 2, we do provide some background on the ‘Z transform’ in order to give a feel for the power of this technique and become capable of understanding the

codec design diagrams. This background obviously does not replace a complete book on the Z transform if your intention is to specialize in codec design and audio-processing algorithms. Similarly, if you do design an H.323, SIP, or MGCP device, you will need to actually read the parts of the standards that relate to your specific application, but *Beyond VoIP Protocols* will help you skip through 90% of the text, as you will already have enough background.

We structured our reference texts as two books, because most projects involve two phases:

In phase 1, *IP Telephony: Deploying Voice-over-IP Protocols* can help you combine the various protocols for the target service, and complements the standards by discussing the most common issues that may result from incomplete protocol implementations, or architectures optimized for private networks which fail in a public environment. This first book focuses on the functional aspects.

In phase 2, you will need to know how many users the application will serve, you will need to select or configure an IP distribution network, you will need to build a business model for the service. In so doing, you will have to answer questions like: ‘How many gateway ports do I need for 150,000 users?’ ‘Is 128 kbit/s upstream sufficient to support VoIP for my DSL users?’ ‘Can this DSLAM support 5,000 users with 10% placing a phone call and 20% watching IP-TV?’ ‘Do I need an IP DSLAM, or will two ATM VCs per DSL user be sufficient to provide support for VoIP and video?’ ‘What is the cost per user of this softswitch, which is sold per simultaneous call?’

Beyond VoIP Protocols will not directly answer 100% of these questions, because every deployment is a unique challenge, but it will provide some useful tools that can be used as building blocks to help formulate a complete deployment strategy. For instance, every question that relates a number of users to a number of ports, calls, or aggregate bandwidth ultimately boils down to an Erlang calculation. The answer to most quality of service-related questions is a properly designed set of differentiated service levels which must obey certain constraints (detailed in Chapter 4), such as similarity of aggregated streams, proper support by the layer 2 transport level, etc.

1.4 Intended audience

The intended audience for *Beyond VoIP Protocols* is:

- Network planning teams, in charge of buying transport capacity, who need to guarantee acceptable end-to-end transmission delays.
- RFI/RFP technical teams who want to evaluate IP access devices (e.g., DSLAMs or BASs). The proper support for per-stream quality of service becomes fundamental in such equipment.
- Technical support teams for marketing departments who want to evaluate the cost side of ‘triple-play’ business models, which depend on the required bandwidth at the access level, as well as the sizing of control softswitches and gateways.

- National telecom regulators who want to evaluate the impact of IP wholesale prices on the viability of competitive VoIP service providers or want to assess the credibility of the threat of ‘virtual service providers’ (e.g., Vonage in the US), using an existing DSL local loop to provide telephony services without a license. Regarding numbering issues, the book provides strong arguments in favor of dedicating specific number ranges to VoIP (the solution adopted in Japan and many other countries), in order to avoid IP to TDM to IP connections, which obviously exceed acceptable delay constraints and introduce unwanted codec tandeming.
- Telecom students who want to understand how to use classic telecom tools in the context of VoIP. The book also provides an overview of many of the active research areas related to multimedia over IP and can help select a ‘hot’ topic for a thesis.

All the chapters in *Beyond VoIP Protocols* are relatively independent. They have been ordered from the most voice-centric to the most network-centric, but the reader can skip the chapters they are not interested in. For instance, if you are comfortable with considering voice coders as ‘black boxes’, you may skip Chapter 2 (or read the higher level description in Chapter 1 of *IP Telephony: Deploying Voice-over-IP Protocols*).

1.5 Conclusion

As with every technical book, despite careful proof-reading, there may be errors, typing mistakes, or you may find that some important new development are missing. We welcome your feedback, which can be sent by email to book@netcentrex.net. As technology is constantly evolving, we welcome all your suggestions for inclusion of new topics in future editions of the book.

Any updated material as well as a log of typos (should these be any) will be maintained at www.netcentrex.net/book. We will also publish there additional documents posted by our readers if we feel they are of interest to the intended audience.

We hope that you will find *Beyond VoIP Protocols* useful and that you will be convinced that, with appropriate design and planning, the technology is now mature enough to support massive deployments.

1.6 References

- [B1] G.A. Moore. *Crossing the Chiasm*, HARPER COLLINS, 1991 (ISBN 0-06-662 001-3).

2

Introduction to Speech-coding Techniques

2.1 A primer on digital signal processing

2.1.1 Introduction

At the beginning of the 20th century, all devices performing some form of signal processing (recording, playback, voice or video transmission) were still using analogue technology (i.e., media information was represented as a continuously variable physical signal). It could be the depth of a groove on a disk, the current flowing through a variable resistance microphone, or the voltage between the wires of a transmission line. In the 1960s, the PCM (pulse code modulation) of audio began to be used in telecom switching equipment. Since 1980 the spectacular performance advances of computers and processors led to an ever-increasing use of digital signal processing.

Today speech signals sampled at 8 kHz can be correctly encoded and transmitted with an average of 1 bit per sample (8 kbit/s) and generic audio signals with 2 bits per sample. Speech coders leverage the redundancies within the speech signal and the properties (the weaknesses) of human ears to reduce the bitrate. Speech coding can be very efficient because speech signals have an underlying vocal tract production model; this is not the case for most audio signals, such as music.

This chapter will first explain in more detail what a 'digital' signal is and how it can be obtained from a fundamentally analogue physical input that is a continuously variable function of time. We will introduce the concepts of sampling, quantization, and transmitted bandwidth. These concepts will be used to understand the basic speech-coding schemes used today for telephony networks: the ITU-T A-law and μ -law encodings at 64 kbits per second (G.711).

At this point the reader may wonder why there is such a rush toward fully digital signal processing. There are multiple reasons, but the key argument is that all the signal transformations that previously required discrete components (such as bandpass filters, delay lines, etc.) can now be replaced by pure mathematical algorithms applied to the digitized signal. With the power of today's processors, this results in a spectacular gain in the size of digital-processing equipment and the range of operations that can be applied to a given signal (e.g., acoustic echo cancellation really becomes possible only with digital processing). In order to understand the power of fully digital signal processing, we will introduce the 'Z transform', the fundamental tool behind most signal-processing algorithms.

We will then introduce the key algorithms used by most voice coders:

- Adaptive quantizers.
- Differential (and predictive ...) quantization.
- Linear prediction of signal.
- Long-term prediction for speech signal.
- Vector quantization.
- Entropy coding.

There are two major classes of voice coders, which use the fundamental speech analysis tools in different ways:

- Waveform coders.
- Analysis by synthesis voice coders.

After describing the generic implementation of each category, the detailed properties of the most well known standardized voice coders will be presented.

We will conclude this chapter by a presentation of speech quality assessment methods.

2.1.2 Sampling and quantization

Analog-to-digital conversion is the process used to represent an infinite precision quantity, originally in a time-varying analog form (such as an electrical signal produced by a microphone), by a finite set of numbers at a fixed sample rate, each sample representing the state of the original quantity at a specific instant. Analog-to-digital conversion is mandatory in order to allow computer-based signal analysis, since computers can only process numbers.

Analog-to-digital conversion is characterized by:

- The rate of **sampling** (i.e., how often the continuously variable quantity is measured).
- The **quantization** method (i.e., the number of discrete values that are used to express the measurement (typically a certain number of bits), and how these values are distributed

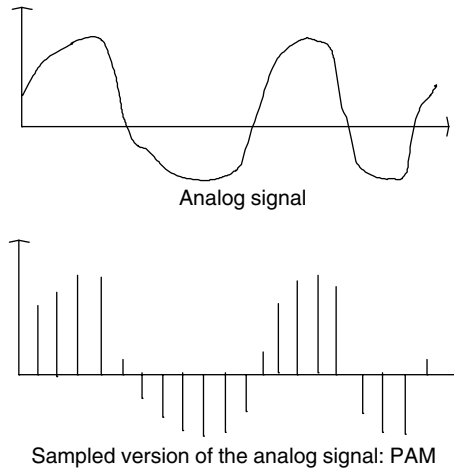


Figure 2.1 Pulse amplitude modulation.

(linearly on the measurement scale, or with certain portions of the measurement scale using a more precise scale than others)).

Mathematically, the sampling process can be defined as the result of the multiplication of an infinite periodical pulse train of amplitude 1 (with a period corresponding to the sampling period), by the original **continuous-time** signal to be sampled. This leads to the **PAM (pulse amplitude modulation) discrete time** representation of the signal (Figure 2.1).

From the PAM signal, it is possible to regenerate a continuous time signal. This is required each time the result of the signal-processing algorithm needs to be played back. For instance, a simple **discrete-to-continuous (D/C)** converter could generate linear ramps linking each pulse value, then filter out the high frequencies generated by the discontinuities.

Analog-to-digital conversion loses some information contained in the original signal, which can never be recovered (this is obvious in Figure 2.2). It is very important to choose the sample rate and the quantization scale appropriately, as this directly influences the quality of the output of the signal-processing algorithm [A2, B1, B2].

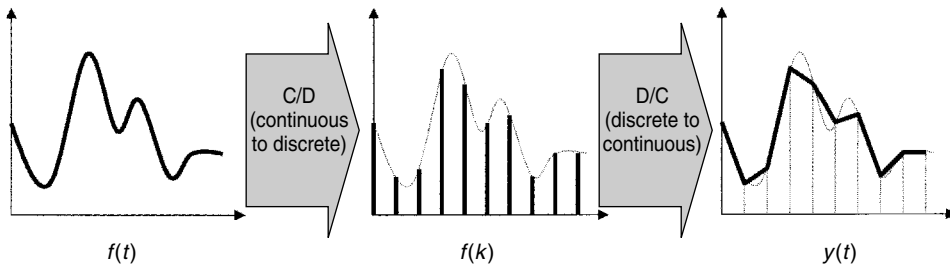


Figure 2.2 Reconstruction of a continuous signal from a discrete signal.

2.1.3 The sampling theorem

The **sampling theorem** states that in order to process a continuous time signal with frequency components comprised between 0 and F_{\max} , the sampling rate should be at least $2 * F_{\max}$. Intuitively, it can be understood by looking at the quantization of a pure sinusoid. In Figure 2.3 the original signal of frequency 1.1 is sampled at frequency 1. The resulting PAM signal is identical to the sampling result at frequency 1 of a signal at frequency 0.1. This is the **aliasing** phenomenon.

In fact if T is the sampling period (**radial frequency** $\Omega_r = 2\pi/T$):

- All sinusoids of frequency $\omega_r + m\Omega_r$ will have the same PAM representation as the sinusoid of frequency ω_r , since $\cos((\omega_r + m(2\pi/T))t)$ is sampled as

$$\cos((\omega_r + m(2\pi/T)kT) = \cos((\omega_r kT + mk2\pi) = \cos(\omega_r kT)$$

- The sinusoids of frequencies $\Omega_r/2 + \omega_r$ and $\Omega_r/2 - \omega_r$ have the same PAM representation because

$$\begin{aligned} \cos((\Omega_r/2 \pm \omega_r)kT) &= \cos(\pi k \pm \omega_r kT) = \cos(\pi k) \cos(\omega_r kT) \mp \sin(\pi k) \sin(\omega_r kT) \\ &= \cos(\pi k) \cos(\omega_r kT) \end{aligned}$$

This is illustrated on Figure 2.4.

The conclusion is that there is one-to-one mapping between a sinusoid and its PAM representation sampled at frequency Ω only if sinusoids are restricted to the $[0, \Omega/2]$ range. This also applies to any signal composed of mixed sinusoids: the signal should not

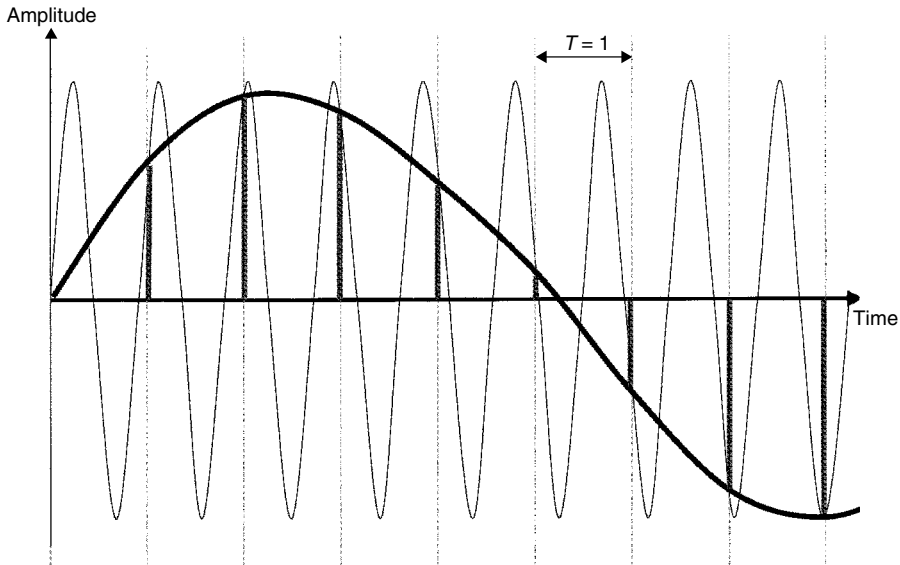


Figure 2.3 Aliasing of frequency 1.1, sampled at frequency 1, wrapped into frequency 0.1.

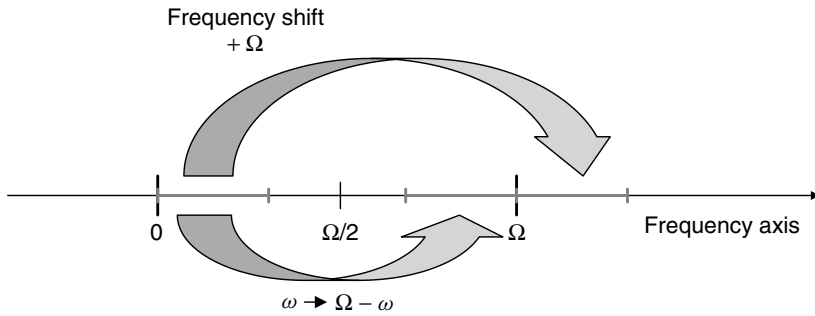


Figure 2.4 The different types of frequency aliasing.

have any frequency component outside the $[0, \Omega/2]$ range. This is known as the **Nyquist theorem**, and $\Omega = 2\omega$ is called the **Nyquist rate** (the minimal required sampling rate for a signal with frequency components in the $[0, \omega]$ range).

The Nyquist (or Shannon) theorem also proves that it is possible to exactly recover the original continuous signal from the PAM representation, if the sampling rate is at or above the Nyquist rate. It can be shown that the frequency spectrum (Fourier transform) of a PAM signal with sampling frequency F_s is similar to the frequency spectrum of the original signal, repeated periodically with a period of F_s and with a scaling factor.

From Figure 2.5 it appears that the original signal spectrum can be recovered by applying an ideal low-pass filter with a cutting frequency of $F_s/2$ to the PAM signal. The unique condition to correctly recover the original analog spectrum is that there is no frequency wrapping in the infinite PAM spectrum. The only way to achieve this

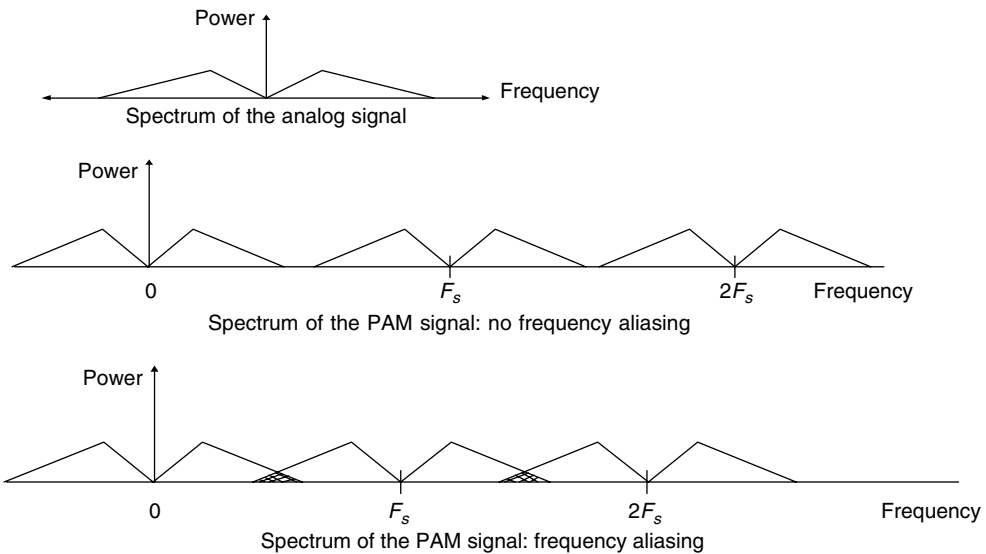


Figure 2.5 The Nyquist rate and frequency wrapping.

is for the bandwidth of the original analog signal to be strictly limited to the frequency band $[0, F_s/2]$. Figure 2.5 shows an ideal situation and a frequency-wrapping situation; in the case of frequency wrapping, the recovered signal is spoiled by frequency aliasing.

The spectrum of real physical signals (such as the electrical signal generated by a microphone) do not have a well-defined frequency limit. Therefore, before the sampling process, it is necessary to cut off any frequency component beyond the Nyquist frequency by using an ‘anti-aliasing’ analog filter. In order to avoid this discrete component (it is not obvious to approximate an ideal low-pass filter with analog technology), modern oversampled noise-shaping analog-to-digital converters (also called sigma delta coders) use a very high-sampling frequency (the input signal is supposed not to have any very high-frequency component) but internally apply digital decimation (subsampling) filters which perform the anti-aliasing task before the sampling rate is reduced.

In the digital-to-analog chain, the reconstruction filter is responsible for transforming the discrete digital signal into a continuous time signal.

The value of the sampling frequency not only determines the transmitted signal bandwidth but also impacts the amount of information to be transmitted: for instance, wide-band, high-quality audio signals must be sampled at high frequencies, but this generates far more information than the regular 8,000-Hz sampling frequency used in the telephone network.

2.1.4 Quantization

With the sampling process discussed in the previous paragraph, we are not yet in the digital world. The PAM signal is essentially an analog signal because the amplitude of each pulse is still a continuous value that we have not attempted to measure with a number. In fact we have lost only part of the information so far (the part of the sampled signal above one-half of the sampling frequency). We will lose even more information when we measure the amplitude of each pulse.

Let’s imagine that a folding rule is used to measure the amplitude of the PAM signal. Depending of the graduation or precision of the scale, the number that represents the PAM signal can be more or less precise . . . but it will never be exact. The PAM signal can be represented by the digital signal with pulses corresponding to the measured values, plus a PAM signal with pulses representing the errors of the quantization process. The signal encoding in which each analog sample of the PAM signal is encoded in a binary code word is called a **PCM (pulse code modulation)** representation of the signal. The analog-to-digital conversion is called quantization.

With a more precise quantization process, we minimize the amplitude of the noise, but we cannot avoid introducing some noise in the quantization process (**quantization noise**). Once quantization noise is introduced in a speech or audio transmission chain, there is no chance to improve the quality by any means. This has important consequences: for instance, it is impossible to design a digital echo canceler working on a PCM signal with a signal-to-echo ratio above the PCM signal’s signal-to-noise ratio.

Therefore there are two sources of loss of information when preparing a signal for digital processing:

- The loss of high-frequency components.
- Quantization noise.

The two must be properly balanced in any analog-to-digital (A/D) converter as both influence the volume of information that is generated: it would be meaningless to encode with a 24-bit accuracy a speech signal which is intentionally frequency-limited to the 300–3,400-Hz band; the limitation in frequency is much more perceptible than the ‘gain’ in precision brought by the 24 bits of the A/D chain.

If uniform quantization is applied (‘uniform’ means that the scale of our ‘folding rule’ is linear) the power of the quantization noise can be easily derived. All the step sizes of the quantizer have the same width D ; therefore, the error amplitude spans between $-D/2$ to $+D/2$ and it can be shown [B1] that the power of this error is:

$$E^2 = \frac{D^2}{12}$$

For a uniform quantizer using N bits (N is generally a power of 2) the maximum **signal-to-noise ratio (SNR)** achievable in decibels is given by:

$$SNR(dB) = 6.02N - 1.73$$

For example, a CD player uses a 16-bit linear quantizer and the maximum achievable SNR is 94.6 dB. This impressive figure hides some problems: the maximum value is obtained for a signal having the maximum amplitude (e.g., a sinusoid going from $-32,768$ to $+32,767$). In fact, the SNR is directly proportional to the power of the signal: the curve representing the SNR against the input power of the signal is a straight line. If the power of the input signal is reduced by 10 dB, the SNR is also reduced by 10 dB. For very low-power sequences of music, some experts (golden ears) can be disturbed by the granularity of the sound reproduced by a CD player and prefer the sound of an old vinyl disk.

Because of this problem, the telecom industry generally uses quantizers with a constant SNR ratio regardless of the power of the input signal. This requires nonlinear quantizers (Figure 2.6).

As previously stated, the sampling frequency and the number of bits used in the quantization process both impact the quality of the digitized signal and the resulting information rate: some compromises need to be made. Table 2.1 [A2] gives an overview of the most common set of parameters for transmitting speech and audio signals (assuming a linear quantizer).

Even a relatively low-quality telephone conversation results in a bitrate around 100 kbit/s after A/D conversion. This explains why so much work has been done to reduce this bitrate while preserving the original quality of the digitized signal. Even the well-known A-law or μ -law PCM G.711 coding schemes at 64 kbit/s, used worldwide in all digital-switching machines and in many digital transmission systems, can be viewed as a speech coder.

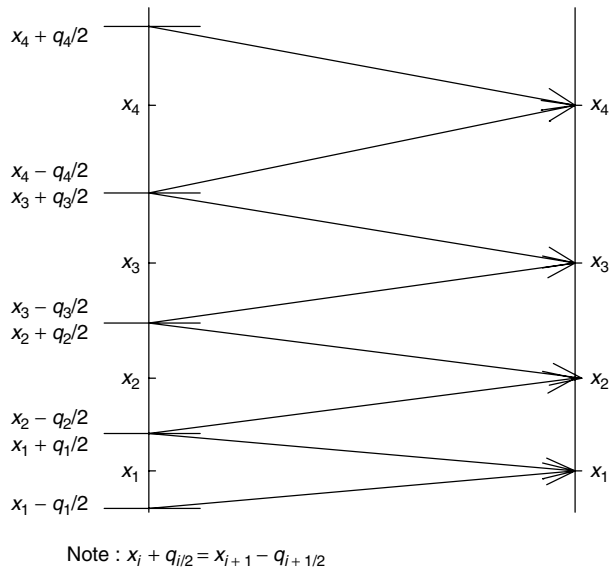


Figure 2.6 Example of a nonlinear quantizer. Any value belonging to $[x_i - q_i/2, x_i + q_i/2]$ is quantized and converted in x_i . The noise value spans in $[-q_i/2, +q_i/2]$.

Table 2.1 Common settings for analog-to-digital conversion of audio signals

Type	Transmitted bandwidth (Hz)	Sampling frequency (kHz)	Number of bits in A/D and D/A converters	Bitrate in kbit/s	Main applications
Telephone speech	300–3,400	8	12 or 13	96 or 104	PSTN, ISDN networks, digital cellular
Wide-band speech (and audio)	50–7,000	16	14 or 15	224 or 240	Video and audio conferencing, FM radio
High-quality speech and audio	30–15,000	32	16	512	Digital sound for analog TV (NICAM)
	20–20,000	44.1	16	706	audio CD player
	10–22,000	48	Up to 24	1,152	Professional audio

2.1.5 ITU G.711 A-law or μ -law, a basic coder at 64 kbit/s

A linear quantizer is not usually optimal. It can be mathematically demonstrated that if the **probability density function (PDF)** of the input signal is known, an optimal

quantizer [B1, B2] can be computed which leads to a maximal SNR for this signal. The resulting quantizer is not linear for most signals. Of course, the main issue is to know the PDF of a given signal; for random speech and audio signal, this is a very difficult task as it may depend on multiple factors (language, speaker, loudness, etc.).

Another approach to finding an optimal quantizer is to look for a quantizer scale which yields an SNR independent of the level of the signal. It can be shown that this requires a logarithmic scale: the step size of the quantizer is doubled each time the input level is doubled. This process is called **companding** (*compress and expanding*): compared with the PAM signal, the digital PCM representation of the signal is ‘compressed’ by the logarithmic scale, and it is necessary to expand each PCM sample to obtain the PAM signal back (with quantization noise).

The ITU telephony experts also noted that the 12–13-bit precision of the linear quantizers discussed above were only useful for very weak signals, and such a precision was not necessary at higher levels. Therefore, a step size equivalent to the step size of a 12-bit linear quantizer would be needed only at the beginning of the logarithmic scale.

The ITU G.711 logarithmic voice coder uses the concept of companding, with a quantization scale for weak signals equivalent to a 12-bit linear scale. Two scales were defined, the A-law (used in Europe and over all international links) and the μ -law (used in North America and Japan). The two laws rely on the same approximation of a logarithmic curve: using segments with a slope increasing by a factor of 2, but the exact length of segments and slopes differ between the A-law and the μ -law. This results in subtle differences between the A-law and the μ -law: the A-law provides a greater dynamic range than the μ -law, but the μ -law provides a slightly better SNR than the A-law for low-level signals (in practice, the least significant bit is often stolen for signaling purposes in μ -law countries, which degrades the theoretical SNR).

G.711 processes a digital, linear, quantized signal (generally, A/D converters are linear) on 12 bits (sign + amplitude; very often A/D outputs are 2’s complements that require to be converted to the sign + amplitude format). From each 12-bit sample, the G.711 converter will output a 8-bit code represented in Figure 2.7:

In Figure 2.7, S is the sign bit, E2E1E0 is the exponent value, and M3M2M1M0 is the mantissa value. A-law or μ -law encoding can be viewed as a floating point representation of the speech samples.

The digital-encoding procedure of the G.711 A-law is represented in Table 2.2 [A1]. The X, Y, Z, T values are come from the code and are transmitted directly as M3, M2, M1, M0 (the mantissa). Note that the dashed area corresponds to quantization noise which is clearly proportional to the input level (constant SNR ratio).

Figure 2.8 represents the seven-segment A-law characteristic (note that, even though we have eight segments approximating the log curve, segments 0 and 1 use the same slope).

On the receiving side, the 8-bit A-law code is expanded into 13 bits (sign + amplitude), representing the linear quantization value. In order to minimize decoded quantization noise, an extra bit is set to ‘1’ for the first two segments (see Table 2.3)



Figure 2.7 The G.711 8-bit code.

Table 2.2 Amplitude encoding in G.711

Segment number (sign bit omitted)			Amplitude coded with 11 bits (sign + amplitude, sign bit omitted)										
			B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
0	0	0	0	0	0	0	0	0	0	X	Y	Z	T
0	0	1	0	0	0	0	0	0	1	X	Y	Z	T
0	1	0	0	0	0	0	0	1	X	Y	Z	T	N
0	1	1	0	0	0	0	1	X	Y	Z	T	N	N
1	0	0	0	0	0	1	X	Y	Z	T	N	N	N
1	0	1	0	0	1	X	Y	Z	T	N	N	N	N
1	1	0	0	1	X	Y	Z	T	N	N	N	N	N
1	1	1	1	X	Y	Z	T	N	N	N	N	N	N

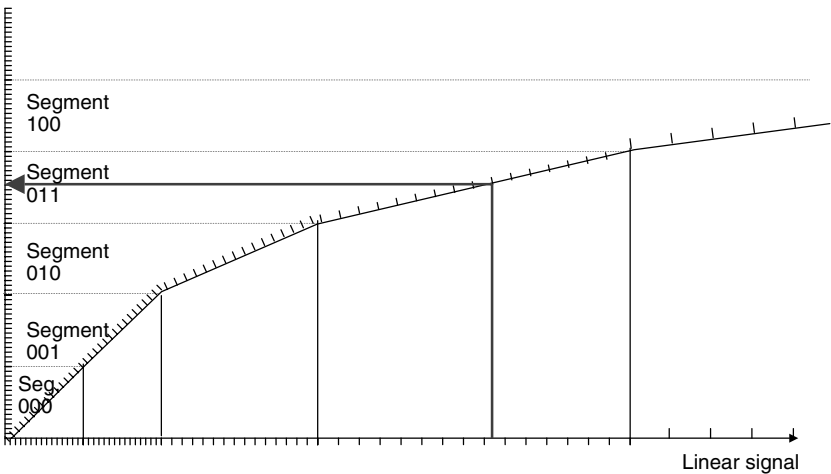


Figure 2.8 Logarithmic approximation used by G.711 A-law.

Table 2.3 Decoding table for G.711 8-bit codes

Exponent	Sign bit	Decoded amplitude using $\frac{1}{2}$ quantization steps (12 bits)											
		B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0	B - 1
0	S	0	0	0	0	0	0	0	M3	M2	M1	M0	1
1	S	0	0	0	0	0	0	1	M3	M2	M1	M0	1
2	S	0	0	0	0	0	1	M3	M2	M1	M0	1	0
3	S	0	0	0	0	1	M3	M2	M1	M0	1	0	0
4	S	0	0	0	1	M3	M2	M1	M0	1	0	0	0
5	S	0	0	1	M3	M2	M1	M0	1	0	0	0	0
6	S	0	1	M3	M2	M1	M0	1	0	0	0	0	0
7	S	1	M3	M2	M1	M0	1	0	0	0	0	0	0

Clearly, the gain of using G.711 is not in quality but in the resulting bitrate: G.711 encodes a 12-bit, linearly quantized signal into 8 bits. If the sampling frequency is 8 kHz (the standard for telecom networks), the resulting bitrate is 64 kbit/s.

The only drawback of G.711 is to reduce the SNR for high-powered input signals (see Figure 2.9) compared with linear quantization. However, experience shows that the overall perceived (and subjective) quality is not dramatically impacted by the reduction of the SNR at high levels (listeners perceive some signal-independent noise).

In fact, most of the information is lost during initial sampling and 12-bit linear quantization. If listeners compare a CD quality sample recorded at a sample rate of 44.1 kHz at 16 bits, the critical loss of perceived quality occurs after subsampling at 8 kHz on 16 bits: there is a net loss of clarity and introduction of extra loudness, especially for the female voice. The reduction of quantization from 16 to 12 bits also introduces some granular noise. The final A- or μ -law logarithmic compression is relatively unimportant in this ‘degradation’ chain.

The A- or (μ)-law compression scheme is naturally a lossy compression: some noise is introduced and the input signal (on 12 bits) can never be recovered. This is true for all coders. All voice coders are designed for a given signal degradation target. The best coders for a given target are those that manage to use the smallest bitrate while still fulfilling the quality target.

Beyond the degradations mentioned above, the audio signal is low-pass-filtered (the conventional transmitted band is 300 Hz to 3,400 Hz in Europe and 200 Hz to 3,200 Hz in the US and Japan). This band limitation for the low frequencies of the speech signal throws out some essential spectral components of speech. It goes beyond the Nyquist requirements and was initially set for compatibility with analog modulation schemes for telephone multiplex links; it also takes into account the non-ideal frequency response of real filters.

Today, with the entire digital network going directly to customers’ premises (ISDN, cellular, and of course VoIP), this limitation is not mandatory and becomes obsolete.

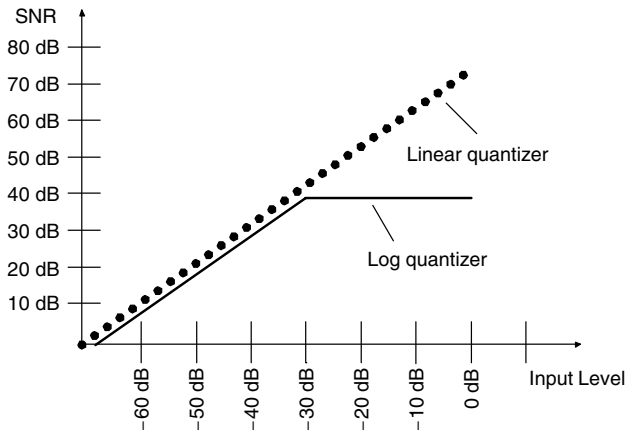


Figure 2.9 G.711 signal-to-noise ratio.

- SNR for linear quantizer: max = 74 dB (..);
- SNR for log type (A- or μ -law) quantizer: max = 38 dB (—).

The G.711 encoding process can be built very easily from off-the-shelf integrated circuits (priority encoders, etc.). G.711 encoding and decoding requires a very low processing power (hundreds of channels can be decoded in real time on a simple PC). In the early days of digital telecommunications, this was mandatory.

We will see that new coders are designed to give the same degradation for a lower bit rate:

- The required processing power increases (mainly for the coding part).
- The coding process introduces more delay (this is because coders need to look at more than one sample of the original signal before being able to produce a reduced bitrate version of the signal).

2.2 The basic tools of digital signal processing

2.2.1 Why digital technology simplifies signal processing

2.2.1.1 Common signal-processing operations

Signal-processing circuits apply a number of operations to the input signal(s):

- Sum.
- Difference.
- Multiplication (modulation of one signal by another).
- Differentiation (derivative).
- Integration.
- Frequency analysis.
- Frequency filtering.
- Delay.

It is obvious that the sum and difference operations are easy to perform with discrete time digitized signals, but they are also very easy to perform with analog systems. On the other hand, all other operations are much simpler to perform with digital systems.

The differentiation of a signal $f(t)$, for instance, typically requires an inductance or a capacitor in an analogue system, both of which are very difficult to miniaturize. But the derivative $f'(t) = \lim_{d \rightarrow 0} \frac{f(t+d) - f(t)}{d}$ can be approximated very easily by $(f(k) - f(k-1))/T$, where $f(k)$ is the discrete time digitized version of $f(t)$ with a sampling period T .

Similarly, the primitive F of a function f can be approximated on the digitized version of f summing all samples $f(k) * T$ (Figure 2.10).

All audio filters realizable using discrete components can today be emulated digitally. With the ever-increasing frequency of modern processors, even radio frequency signals are now accessible to digital signal processing.

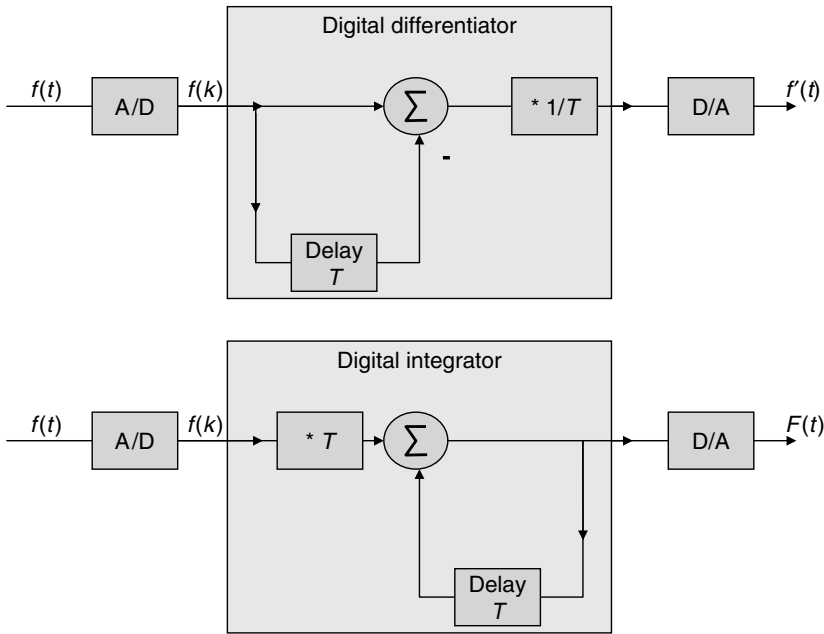


Figure 2.10 Differentiation and integration with digital filters.

The tools presented below allow engineers to synthesize digital filters that implement a desired behavior or predict the behavior of a given digital filter.

2.2.1.2 Example of an integro-differential filter

Most filters can be represented as a set of integro-differential equations between input signals and output signals. For instance, in the following circuit (Figure 2.11) the input voltage and the resulting current are linked by the following equation:

$$y''(t) + 3y'(t) + 2y(t) = f'(t)$$

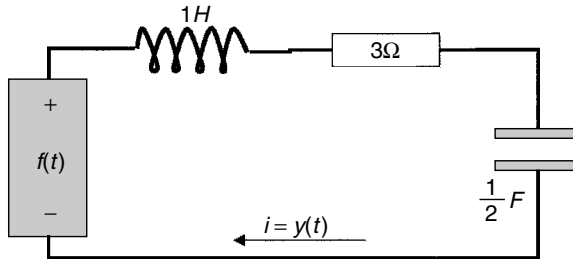


Figure 2.11 Simple circuit that can be modeled by an integro-differential equation.

or if D is the symbol of the differentiation operator:

$$(D^2 + 3D + 2)y(t) = Df(t)$$

The $D^2 + 3D + 2$ part is also called the characteristic polynomial of the system. The solutions of $x^2 + 3x + 2 = 0$ also give the value of the exponents of the pure exponential solutions of the equation when the input signal $f(t)$ is null ('zero input solution'). The reader can check that -1 and -2 are the roots of $x^2 + 3x + 2$ and that e^{-t} and e^{-2t} are solutions of the $y''(t) + 3y'(t) + 2y(t) = 0$ equation. The solutions are complex in general, but should occur as pairs of conjugates for real systems (otherwise the coefficients of the characteristic polynomial are not real), so that real solutions can be obtained by combining exponentials obtained from conjugate roots. Repeated roots r yield solutions of the form $t^{i-1}e^{rt}$ if the root is repeated i times.

Let's assume a sampling period of 1. The system equation can readily be transformed in a discrete time form (linear difference equation):

$$\frac{y[k+2] - 2y[k+1] + y[k]}{T^2} + 3\frac{y[k+1] - y[k]}{T} + 2y[k] = f[k+1] - f[k]$$

or if E denotes the 'advance operator' $E(f(k)) = f(k+1)$:

$$\left(\frac{E^2}{T^2} + \left(\frac{-2}{T^2} + \frac{3}{T}\right)E + \left(\frac{1}{T^2} - \frac{3}{T} + 2\right)\right)y[k] = (E - 1)f[k]$$

The left-hand side of this equation accepts solutions of the form $c\gamma^k$, where γ is a solution of the $\frac{x^2}{T^2} + \left(\frac{-2}{T^2} + \frac{3}{T}\right)x + \left(\frac{1}{T^2} - \frac{3}{T} + 2\right)$ polynomial. In the case of repeated roots, there are also solutions of the form $k^n\gamma^k$ (γ is generally a complex number).

2.2.2 The Z transform and the transfer function

2.2.2.1 Definition

The unilateral¹ Z transform of a discrete time function $f(k)$ is defined as $F(z) = \sum_{k=0}^{\infty} f(k)z^{-k}$. It is only defined on a certain domain of convergence of the complex variable z . The Z transform can be inverted:

$$f(k) = \frac{1}{2\pi j} \oint F(z)z^{k-1}dz$$

(the integral is performed on a closed path within the convergence domain in the complex plane).

¹ The bilateral Z transform also exists but is useful only for the analysis of non-causal systems. For the bilateral Z transform the sum starts at $-\infty$.

Table 2.4 Short extract of a Z transform table

$f(k)$	$F(z)$
$u(k)$ (step function $u(k) = 0, k < 0; u(k) = 1, k \geq 0$)	$\frac{z}{z-1}$
$ku(k)$	$\frac{z}{(z-1)^2}$
$\gamma^{k-1}u(k-1)$	$\frac{1}{z-\gamma}$
$k\gamma^k u(k)$	$\frac{\gamma z}{(z-\gamma)^2}$
$k^2\gamma^k u(k)$	$\frac{\gamma z(z+\gamma)}{(z-\gamma)^3}$

The Z transform is a linear operator: any linear combination of functions is transformed into the same linear combination of their respective Z transforms.

In practice these complex calculations are simplified by the use of transform tables that cover most useful signal forms. A small extract is presented in Table 2.4.

2.2.2.2 Properties

The Z transform has important properties. If $u(k)$ designates the step function ($u(k) = 0, k < 0; u(k) = 1, k \geq 0$) and $F(z)$ is the Z transform of $f(k)u(k)$, then:

- The Z transform of $f(k-1)u(k-1)$ is $1/z \cdot F(z)$. This is the delay property.
- The Z transform of $f(k-m)u(k-m)$ is $1/(z^m) \cdot F(z)$.
- The Z transform of $f(k-1)u(k)$ is $1/z \cdot F(z) + f(-1)$.
- The Z transform of $f(k+1)u(k)$ is $zF(z) - zf(0)$. This is the advance property.
- The Z transform of $f(k+2)u(k)$ is $z^2F(z) - z^2f(0) - zf(1)$.

The Z transform is a powerful tool to solve linear difference equations with constant coefficients.

2.2.2.3 Notation

Note that the Z transform of a unit delay is '1/z' and the z transform of a unit advance is 'z'. Both expressions will appear in diagrams in the following subsections.

In the following subsections, some figures will show boxes with an input, one or more outputs, adders, and multipliers, similar to Figure 2.12.

The meaning is the following: the sampled signal E is filtered by $H_1(z)$ resulting in response signal Y . Then, signal T is obtained by subtracting the previous output S (one sample delay) from signal Y . Finally, signal S is obtained by filtering signal T by filter $H_2(z)$.

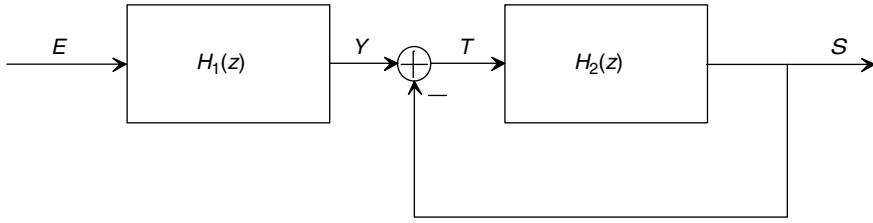


Figure 2.12 Typical digital filter representation.

2.2.2.4 Using the Z transform. Properties of the transfer function

With $T = \frac{1}{4}$ in the discrete difference equation above, for instance, we have:

$$(16E^2 - 20E + 6)y(k) = (E - 1)f(k)$$

If the Z transform of $y(k)u(k)$ is $Y(z)$ and the Z transform of $f(k)u(k)$ is $F(z)$, we have:

$$\begin{aligned} 16y(k+2) &\longrightarrow 16z^2Y(z) - 16z^2y(0) - 16zy(1) \\ -20y(k+1) &\longrightarrow -20zY(z) + 20zy(0) \\ 6y(k) &\longrightarrow 6Y(z) \\ f(k+1) &\longrightarrow zF(z) - zf(0) \\ -f(k) &\longrightarrow -F(z) \end{aligned}$$

We see that we have terms using $y(0)$ and $y(1)$ which are usually not known (but could be found by solving the equation iteratively for a given $f(k)$).

Let's try another approach and rewrite the equation as:

$$16y(k) - 20y(k-1) + 6y(k-2) = f(k-1) - f(k-2)$$

We get:

$$\begin{aligned} 16y(k) &\longrightarrow 16Y(z) \\ -20y(k-1) &= -20(y(k-1)u(k)) \\ &\longrightarrow -20(1/z \cdot Y(z) + y(-1)) = -20Y(z)/z \quad \text{if } y(-1) = 0 \\ 6y(k-2) &\longrightarrow 6(Y(z)/z^2 + y(-1)/z + y(-2)) = 6Y(z)/z^2 \quad \text{if } y(-2) = 0 \\ f(k-1)u(k) &\longrightarrow F(z)/z + f(-1) = F(z)/z \quad \text{if } f(p < 0) = 0 \text{ (causal input)} \\ f(k-2)u(k) &\longrightarrow F(z)/z^2 + f(-1)/z + f(-2) = F(z)/z^2 \quad \text{(causal input)} \end{aligned}$$

We obtain:

$$Y(z) = \frac{F(z)(1/z - 1/z^2)}{(16 - 20/z + 6/z^2)}$$

If we assume we want to find a solution with $f(k) = (2)^{-k}u(k) = (0.5)^k u(k)$ (the transform table tells us that $F(z) = z/(z - 0.5)$) we obtain:

$$Y(z) = \frac{z(z - 0.5)(1/z - 1/z^2)}{16 - 20/z + 6/z^2}$$

Decomposing the rational fraction into simpler components and using the transform table would give us $y(k)$.

The expression $Y(z)/F(z)$ is called the **transfer function** $H(z)$ of the system. We notice that the coefficients of $H(z)$ look familiar:

$$H(z) = \frac{z - 1}{16z^2 - 20z + 6}$$

When considering the equation using the advance operator form $(16E^2 - 20E + 6)y(k) = (E - 1)f(k)$, the numerator coefficients are the same as the coefficients for $f(k)$ and the denominator coefficients are the same as the coefficients for $y(k)$.

This is generally the case: the transfer function $H(z)$ can be obtained very simply from the coefficients of the difference equation using the advance operator (which will be shown in Subsection 2.2.2.5). This is one of the reasons the Z transform is so useful, even without complex calculations!

Another interesting property is that the Z transform of the **impulse response** of the system $h(k)$ is $H(z)$. The impulse function $\delta(0)$ is the input signal with $e(0) = 1$ and $e(k) = 0$ everywhere else. The impulse response is the response $s(k)$ of the system when the input is $\delta(0)$. The proof goes beyond the scope of this book.

2.2.2.5 Application for FIR and IIR filters

The impulse response $h(k)$ of a linear, time-invariant, discrete time filter determines its response to any signal:

- Because of linearity, the response to an impulse of amplitude a is $ah(k)$.
- Because of time invariance, the response to a delayed impulse $\delta(k - x)$ is $h(k - x)$.

Any input signal $e(k)$ can be decomposed into a sum of delayed impulses, and because of the linearity of the filter we can calculate the response. Each impulse $e(x)$ creates a response $e(x)h(k - x)$, where k is the discrete time variable. The response $s(n)$ at instant n is $e(x)h(n - x)$. The sum of all the response components at instant n for all $e(x)$ is:

$$s(n) = \sum_{x=-\infty}^{\infty} e(x)h(n - x)$$

This is a convolution of e and h in the discrete time domain. This relation is usually rewritten by taking $m = n - x$:

$$s(n) = \sum_{m=-\infty}^{m=+\infty} e(n - m)h(m)$$

For a physically realizable system (which cannot guess a future input signal and therefore cannot react to $\delta(0)$ before time 0), we must have $h(x) = 0$ for $x < 0$. Physically realizable systems are also called **causal systems**.

Filters that only have a finite impulse response are called **finite impulse response (FIR) filters**. The equation for an FIR filter is:

$$s(n) = \sum_{k=0}^{k=N} e(n-k)h(k)$$

where the $h(k)$ for $k = 0$ to N are constants that characterize the system. In voice-coding filters these constants are sometimes dynamically adapted to the signal, but with a timescale that is much lower than the variance of the signal itself: they are in fact a succession of FIR filters with varying coefficients.

Filters that have an infinite impulse response are called **infinite impulse response (IIR) filters**. In many filters the response is infinite because recursivity has been introduced in the equation of the filter. The equation for a recursive IIR filter is:

$$s(n) = \sum_{k=0}^{k=N} e(n-k)a(k) - \sum_{k=1}^{k=L} s(n-k)b(k)$$

Note that we have introduced the past values of the output ($s(n-k)$) in the formula and that the values $a(k)$ and $b(k)$ characterize the system. When we compute the Z transform of the time domain equation of an IIR filter:

$$s(n) = \sum_{k=0}^{k=N} e(n-k)a(k) - \sum_{k=1}^{k=L} s(n-k)b(k)$$

we obtain²:

$$S(z) = E(z) \sum_{k=0}^{k=N} z^{-k}a(k) - S(z) \sum_{k=1}^{k=L} z^{-k}b(k)$$

or

$$S(z) \left(1 + \sum_{k=1}^{k=L} z^{-k}b(k) \right) = E(z) \sum_{k=0}^{k=N} z^{-k}a(k)$$

$$\begin{aligned} 2 \sum_{n=-\infty}^{\infty} \left(\sum_{k=0}^N e(n-k)a(k) \right) z^{-n} &= \sum_{k=0}^N \sum_{n=-\infty}^{\infty} e(n-k)a(k)z^{-n} = \sum_{k=0}^N a(k) \sum_{n=-\infty}^{\infty} e(n-k)z^{-n} \\ &= \sum_{k=0}^N a(k)z^{-k}E(z) \end{aligned}$$

We can also calculate the output-to-input ratio in the Z domain:

$$S(z) = E(z)H(z) \quad \text{with} \quad H(z) = \frac{\sum_{k=0}^{k=N} z^{-k} a(k)}{1 + \sum_{k=1}^{k=L} z^{-k} b(k)}$$

We see that the transfer function in the Z domain has an immediate expression from the coefficients of the filter equation.

2.2.2.6 System realization

A given transfer function $H(z)$ is easily realizable by a discrete time filter. For instance, if:

$$H(z) = \frac{b_3 z^3 + b_2 z^2 + b_1 z + b_0}{z^3 + a_2 z^2 + a_1 z + a_0}$$

then a first step would be to obtain:

$$X(z) = \frac{1}{z^3 + a_2 z^2 + a_1 z + a_0} * F(z)$$

which is easy by considering the corresponding difference equation:

$$x(k+3) = -a_2 x(k+2) - a_1 x(k+1) - a_0 x(k) + f(k)$$

which is realized by a system like that in Figure 2.13.

A second step is to obtain $Y(z)$ by a linear combination of the $z^i X(z)$, as in Figure 2.14.

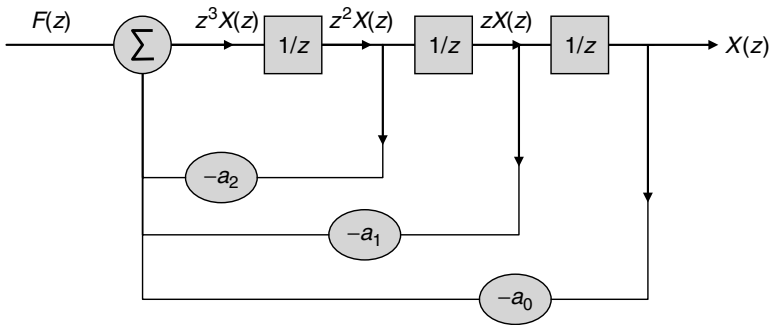


Figure 2.13 Realization of $H(z)$ denominator.

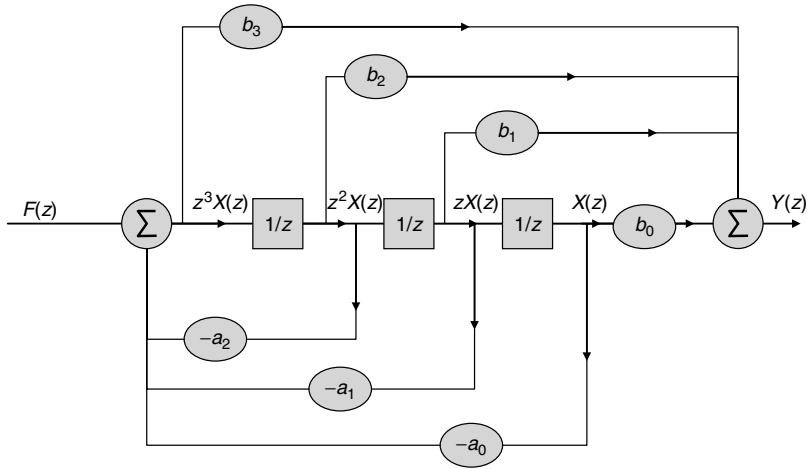


Figure 2.14 Full realization of $H(z)$.

2.2.2.7 Realization of frequency filters

The fact the transfer function $H(z) = Y(z)/F(z)$ is also the Z transform of the impulse response makes it very useful to determine the frequency response of a discrete time filter. If $h(k)$ is the impulse response of a system, the system response $y(k)$ to input z^k is:

$$y(k) = \text{convolution}(h(k), z^k) = \sum_{m=-\infty}^{\infty} h(m)z^{k-m} = z^k \sum_{m=-\infty}^{\infty} h(m)z^{-m} = H(z)z^k$$

where $H(z)$ is the Z transform of the filter impulse response and the transfer function as well. A sampled continuous time sinusoid $\cos(\omega t)$ is of the form $\cos(\omega T k) = \text{Re}(e^{j\omega T k})$ where T is the sampling period. A sample sinusoid respecting the Nyquist limit must have $\omega < \pi/T$. If we take $z = e^{j\omega T}$, the above result tells us that the frequency response of the filter to the discrete time sinusoid is:

$$y(k) = H(e^{j\omega T}) \cdot e^{j\omega T k}$$

Therefore, we can predict the frequency response of a system by studying $H(e^{j\omega T})$. $H(z)$ can be rewritten as a function of its zeros z_i and its poles p_i :

$$H(z) = b_n \frac{(z - z_1)(z - z_2) \cdots (z - z_n)}{(z - p_1)(z - p_2) \cdots (z - p_m)}$$

for stable systems the poles must be inside the unit complex circle, and for physically realizable systems we must have $n < m$ and poles and zeros should occur as pairs of conjugates.

A graphical representation of the transfer function (Figure 2.15) for two zeros and two poles makes it simple to understand how H behaves as a function of ω .

The amplitude of the original sinusoid is multiplied by:

$$|H(e^{j\omega T})| = b_n \frac{d_{z_1} d_{z_2} \cdots d_{z_n}}{d_{p_1} d_{p_2} \cdots d_{p_m}}$$

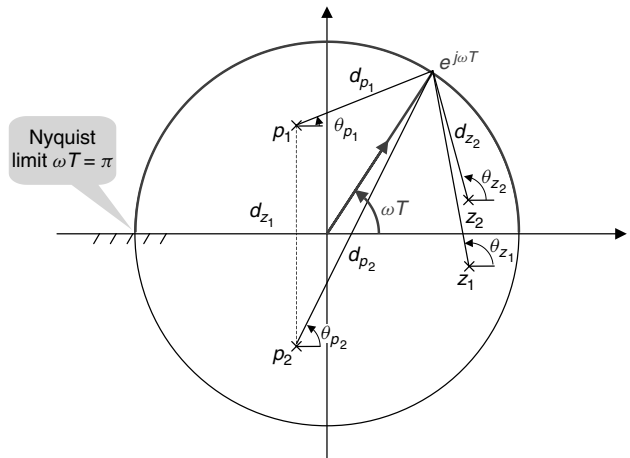


Figure 2.15 Graphical interpretation of the transfer function.

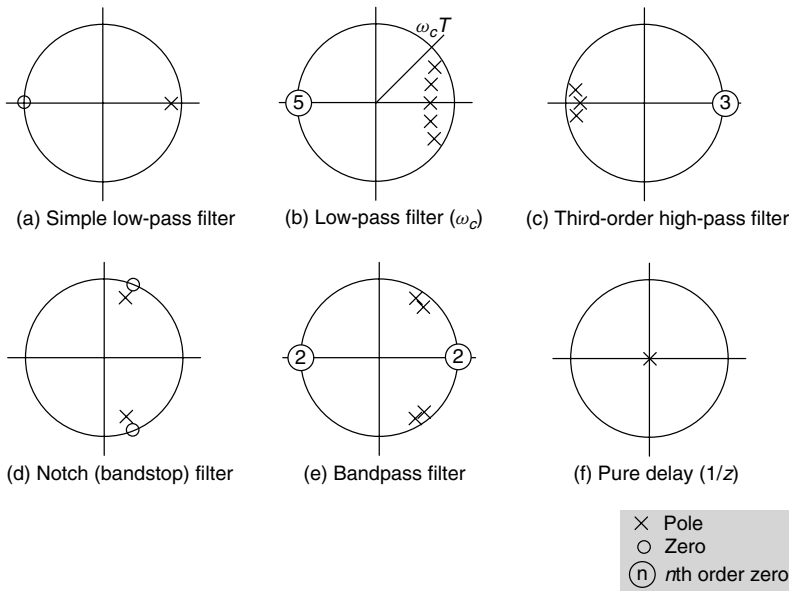


Figure 2.16 Graphical representation of common filters.

and the phase of the original sinusoid is changed by the angle:

$$\angle H(e^{j\omega T}) = (\theta_{z_1} + \theta_{z_2} + \dots + \theta_{z_n}) - (\theta_{p_1} + \theta_{p_2} + \dots + \theta_{p_n})$$

Frequency filters can be realized by placing poles near the frequencies that need to be amplified and zeros near the frequencies that need to be attenuated. Figure 2.16 gives a few examples.

In a simple low-pass filter (case A), a pole is placed near point 1 (it needs to be inside the unit circle for a stable system), and a zero at point -1 (zeros can be anywhere). The cut-off frequency for such a filter is at $\omega T = \pi/2$.

In order to ensure that the gain is sustained on a specific band $[0 - \omega_c]$, more poles must be accumulated near the unit circle in the band where the gain must be close to unity (case B).

The principle of the high-pass filter (C) is similar, but the roles of the zeros and poles are inverted. A higher order filter will have a sharper transition at the cut-off frequency and therefore will better approach an ideal filter. Note that a realizable system (where the future is not known in advance) requires more poles than zeroes or an equal number of poles and zeros.

A notch (bandstop) filter (D) is obtained by placing a zero at the frequency that must be blocked. A zero must be placed at the conjugate position for a realizable system (all the coefficients of the polynomials of the transfer function fraction must be real). Poles can be placed close to the zeros to quickly recover unity gain on both sides of the blocked frequency.

A bandpass filter (E), can be obtained by enhancing the frequencies in the transmission band with poles and attenuating frequencies outside this band by placing zeros at points 1 and -1 .

Note that a pole placed at the origin (F) does not change the amplitude response of the filter, and therefore a pole can always be added there to obtain a physically realizable system (more poles than zeros). A filter with a single pole at the origin is in fact a pure delay of period T (linear phase response of $-\omega T$). This is logical: filters cannot be realized if they need to know a future sample \dots and can be made realizable by delaying the response of the filter in order to accumulate the required sample before computing the response. Similarly a zero at the origin is a pure advance of T .

The ease with which arbitrary digital filters can be realized using the results of this section and the method of the previous section sharply contrasts with the complexity of analog filters, especially for high-order filters. This is the reason discrete time signal processing has become so prevalent.

2.2.3 Linear prediction for speech-coding schemes

2.2.3.1 Linear prediction

Linear prediction is used intensively in speech-coding schemes; it uses a linear combination of previous samples to construct a predicted value that attempts to approach the next input sample:

$$s_p(n) = \sum_{k=1}^{k=p} a_k s(n-k)$$

gives the predicted value at time n .

The coefficients a_k must be chosen to approach the $s(n)$ value. If $s_p(n)$ is indeed similar to $s(n)$, then the error signal $e(n) = s(n) - s_p(n)$ can be viewed as a residual signal resembling a white noise.

With this remark, we can decompose the issue of transmitting speech information (the waveform) into two separate problems:

- The transmission of the set of coefficients a_k (or some coded representations).
- The transmission of information related to the error signal $e(n)$.

Ideally, if $e(n)$ was **white noise**, then only its power should be sent. In reality, $e(n)$ is not white noise and the challenge to speech coder experts is to model this error signal correctly and to transmit it with a minimal number of bits.

The a_k coefficients are called **linear prediction coefficients (LPCs)** and p is the order of the model. Each LPC vocoder uses its own methods for computing the optimal a_k coefficients. One common method is to compute the a_k that minimize the quadratic error on the samples to predict, which leads to a linear system (the equation of Yule–Walker) that can be solved using the Levinson–Shur method. Usually, these coefficients are computed on a frame basis of 10–30 ms during which the speech spectrum can be considered as stationary.

2.2.3.2 The LPC modeling filter

In the previous subsection, we showed that a speech signal s could be approached by a linearly predicted signal s_p obtained by an LPC filter L . Another way to view this is to say that the speech signal, filtered by the $(1 - L)$ filter, is a residual error signal ideally resembling white noise.

At this point, it is interesting to wonder whether the inverse filter can approach the original speech spectrum by filtering an input composed of white noise. To find the expression of the inverse filter, we can use the previous equation, replacing $s_p(n)$ by its expression as a function of s :

$$e(n) = s(n) - \sum_{k=1}^{k=p} a_k s(n-k)$$

or in the Z domain:

$$E(z) = S(z) \left(1 - \sum_{k=1}^{k=p} a_k z^{-k} \right)$$

So we have:

$$S(z) = \frac{E(z)}{\left(1 - \sum_{k=1}^{k=p} a_k z^{-k} \right)}$$

which gives the ‘speech’ signal by filtering an input composed of white noise.

The digital filter:

$$H(z) = \frac{1}{\left(1 - \sum_{k=1}^{k=p} a_k z^{-k} \right)} = \frac{1}{A(z)}$$

is called the LPC modeling filter. It is an all pole filter (no zero) that models the source (speech). If we want to evaluate the residual error signal we only need to filter the speech signal ($s(n)$) by the filter $A(z)$ because we have $E(z) = S(z)A(z)$. $A(z)$ is often called

the LPC analysis filter (giving the residual signal) and $H(z) = 1/A(z)$ the LPC synthesis filter (giving the speech signal from the residual signal). These concepts are intensively used in the low-bitrate speech coder schemes discussed in the following section.

Note that we are only trying to approach the frequency spectrum of the original speech signal, not the exact time representation: this is because human hearing is not sensitive to the exact phase or time representation of a signal, but only to its frequency components.

2.3 Overview of speech signals

2.3.1 Narrow-band and wide-band encoding of audio signals

Audio engineers distinguish five categories of audio quality:

- The telephony band from 300 Hz to 3,400 Hz. An audio signal restricted to this band remains very clear and understandable, but does alter the natural sound of the speaker voice. This bandwidth is not sufficient to provide good music quality.
- The audio wide-band from 30 Hz to 7,000 Hz. Speech is reproduced with an excellent quality and fidelity, but this is still not good enough for music.
- The hi-fi band from 20 Hz to 15 kHz. Excellent quality for both voice and music. Hi-fi signals can be recorded on one or multiple tracks (stereo, 5.1, etc.) for spatialized sound reproduction.
- The CD quality band from 20 Hz to 20 kHz.
- Professional quality sound from 20 Hz to 48 kHz

Table 2.5 shows the bitrate that needs to be used for each level of audio quality, without compression.

2.3.2 Speech production: voiced, unvoiced, and plosive sounds

Speech sounds are characterized by the shape of the vocal tract which consists of the vocal cords, the lips, and the nose [B1]. The overall frequency spectrum of a speech sound is determined by the shape of the vocal tract and the lips (Figure 2.17). The vocal

Table 2.5 Uncompressed bitrate requirements according to audio quality

	Sampling frequency (kHz)	Quantization (bits)	Nominal bitrate (kbit/s)
Telephony	8	13	104
Wide-band	16	14	224
Hi-fi	32	16	512 mono (1,024 stereo)
CD	44.1	16	705.6 mono (1,411 stereo)
Professional	96	24	13,824 (5.1 channels)

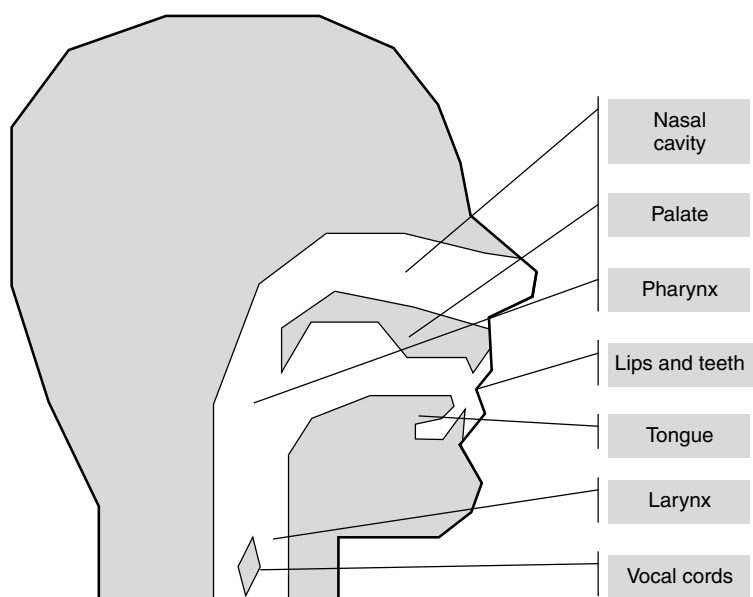


Figure 2.17 Human voice production.

tract introduces resonance at certain frequencies called formants. This resonance pattern carries a lot of information.

There are mainly three types of speech sounds: **voiced**, **unvoiced**, and **plosive**.

Periodically closing and opening the vocal cords produces voiced speech. The period of this closing and opening cycle determines the frequency at which the cords vibrate; this frequency is known as the pitch of voiced speech. The pitch frequency is in the range of 50–400 Hz and is generally lower for male speakers than for female or child speakers. The spectrum of a voiced speech sample presents periodic peaks at the resonance frequency and its odd harmonics (the formants). The voiced speech spectrum can be easily modeled by an all-pole filter with five poles or ten real coefficients computed on a frame length of 10–30 ms.

During unvoiced speech, such as ‘s’, ‘f’, ‘sh’, the air is forced through a constriction of the vocal cords; unvoiced speech samples have a noise-like characteristic and consequently their spectrum is flat and almost unpredictable.

Speech is produced by the varying state of the vocal cords, and by the movement of the tongue and the mouth. Not all speech sounds can be classified as voiced or unvoiced. For instance, ‘p’ in ‘puff’ is neither a voiced nor an unvoiced sound: it is of the plosive type.

Many speech sounds are complex and based on superimposing modes of production, which makes it very difficult to correctly model the speech production process and consequently to encode speech efficiently at a low bitrate.

Figures 2.18–2.23 give some samples of voiced, unvoiced, and mixed speech segments, and their corresponding frequency spectrum associated with a 10th order LPC modeling filter frequency response.

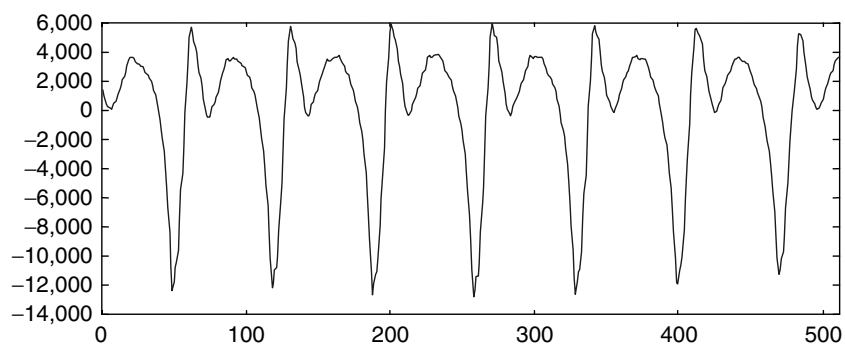


Figure 2.18 Time representation of a voiced speech sequence (in samples).

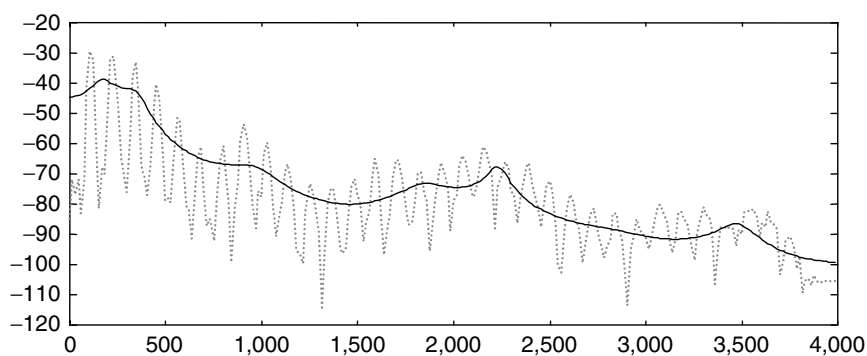


Figure 2.19 Frequency spectrum of the voiced speech segment (dotted line) and the 10th order LPC modelling filter response.

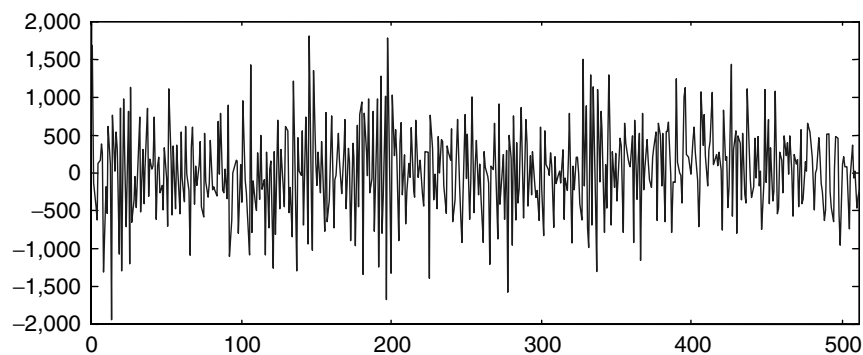


Figure 2.20 Time representation of an unvoiced speech sequence (in samples).

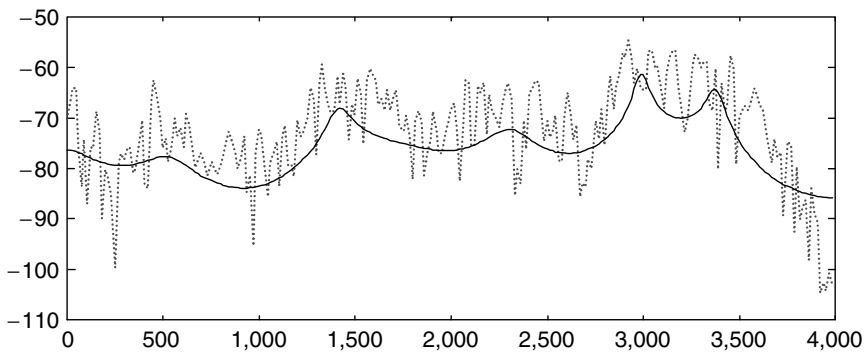


Figure 2.21 Frequency spectrum of the unvoiced speech segment (dotted line) and the 10th order LPC modelling filter response.

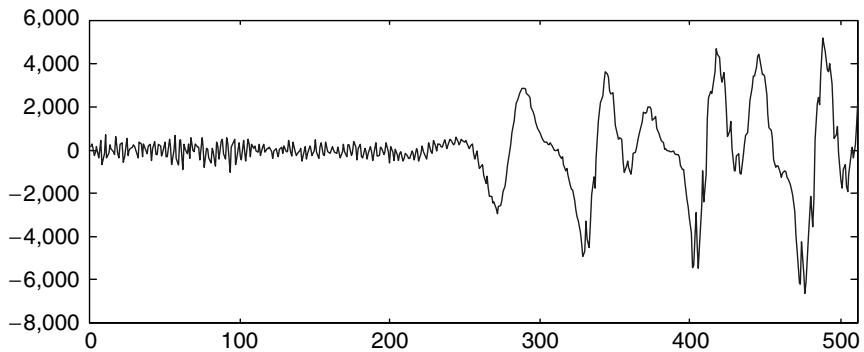


Figure 2.22 Time representation of a mixed speech sequence (in samples).

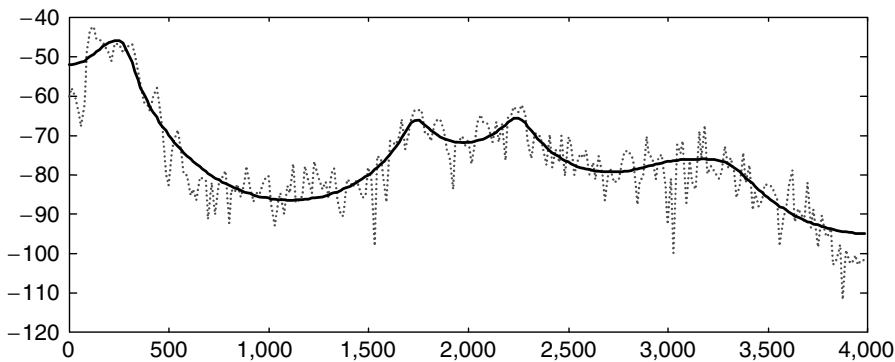


Figure 2.23 Frequency spectrum of the mixed speech segment (dotted line) and the 10th order LPC modelling filter response.

2.3.3 A basic LPC vocoder: DOD LPC 10

By being able to distinguish voiced and unvoiced speech segments, it is possible to build a simple source filter model of speech (Figure 2.24) and a corresponding source speech coder, also called a **vocoder** (Figure 2.25). The detection of voiced segments is based on the autocorrelation of the processed frame after filtering through the LPC analysis filter. If the autocorrelation is rather flat and there is no obvious pitch that can be detected, then the frame is assumed to be unvoiced; otherwise, the frame is voiced and we have computed the pitch.

The DOD 2,400-bit/s LPC 10 [A8] speech coder (called LPC 10 because it has ten LP coefficients) was used as a standard 2,400-bit/s coder from 1978 to 1995 (it was subsequently replaced by the mixed excitation linear predictor, or MELP, coder). This vocoder has parameters as shown in (Table 2.6).

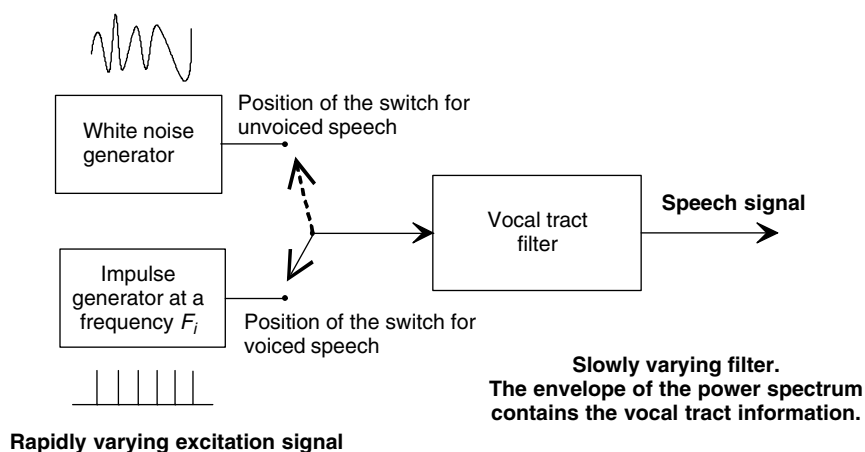


Figure 2.24 DOD LPC 10 voice synthesis for voiced and unvoiced segments (a source filter model of speech).

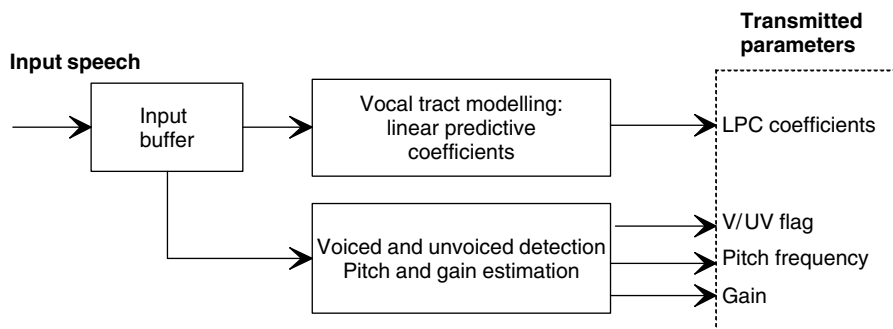


Figure 2.25 Basic principle of a source speech coder called a vocoder.

Table 2.6 DOD LPC 10 frame size, bit allocation and bitrate

Sampling frequency	8 kHz
Frame length	180 samples = 22.5 ms
Linear predictive filter	10 coefficients = 42 bits
Pitch and voicing information	7 bits
Gain information	5 bits
Total information	54 bits per frame = 2,400 bit/s

The main disadvantage of source coders, based on this simple voiced/unvoiced speech production model, is that they generally give a very low speech quality (synthetic speech). Such coders cannot reproduce toll-quality speech and are not suitable for commercial telephony applications. The MELP coder made some progress by being able to model voice segments as a mix of voiced and unvoiced sounds, as opposed to a binary choice.

2.3.4 Auditory perception used for speech and audio bitrate reduction

The coders described previously attempt to approach the exact frequency spectrum of the source speech signal. This assumes that human hearing can perceive all frequencies produced by the speaker. This may seem logical, but human hearing cannot in fact perceive any speech frequency at any level. All acoustical events are not audible: there is a curve giving the perception threshold, depending on the sound pressure level and the frequency of the sound [A4, A9, A14]. Weak signals under this threshold cannot be perceived. The maximum of human hearing sensitivity is reached between 1,000 Hz and 5,000 Hz. In addition some sounds also affect the sensitivity of human hearing for a certain time. In order to reduce the amount of information used to encode speech, one idea is to study the sensitivity of human hearing in order to remove the information related to signals that cannot be perceived. This is called ‘perceptual coding’ and applies to music as well as voice signals.

The human ear is very complex, but it is possible to build a model based on **critical band** analysis. There are 24 to 26 critical bands that overlap bandpass filters with increasing bandwidth, ranging from 100 Hz for signals below 500 Hz to 5,000 Hz for signals at high frequency.

In addition a low-level signal can be inaudible when masked by a stronger signal. There is a predictable time zone, almost centered on the masker signal, that makes all the signals inside this area inaudible, even if they are above their normal perception threshold. This is called **simultaneous frequency domain masking**, which is used intensively in perceptual audio-coding schemes and includes pre- and post-masking effects.

Although these methods are not commonly used in low-bitrate (4–16 kbit/s) speech coders, they are included in all the modern audio coders (ISO MPEG-1 Layer I, II, III,³ MPEG-2 AAC, AC3, or Dolby Digital). These coders rely on temporal to frequency domain transformation (analysis filter bank) coupled to an auditory system-modeling

³ The MPEG-1 Layer III audio coder is also known as MP3 for Web users.

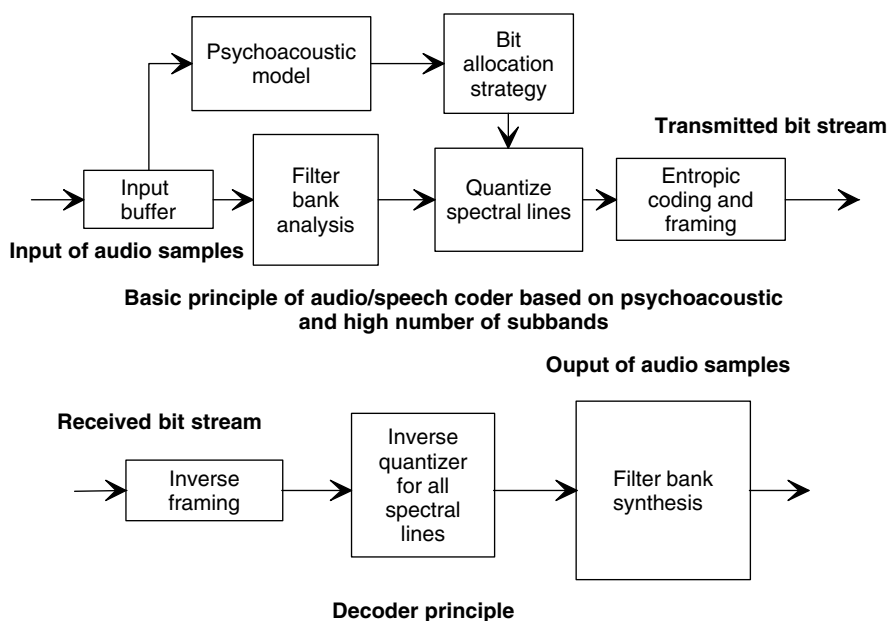


Figure 2.26 Usage of filter banks for audio signal analysis and synthesis.

procedure that calculates masking thresholds and drives a dynamic bit allocation function. Bits are allocated to each band in order to fit the overall bitrate and masking threshold (see Figure 2.26) requirements.

Today, audio signals can be efficiently encoded (almost CD-like quality [A9, A10, A11, A12]) in about 64 kbit/s for a single monophonic channel with the most advanced audio-coding (AAC) schemes. Wideband (20–7,000 Hz) speech and audio coders can use the same scheme to encode in only 24 kbit/s or 32 kbit/s (although there are some issues related to the analysis filter bank—overlap and add procedure in the decoder—that result in annoying pre-echo phenomena. This is mainly due to the nonstationary characteristic of the speech signal and is very perceptible when onset appears).

Some low-bitrate speech coders do not use the perceptual model for speech coding itself, but rather to better evaluate the residual error signal. **Analysis by synthesis (ABS)** speech coders (addressed later) ponder the error signal used in the closed-loop search procedure by a perceptual weighting filter derived from the global spectrum of speech. The function of this perceptual weighting filter is to redistribute the quantizing noise into regions where it will be masked by the signal. This filter significantly improves subjective coding quality by properly shaping the spectrum of the error: error noise is constrained to remain below the audible threshold when the correlated signal is present. In ABS decoders, a post-filter may also be used to reduce noise between the maxima of the spectrum (formants) by reducing the signal strength in these regions and boosting the power of formants. This significantly improves the perceived quality on the MOS scale, but there is a price to pay: post-filters do alter the naturalness (fidelity) of the decoded speech. An example of such a filter is given in the introduction to Section 2.7.

2.4 Advanced voice coder algorithms

2.4.1 Adaptive quantizers. NICAM and ADPCM coders

We have already mentioned that if the **probability density function (PDF)** of the input is known one optimal quantizer can be computed for the signal. Linear or logarithmic quantizers are time-unvarying systems: their step sizes are fixed for the entire duration of the signal. Logarithmic quantizers (such as G.711) are an optimization that provides an SNR independent of the level of the signal.

It is also possible to adapt the quantizer dynamically to best match the instantaneous characteristics of the signal.

Many voice coders use dynamic quantization algorithms. The rules and types of adaptation used to encode the signal can be transmitted with the encoded signal (forward adaptive quantizers), but this is not required: backward adaptive quantizers use only the characteristics of the previously transmitted encoded signal to optimize the processing of the current sample(s), enabling the receiver to also compute the optimal adaptation that will be used for the next received encoded signal.

In addition, the optimal characteristics of the adaptive quantizer can be selected (or computed) for each sample, based on the characteristics of a group of contiguous samples (such a group is called a **frame**). A frame-based adaptation procedure is more efficient in terms of transmitted bitrate, especially when forward quantizer selection is used. The size of the frame must be selected carefully: if the size is too small there may be a large overhead for transmitting the scaling information, but if the block size is too large the quantizing steps may become inadequate for some portions of the frame, leading to large errors in the quantization process.

Figure 2.27 shows the principle of a forward adaptive quantizer and Figure 2.28 shows the principle of an inverse forward adaptive quantizer.

NICAM is an example of a coder using a forward adaptive quantizer (Figure 2.29). The **NICAM (near-instantaneous companding and multiplexing)** system is used to transmit the audio stereo signal digitally on analog TV channels using the PAL and SECAM TV color systems. The NICAM system transmits two stereo audio channels sampled at

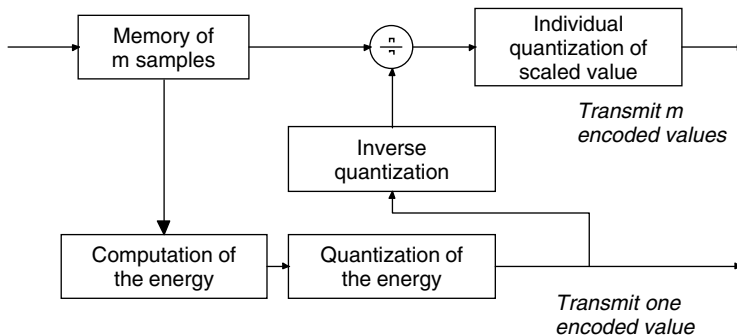


Figure 2.27 Principle of a forward adaptive quantizer.

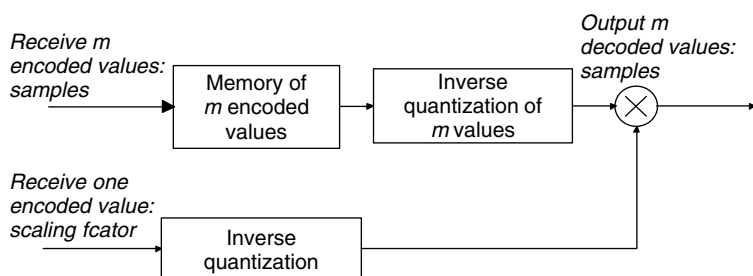


Figure 2.28 Principle of a forward adaptive inverse quantizer.

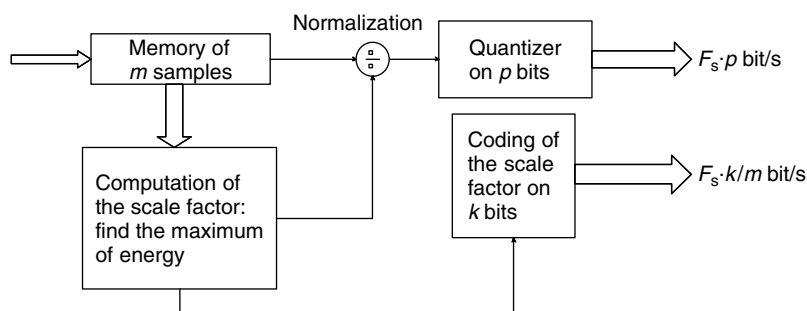


Figure 2.29 Near-instantaneous quantizer using: $F_s \cdot (p + k/m)$ bit/s.

32 kHz in a bitrate of 728 kbit/s. NICAM memorizes a buffer (the near-instantaneous characteristic . . .) of 32 samples and evaluates the mean power during this period of time, which is used to normalize the input samples. A fixed 10-bit logarithmic quantizer is then used on the normalized signal. The transmitted frame comprises the individual quantized samples, the scaling factor information, framing information, and some parity bits that protect the compressed audio signal against transmission errors.

It has been shown that, for the same subjective quality, the use of the quasi-instantaneous (32 samples) system requires 10.1 bits per sample compared with 11 bits per sample using a classical sample by sample logarithmic quantizer. There is a gain of 10% resulting from the use of block analysis and the forward ‘adaptive’ quantizer.

For backward adaptive quantizers, there is no need to transmit any information related to the scaling procedure; the mean power is estimated on the quantized signal and, therefore, the inverse quantizer can reconstruct this information exactly (Figures 2.30 and 2.31).

A very simple but efficient backward adaptive quantizer called ‘one-word memory’ is used in the ADPCM G.726 and G.727 ITU-T speech coders [A13]. A simple coefficient M_i depending only on the previous quantized sample determines the compression or expansion of the quantization steps for the next sample. If the quantizer has 4 bits (1 sign bit and 8 ranges of quantization), there are 8 M_i fixed coefficients (each implicitly associated with a quantizing range) insuring the compression or expansion of the quantizer. When large values are input to the quantizer, the multiplier value is greater than 1 and for small previous values the multiplier value is less than 1. This tends to force the

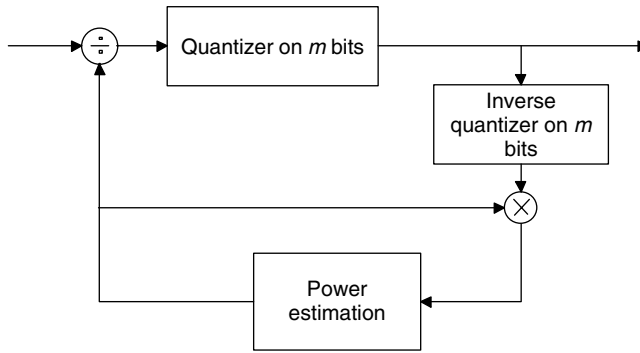


Figure 2.30 Principle of a backward adaptive quantizer.

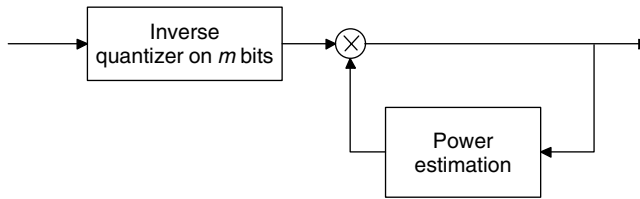


Figure 2.31 Principle of an m bit backward inverse quantizer.

adaptive quantizer to track the dynamics of the input signal (we can also consider that the previous measurement gave us some information on the probability density for the next sample, which we use to optimize the quantification). A fixed quantizer can be used and there is no need to transmit any scaling information to the decoder side (see Figures 2.32 and 2.33). Transmission errors will cause desynchronization of the coder and the decoder for a single sample.

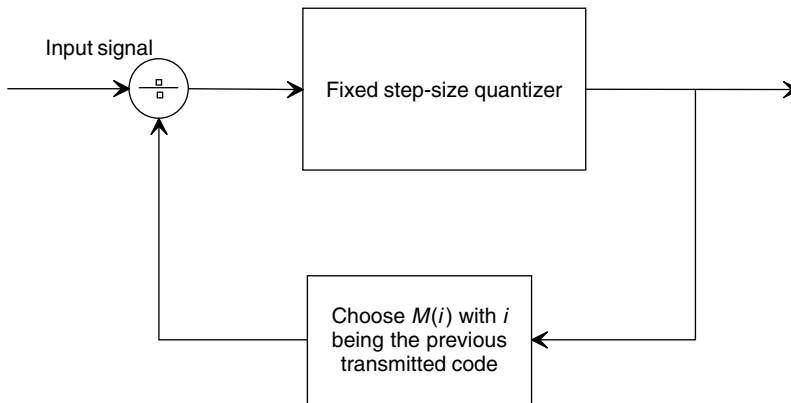


Figure 2.32 One-word memory adaptive quantizer.

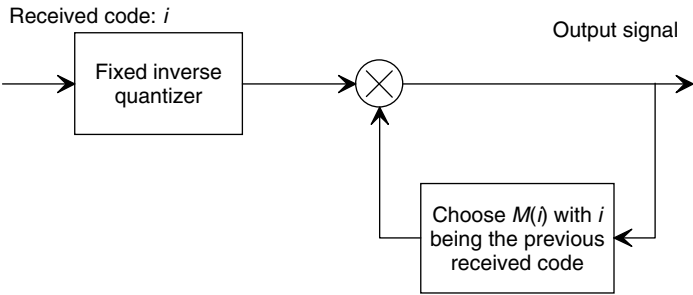


Figure 2.33 One-word memory adaptive inverse quantizer.

Table 2.7 Eight expansion coefficients attached to eight quantization ranges

M_0	0.969
M_1	0.974
M_2	0.985
M_3	1.006
M_4	1.042
M_5	1.101
M_6	1.208
M_7	1.449

Table 2.7 gives the M_i values for an eight-level quantizer (for each sign) optimized for exponential distribution.

The G.726 quantizer only needs to send 4 bits per sample (32 kbit/s), instead of 8 for G.711. G.726 is commonly used on many PSTN communication links when there is a need to reduce the transmitted bitrate.

2.4.2 Differential predictive quantization

In speech and audio signals, there is a strong correlation between the present sample and the previous one. The consequence is that if we subtract the previous sample from the present one, the variance of the difference signal will be lower than the variance of the original signal: it will require less bits to be quantized.

Unfortunately, we cannot directly use the exact previous sample value because it is inaccessible to the decoder. Instead, we must use the value of the previous sample as decoded by the receiver. In order to do this, the encoder relies on a ‘local decoder’ feedback loop which is common in speech and audio compression schemes. We have:

$$E(n) = X(n) - X_d(n - 1)$$

where $X_d(n - 1)$ is the decoded value at time $n - 1$, and we transmit the quantized version of $E(n)$, which is $Q[E(n)]$.

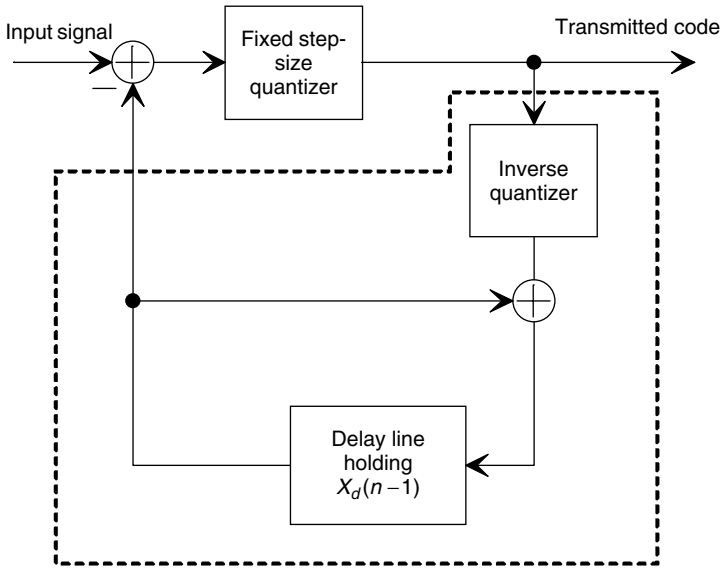


Figure 2.34 Principle of a differential quantizer (one-word memory prediction). The local decoder is in the dotted box and is identical to the distant decoder.

At the decoder side, we can compute the decoded value at time n :

$$X_d(n) = X_d(n-1) + Q^{-1}[Q[E(n)]] = X(n) + (Q^{-1}[Q[E(n)]] - E(n))$$

$X_d(n)$ approaches $X(n)$, but has the small difference introduced by quantization noise: $(Q^{-1}[Q[E(n)]] - E(n))$. If Q was ideal, then the noise signal would be zeroed.

Figure 2.34 illustrates the basic principle of a waveform speech or audio coder (called waveform because it tracks the temporal shape of the signal, as opposed to its frequency spectrum): all the concepts such as prediction or differential encoding are present.

The previous scheme is not a realistic one due to its sensibility to transmission errors: any transmission error will permanently desynchronize the decoder.

A more robust solution is to use a correlation coefficient:

$$E(n) = X(n) - X_d(n-1)$$

is replaced by:

$$E(n) = X(n) - C_1 * X_d(n-1)$$

where C_1 is the correlation coefficient. A value below unity will decrease the influence of a transmission error over time.

Like all linear prediction schemes, this works only if there is some correlation in the input signal (i.e., it does not exhibit a flat frequency spectrum (white noise)). In the case of noise, there is no correlation between adjacent input samples. There is no chance to predict the future sample knowing the previous one. By contrast, speech and audio

signals, due to their production mode, exhibit a non-flat spectrum and consequently high correlation exists between samples.

This differential encoding method can be generalized by using more than one previous sample to build the predicted term and by using a dynamically computed correlation factor:

- In waveform or temporal coders working on a sample-by-sample basis, a temporal prediction of the signal is built from a linear combination of previous (decoded) samples. The coefficients are not transmitted; they are computed by a symmetrical procedure in the decoder.
- Vocoder or ABS speech coders filter the input signal by an inverse model based on correlation coefficients. It is the residual signal (output of the filter) which is encoded and transmitted with the modeling filter coefficients (called linear prediction coefficients, LPCs). LPC analysis is typically performed on a time frame of 10–30 ms at a sampling frequency of 8 kHz. This is a period of time where the speech signal can be considered as quasi-stationary.

In these algorithms, based on a history of more than one sample, the term $X_d(n)$ is replaced by $X_p(n)$, which is the value of the predicted signal based on previous samples:

$$X_p(n) = \sum_{i=1}^{i=N} A_i X(n-i)$$

As indicated in Figure 2.35, coefficients A_i can be fixed or ‘adaptive’ (i.e., computed for each new sample). When fixed, they are nonoptimal and derived from the average (if it really exists) frequency spectrum of the signal.

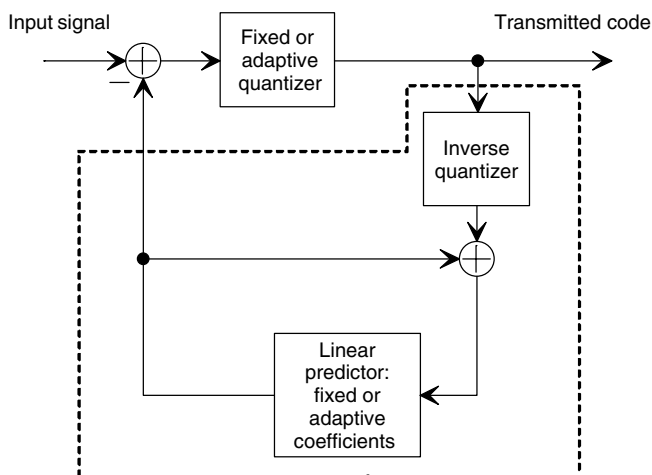


Figure 2.35 General principle of a differential (fixed or adaptive) coder. The local decoder is in the dotted box and is identical to the distant decoder.

Computation of the set of coefficients A_i in order to minimize quadratic error requires solving a set of linear equations [B2]. Even for a frame-by-frame analysis (such as for vocoder or ABS coders), this is a complex computational task which is out of reach of most real-time implementations. Many approximation algorithms have been developed to reduce computational complexity:

- For waveform coders, the set A_i , which is generally not transmitted, is continuously (on a sample-by-sample basis) adapted by the ‘stochastic gradient algorithm’ or by a simple ‘sign’ algorithm, where the absolute value of coefficients with the same sign as the error are reduced, and vice versa.
- For frequency or analysis by synthesis speech-coding schemes, the set A_i must be quantized and transmitted to the decoder side. The set of coefficients A_i or similar quantities modeling the short-term (10–30 ms) spectrum of the speech signal have to be computed. The direct inversion of the matrix obtained by expressing the minimization of quadratic errors is not used. More efficient algorithms have been studied and tuned to efficiently compute LPCs and to quantize them. Among them, the Levinson–Durbin algorithm and the Schur recursion are the most frequently used iterative methods to compute the A_i (Levinson–Durbin algorithm) or some partial coefficients called *parcours* (Schur recursion).

2.4.3 Long-term prediction for speech signal

Once a linear predictor (LPC, [B1]) has been used to filter the original speech signal, the correlation between adjacent samples is removed: the LPC filter $1/A(z)$ models the average (short-term) spectrum of speech.

However, for voiced speech, the pitch introduces a long-term correlation. The fine structure of the speech spectrum is present in this residual signal. Due to pitch-induced quasi-periodicity, the residual signal still exhibits large variations. A pitch predictor can be used to remove the long-term correlation remaining in the residual signal. The simplest form of this pitch predictor filter (called the **long term predictor (LTP) filter**) is $B(z) = 1 - \beta z^{-M}$, where M is the pitch period and β a scalar gain. This filter subtracts from the current speech sample the value of a previous sample (at a distance of M samples) with a scaling factor of β . This procedure reduces the quasi-periodic behavior of the residual signal. A more generalized form of this LTP filter is $B(z) = 1 - \sum_i \beta_i z^{-M-i}$, called a multi-tap LTP filter.

In speech processing and coding, one of the main issues is to find the parameters of this LTP filter: the gain and lag values (β and M). These coefficients can be computed by evaluating the inter-correlation between frames of speech with different lag values and to find the maximum of these inter-correlation values; each maximum determines a lag value. Then the gain can be obtained by the normalization procedure (division of the power of the frame by the maximum inter-correlation found; sometimes an LTP gain of greater than unity can be found). This procedure is known as an open-loop search procedure as opposed to the closed-loop search found in some advanced CELP coders (adaptive codebook for long-term prediction).

Very often, since the frame length of speech coders is generally in the range 160–240 samples and the number of samples between two pitch periods is between 20 and 140, an LTP analysis is done on a subframe basis; this is also due to the fact that the pitch lag varies faster than the vocal tract (LPC filter). Moreover, the pitch lag may be not exactly equal to an entire number of samples, leading to the concept of fractional lags used in the LTP filter. The procedure to find this fractional lag must upsample the signal to be analyzed in order to find a fractional lag; for example, upsampling by a factor of 8 allows us to find a lag with a precision equal to one-eighth of the sampling period (generally for speech, 125 μ s). This fractional lag LTP is much more time-consuming, but it significantly improves the quality of decoded speech.

2.4.4 Vector quantization

Up to now, we have focused on sample-by-sample quantizers. With sample-by-sample, or scalar, quantization, each sample is mapped or rounded off to one discrete element of the codebook. This can be optimized by forming vectors of samples (or other quantities such as LPC or LSP coefficients) which can be quantized jointly in a single operation. Vector quantization is one of the most powerful tools used in modern speech and audio coders. In vector quantization, a block of M samples (or other items such as linear predictive coefficients) forms a vector that is mapped at predetermined points in M -dimensional space and portioned into cells; Figure 2.36 shows the case of bidimensional space.

For scalar quantization, quantization noise is added to each sample to be encoded and decoded; on the other hand, for vector quantization, the noise is concentrated around

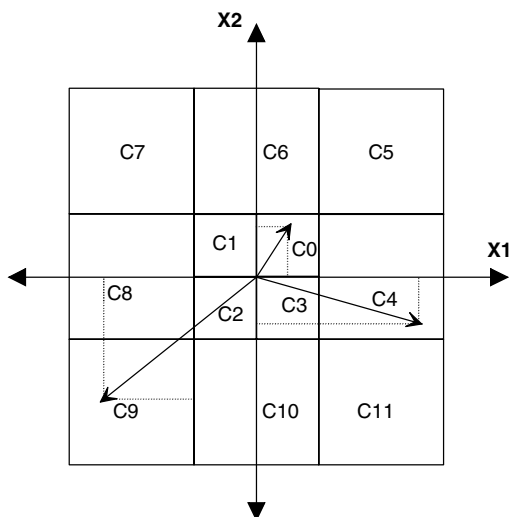


Figure 2.36 Vector quantization. Two-dimensional space for a vector quantizer. Vectors of components X_1 and X_2 are localized in cells C_0 to C_{11} ; the index of the cell is transmitted at the decoder.

the selected vector and correlated for all components. Generally, vector quantization is more efficient than scalar quantization because the codebook can be optimized to use this correlation. For example, in vocoders or source speech coders, such as the LPC 10, independent scalar quantization of the ten LPC coefficients requires about 50 bits per frame (20–30 ms), but vector quantization needs only 25 bits per frame for the same subjective and perceived quality.

This is a significant improvement, but the counterpart is that vector quantization requires much more processing power and is also more sensitive to transmission errors than scalar quantization: an error on one decoded vector impacts all the individual elements of the vector.

There are several types of vector quantization procedures, such as binary, gain shape, split, etc.: in each case the design and optimization of the codebook is of prime importance. Optimizing space partitioning and finding the best vector representatives requires a very large database so that the codebook can be optimized. Distortion measures correlated with human perception and some subjective tests are sometimes required to choose the best codebook.

2.4.5 Entropy coding

This technique is not specific to speech and audio coders, it is also used for most video coders and fax, as well as many file compression tools. The principle of entropy encoding is to map the parameters to be transmitted (e.g., a bit pattern) to code words of variable length, and to use shorter (with a minimum number of bits) **code words** to represent more frequently transmitted parameter values and longer code words for the least used. Huffman codes and RLC (running length code) are some representatives of such codes.

Huffman coding [A21] represents an object with a number of bits that is smaller for objects with larger probabilities. The algorithm builds a binary tree iteratively by first assembling the two objects with the lowest probabilities ω_1 and ω_2 in a node associated with weight $\omega_1 + \omega_2$. The object with the smallest probability ω_1 is located to the left of the node. The new node with weight $\omega_1 + \omega_2$ is added to the collection of objects and the algorithm is restarted (Figure 2.37).

Such an entropy-coding scheme can be placed after a classical speech or audio coder on the bitstream to be transmitted. No additional framing information is required in the encoded bitstream (prefix condition code).

2.5 Waveform coders. ADPCM ITU-T G.726

Waveforms coders are also called temporal speech coders; they rely on a time domain and sample-by-sample approach. Such coders use the correlation between continuous samples of speech and are based on adaptive quantizers and adaptive (generally backward) predictors. They are very efficient in the range 40–24 kbit/s, but quality degrades quickly (around 16 kbit/s).

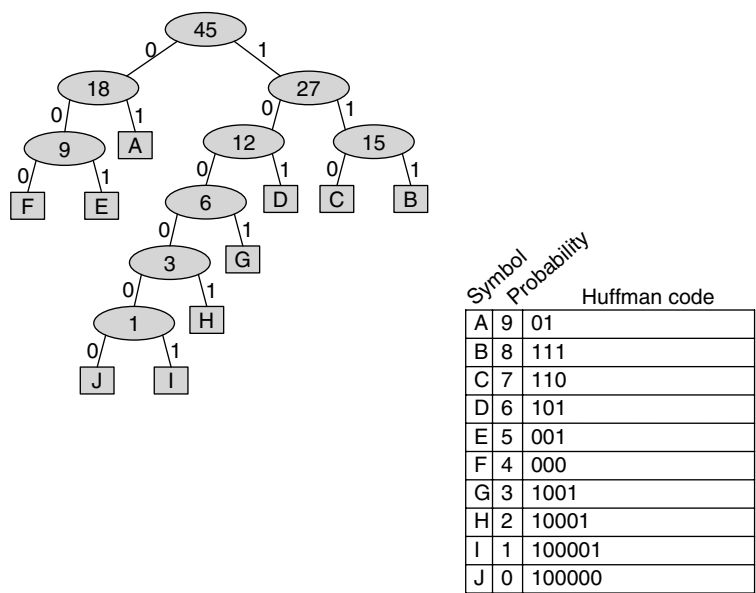


Figure 2.37 Principle of Huffman encoding.

The most widely used standardized waveform coder (excluding ITU-T G.711) is the ADPCM ITU-T G.726 [A13] speech coder which operates at 16, 24, 32,⁴ or 40 kbit/s. The 32-kbit version is used in DECT (digital enhanced cordless telecommunication) wireless phones in Europe, in PHS (personal handy-phone system) phones in Japan, or in **DCME** (**digital circuit multiplication equipment**) device on submarine cables.

ADPCM stands for adaptive differential pulse code modulation; the name itself explains the basic principle of the G.726 speech coder (see Figure 2.38).

The adaptive quantizer is a one-word memory type (or Jayant type) as described in Section 2.2. The adaptive predictor is a mixed structure with six zeros and two poles; it processes the reconstructed signal using a two-coefficient adaptive filter (the poles) and the decoded difference signal using six-coefficient adaptive filter (the zeros).

The basic scheme (Figure 2.38) does not include some useful features such as a dynamic switch for selecting alternative strategies when voice band modem signals are detected in order to allow the ADPCM coder to adapt to modem signals. One of the major drawbacks of coding schemes that reduce the bitrate and rely on the speech characteristics is that they fail for non-speech signals: voice band modem signals are completely synthetic and do not fit the prediction and adaptation procedures tailored for speech signals. The dynamic strategy switch allows transmission of a 9,600-bit/s modem signal for 32 kbit/s ADPCM and a 144,00-bit/s signal (V.33) for 40 kbit/s ADPCM.

The G.726 and its predecessor G.721, standardized in 1984, were the first bit reduction schemes used for civilian telecommunications. It is still one of the most widely used coders

⁴ The old ITU-T G.721 speech coder used in voice storage systems is equivalent to G.726 at 32 kbit/s.

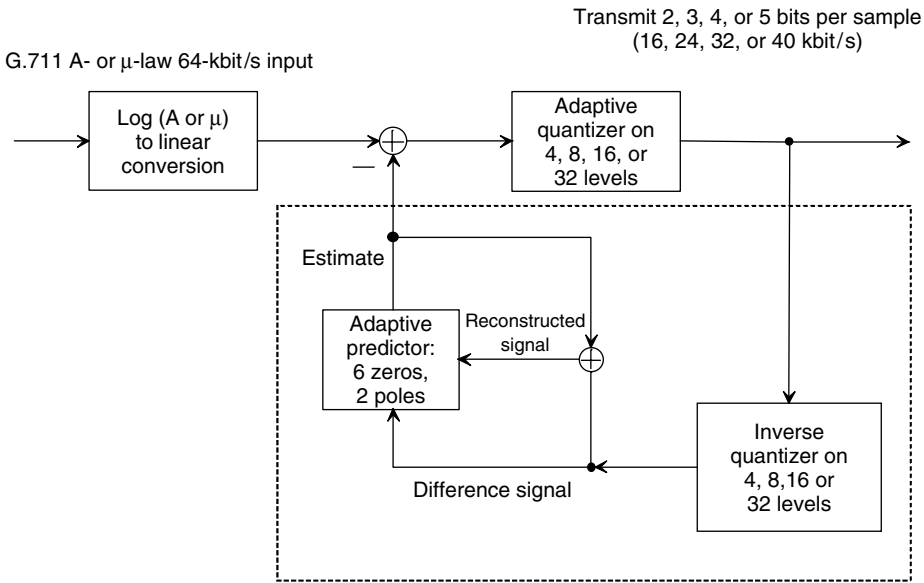


Figure 2.38 ITU-T G.726 ADPCM (16, 24, 32, or 40 kbit/s) basic scheme. The distant decoder is equivalent to the local decoder inside the dotted box.

over terrestrial and submarine cables, in combination with speech interpolation. Speech interpolation relies in the statistical distribution of speech activity on a large number of affluent speech links. In a conventional conversation, each speaker is active less than 50% of the time on each side of the transmission link; the corresponding bandwidth can be used to transmit another voice channel. This becomes even easier with VoIP by using discontinuous speech transmission. Using speech interpolation and ADPCM (G.726 ADPCM at 32 kbit/s) a DCME can achieve a compression gain of 4 to 5.

Due to the symmetrical form of the encoder and decoder (they only differ by their quantizer procedures) of ADPCM, both use a similar processing power of approximately 5 MIPS (16-bit fixed point). Despite this low complexity, the speech quality of G.726 is very good (above 24 kbit/s), as indicated in Figure 2.39.

One interesting feature of the ADPCM coder is its relative immunity to bit errors compared with PCM. As shown in Figure 2.40, there is a significant difference for a **BER (bit error rate)** of 10^{-3} in favor of the ADPCM coder. There are two main reasons: PCM is very sensitive to an error on the sign bit, and ADPCM combines the state variables of the algorithm and consequently, it becomes more robust. This is a typical difference that disappears in VoIP, as errors do not occur as isolated bit errors, but result in complete frame loss (as a packet is rejected if the checksum is wrong).

Although ADPCM coders are not based on a frame-by-frame analysis and speech-coding procedure, in some circumstances (e.g., for voice over IP), ADPCM codes may be transmitted in a packet form. One packet assembles several codes (typically 10–30 ms), each corresponding to one unique sample. In the case of packet loss or ‘frame’ errors, the situation with PCM or ADPCM can be disastrous compared with hybrid or ABS (analysis

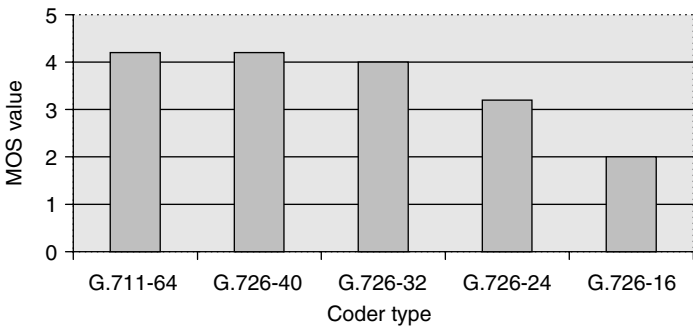


Figure 2.39 Typical MOS scores of common voice coders.

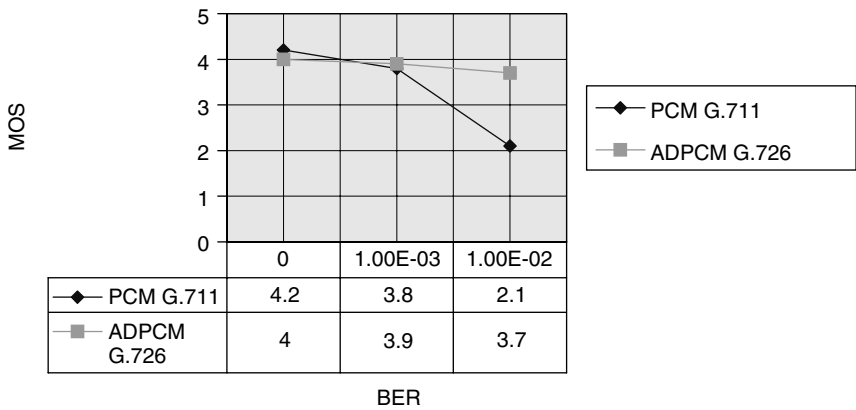


Figure 2.40 Comparison between the BER sensitivity of ADPCM and that of PCM.

by Synthesis) speech coders which can rely on the last valid received parameters (such as LPC and LTP coefficients) to rebuild an approximation of the complete form of the signal for the lost frame. For ADPCM, the loss of many code words breaks the pursuit of the distant decoder against the local decoder, and a long time (250–500 ms) is needed to recover a stable state.

2.5.1 **Coder specification . . . from digital test sequences to C code**

The G.726 (or more exactly its predecessor G.721) was the first speech coder whose specification included an exhaustive set of digital test vectors. This is required to insure interoperability between equipment built by different manufacturers.

The set of test vectors was required because G.726 did not include a C reference, but an extensive documentation on a fixed point implementation. The fixed point implementation is a strong requirement for economical implementations in DSPs (digital signal processors) or for dedicated VLSI. The ITU-T recommendation includes the exact format (fixed

point) of the variables, constants, state variables, and tables used in the algorithm. It also describes most of the operations required by the algorithm, such as addition, subtraction, fixed point multiplication, and control of possible saturation (which may happen frequently in fixed point arithmetic).

The lack of a reference code was a problem, and later ITU-T introduced reference fixed point ANSI C code for new coders, where all mathematical operations (add, multiply, etc.) are fully specified (this reference implementation is called basic op for 'basic operation'). Today, an ANSI C reference code is the main part of the recommendation of many speech coders, such as ITU-T G.723.1 or G.729. Test vectors are also provided to facilitate the verification of compliance to the standard. These test vectors are designed to provide an extensive coverage of the algorithms used in the implementation for both coding and decoding.

Floating point versions of some algorithms are also useful to improve the quality of implementations in PCs and workstations, and eliminate interoperability issues between fixed point and floating point implementations (e.g., a VoIP gateway using fixed point DSPs and a client PC software using native floating point arithmetic for efficiency). Specific test vectors also help verify the interoperability between different floating point implementations, due to the variety of floating point number representations.

2.5.2 Embedded version of the G.726 ADPCM coder G.727

One desirable feature of a coder is the ability to dynamically adjust coder properties to the instantaneous conditions of transmission channels. This requires some synchronization between the encoder and the decoder when the encoding properties change.

ADPCM can dynamically switch between one of the multiple encoding rates. In this case **embedded** means that a core quantizer is used for the fundamental operations of the coder, and additional quantification bits are allocated to an 'enhancement' quantizer. The scale used by the core quantizer is subdivided to form the scale of the enhancement quantizer. In order to ensure that synchronization is not lost even if some 'enhancement' bits are changed or even not transmitted, the decoder synchronization state is based only on the bits from the 'core' quantizer. This makes it possible to steal or remove some bits in the transmitted code words without desynchronizing the distant decoder, allowing a 'graceful' degradation in the decoded speech without requiring external signaling transmission means. This feature is very useful in applications, such as DCME or PCME (packet circuit multiplication equipment), in overload situations (too many active channels present at the same time) or for 'in band' signaling or 'in band' data transmission.

This concept is used in the embedded version of the G.726 (ITU-T, G.727 recommendation [A1]). In order to insure that the distant decoder tracks the local decoder correctly and due to the fact that this distant decoder may receive code words with robbed bits, the inner loop of prediction relies on the inverse core version of the quantizer:

- On the encoder side, the difference signal is encoded with the full number of steps of the enhanced quantizer, but bits in excess in the enhanced version are masked before feeding the inverse core quantizer.

- On the decoder side, the excess bits of the received code word are masked in order to feed the core inverse quantizer which is used in the prediction and reconstruction inner loop, but the entire received code word enters the enhanced adaptive quantizer, whose output is used to build the final output.

If there are no robbed bits, the output quality is enhanced, but is not as good as if the enhanced version of the quantizer had been used in the inner loop of the encoder and decoder, using all available quantization bits: that is the price to pay for the ‘embedded’ feature.

Figures 2.41 and 2.42 illustrate the G.727 concept.

2.5.3 Wide-band speech coding using a waveform-type coder

2.5.3.1 G.722

In the world of telephony, G.711 is frequently used as ‘the’ reference of voice quality, ignoring the fact that G.711 encodes only the 300–3,400-Hz band. The truth is that it is very difficult to go beyond G.711 quality in traditional telephone networks, because most of the components, from switches to transmission links, assume a G.711 signal (with the exception of transparent ISDN, which is available in some countries).

This is no longer true with voice over IP, where virtually any encoding scheme can be used end to end on the IP network. There are strong requirements to offer a better speech

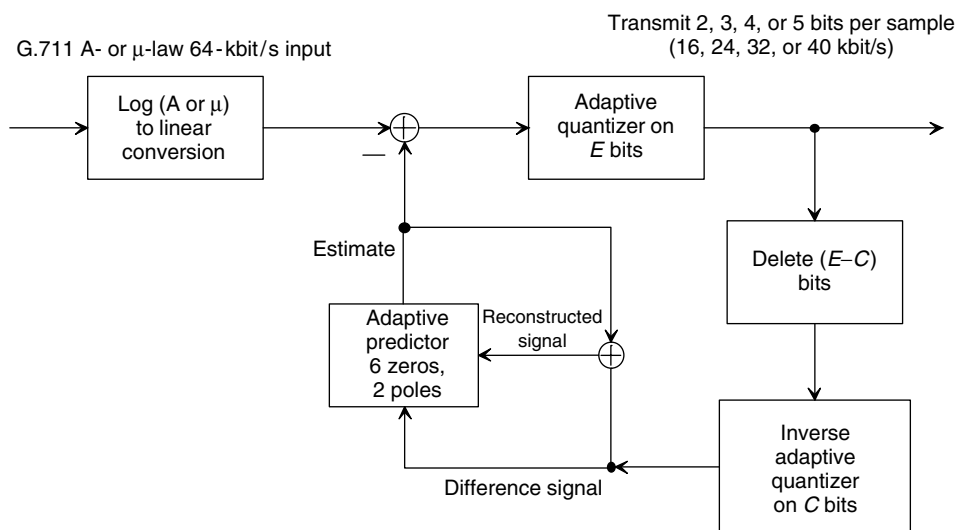


Figure 2.41 G.727 encoder. ITU-T G.727 embedded ADPCM (16, 24, 32, or 40 kbit/s) basic scheme. G.727 is characterized by the enhance and core pairs (E , C) values for quantizers. C can have 2, 3, or 4 as values and E 2, 3, 4, or 5.

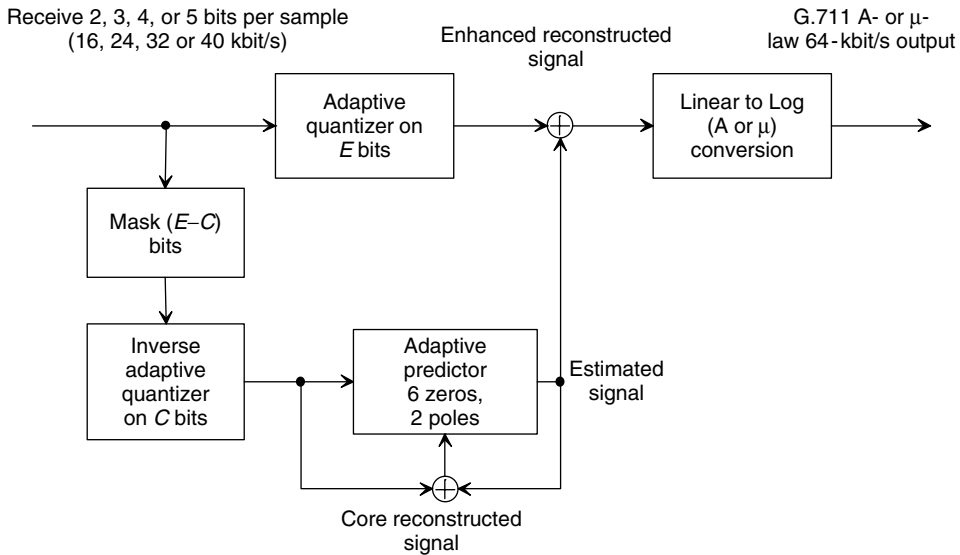


Figure 2.42 G.727 decoder. ITU-T G.727 embedded ADPCM (16, 24, 32, or 40 kbit/s) decoder basic scheme.

and audio quality for videoconference and audioconference systems [A4, A14]. While most coders focus on providing an acceptable voice quality for the lowest possible bitrate, it is also possible to increase the audio quality as much as possible for a given bitrate.

Scientists and engineers were well aware of the possibilities of waveform ADPCM speech coders to reduce the bitrate by a factor of about 0.5 and naturally tried to use a similar technique to encode wide-band speech. Wide band refers to a transmitted frequency band of 50 Hz up to 7,000 Hz compared with the traditional telephony bandwidth (300 Hz to 3,400 Hz).

G.722 was proposed by France Telecom and NTT, and adopted by ITU in 1988. The fundamental idea is to split the band to be transmitted in two subbands: a lower subband spanning from 0 Hz to 4,000 Hz and a higher subband spanning from 4,000 Hz to 8,000 Hz. Then, after a subsampling procedure reducing the sampling frequency from the original 16 kHz down to 8 kHz, two ‘classical’ ADPCM encoders can be applied to reduce the bitrate. Subsampling is possible because subband frequency filtering has eliminated the aliasing effect.

Subband separation uses a pair of quadratic mirror filters. QMF filters are the precursors of the filter bank theory used for psychoacoustic coders.⁵ In many ways the wide-band ITU-T G.722 speech and audio coder is a precursor of the more recent psychoacoustic audio coders: the splitting of the original band into two subbands and the allocation of more bits in the lower subband optimizes the efficiency of the prediction that the most sensitive frequency band performs noise quantization masking. The energy of speech

⁵ These filter banks (with a number of bands from 32 up to 1,024) are intensively used in audio bitrate reduction (ISO-MPEG, AAC, Dolby Digital, etc. [A14]).

signals is more concentrated in the lower subband, and allocating more bits in this subband increases the quality of decoded speech.

G.722 encodes a wide-band signal into a bitstream of 64 kbit/s (the basic PCM bitrate). In the lower subband, 6 bits are used for the adaptive quantizer with an embedded characteristic: the core quantizer uses 4 bits and the enhanced version uses 6 bits. This scheme is very similar to the one found in the embedded version (G.727). This allows the system to steal some bits for signaling purposes (framing with H.221) and to transmit some ancillary data. The decoder should be signaled the mode of operation (64, 56, or 48 kbit/s), although some realizations do not signal the mode and permanently use the full 6 bits. In the higher subband, a 2-bit adaptive quantizer (nonembedded) is used producing a 16-kbit/s bitrate (much lower than the 48 kbit/s used for the lower subband which is perceptually more important).

The coding scheme of G.722 is illustrated in Figure 2.43, and the decoding principle of G.722 is shown on Figure 2.44.

The ITU-T G.722 wide-band speech coder is commonly used in teleconference systems adhering to the H.320 recommendation. The quality is quite good for speech and music at 64 kbit/s and 56 kbit/s (MOS of 4.3 and 4 compared with an original with the same bandwidth rated at 4.3). As there is no specific ‘production model’ (e.g., for speech) in that waveform coder, samples of music are correctly encoded.⁶ When used at 48 kbit/s, reproduced speech becomes more noisy (due to the 4-bit quantizer in the lower subband).

G.722 shares with other waveform ADPCM coder types a relative immunity to bit errors and is more robust than a direct PCM stream. The low-delay characteristic of the G722 is also a major advantage compared with more recent frame-based audio coding schemes. All

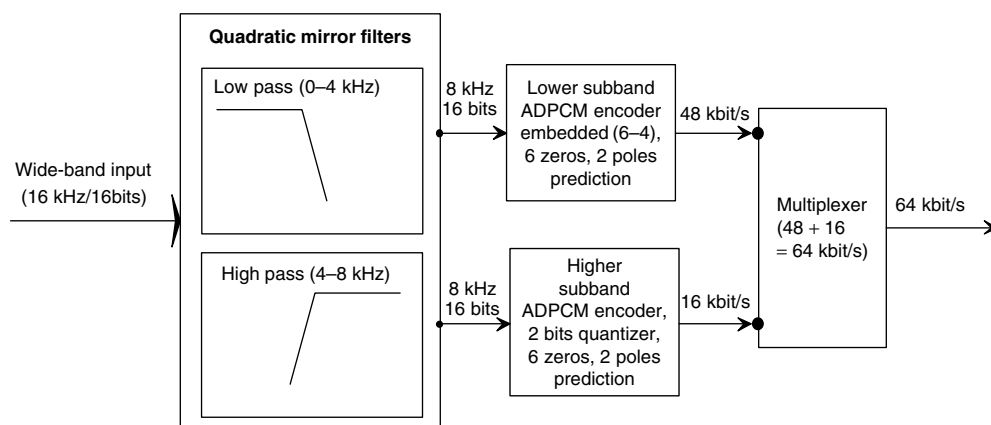


Figure 2.43 G.722 encoder. ITU-T G.722 wide-band encoder, subband ADPCM with QMF filter (48-kbit/s embedded ADPCM in lower subband and 16-kbit/s ADPCM in higher subband).

⁶ Although a bit allocation more favorable to the upper subband, such as 5 bits in the lower band and 3 bits in the higher band, has performed better on many music samples. As the main applications were for teleconference systems, preference was given to the fixed bit allocation strategy that favors speech quality.

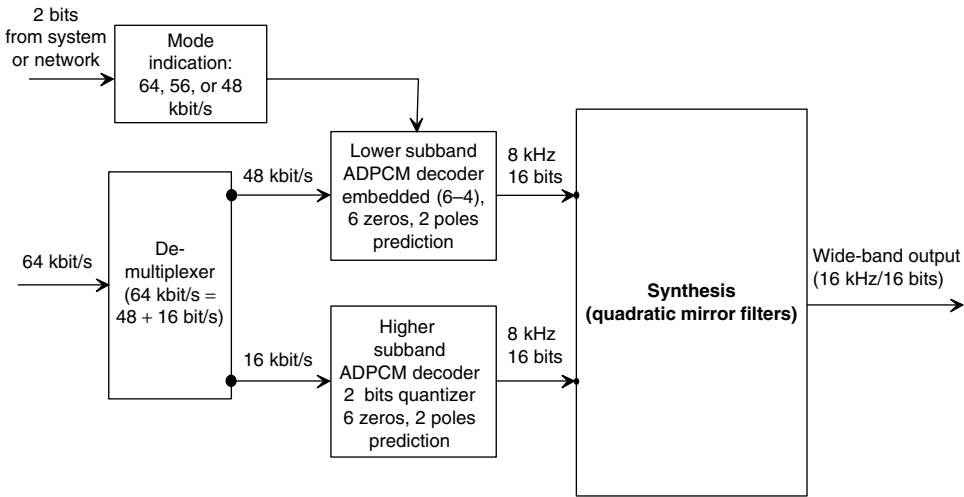


Figure 2.44 G.722 decoder. ITU-T G.722 wide-band decoder, subband ADPCM with QMF filter (48-kbit/s embedded ADPCM in lower subband and 16-kbit/s ADPCM in higher subband).

the waveform coders, such as ADPCM and PCM, have very low algorithmic delay ranging from three to four samples (300–500 μ s with an 8-kHz sampling frequency). In the case of G.722, QMF analysis and synthesis filters add a delay of about 3 ms. The resulting total delay remains excellent and ensures good interactivity for teleconference systems.

G.722 is one of the coders recommended for use in H.323 systems and is available in several commercial implementations.

2.5.3.2 G.722.1

One of the limitations of G.722 is that it cannot be used below 48 kbit/s. The more recent G.722.1 (September 1999) can encode a wide-band signal with a bitrate of 24 kbit/s or 32 kbit/s (a proprietary Pictoretel version exists at 16 kbit/s, called Siren™).

G.722.1 works on frames of 40 ms (640 samples sampled at 16 kHz) with an overlap of 20 ms. On each frame of 40 ms, it multiplies the signal by a sinusoid (therefore the amplitude of the signal at both ends of the frame converges to 0), then performs a **discrete cosine transform (DCT)**. The whole operation is called the **modulated lapped transform (MLT)**; it is illustrated in Figure 2.45.

The result is the encoding of a 20-ms frame using 480 bits at 24 kbit/s and 640 bits at 32 kbit/s. Each frame is encoded independently; there is no state at the receiver. This interesting property prevents frame de-synchronization in the case of frame erasures, typically on VoIP systems. The resulting spectrum is analysed in 16 regions, in order to determine which region is more important (perception model) for the listener. Each frequency region is then quantized and vector-encoded using a Huffman encoding. The more important frequency regions (from a perception point of view) are allocated more bits than the less important frequency regions.

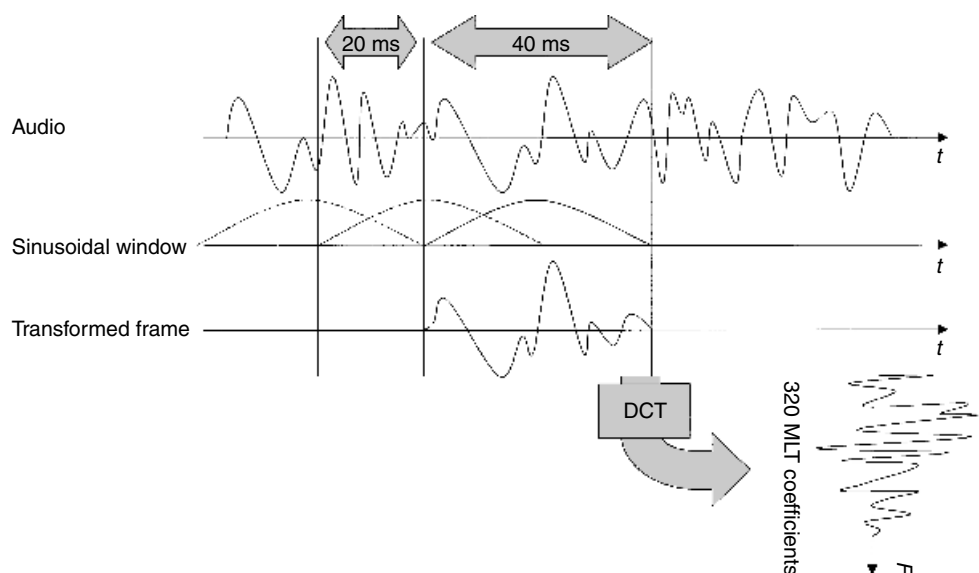


Figure 2.45 Modulated lapped transform used in G.722.1.

This coder uses about 14 MIPS (3% of a Pentium PIII-600) and is supported in the Windows XP® Messenger softphone under the proprietary 16-kbit/s version (Siren™).

2.6 Hybrids and analysis by synthesis (ABS) speech coders

2.6.1 Principle

In previous sections we have studied two types of coders:

- Waveform coders that remove the inter-sample correlation by using linear prediction. The differential coding scheme used with adaptive quantizers gives good performances with a bitrate between 32 kbit/s and 24 kbit/s.
- Linear predictive coders (or vocoders) use a simple model of speech production (voiced or unvoiced types), modeled by a slowly variable filter (updated on a 20–30-ms frame basis) which shapes the spectrum of the decoded speech. LPC coders are used for very low-bitrate speech coders (1,200–2,400 bit/s), but speech quality is low ('synthetic' quality).

Hybrids and analysis by synthesis (ABS) coders combine the best of the two approaches in order to build efficient coding schemes using a bitrate between 6 kbit/s and 16 kbit/s.

ABS coders use a frame of samples to compute the LPC filter coefficients modeling the vocal tract, as well as a long-term predictive (LTP) filter that removes the 'pitch'

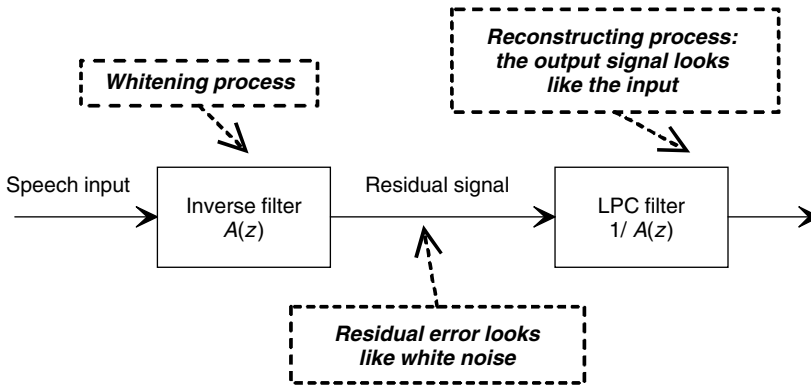


Figure 2.46 The residual error signal after filtering speech by the inverse filter.

correlation. Both LPC and LTP coefficients are encoded (vector quantization is frequently used) and transmitted. But, unlike LPC coders that need to classify the speech type between ‘voiced’ or ‘unvoiced’ and transmit this information, hybrids and ABS speech coders avoid such classification by finding some means of encoding the residual error signal between the inverse LPC/LTP filter (see Figure 2.46) and the original signal.

In **residual excited linear predictive (RELP)** speech coders, the residual signal is fed to a low-pass filter and the resulting signal is classically encoded in PCM form. RELP coders give good results around 10 kbit/s by transmitting the LPC/LTP coefficients and the encoded residual signal. RELP speech coders do not attempt to remove the pitch contribution (they do not apply a dedicated, long-term predictive filter).

Analysis by synthesis (ABS) speech coders use a slightly different method. Instead of encoding the residual error signal (a method focused on the ‘output’), they attempt to compute which excitation input signal to the inverse LPC/LTP filter will result in a decoded speech signal as close as possible to the original signal. The excitation parameters are transmitted to the decoder.

The ABS principle is shown in Figure 2.47.

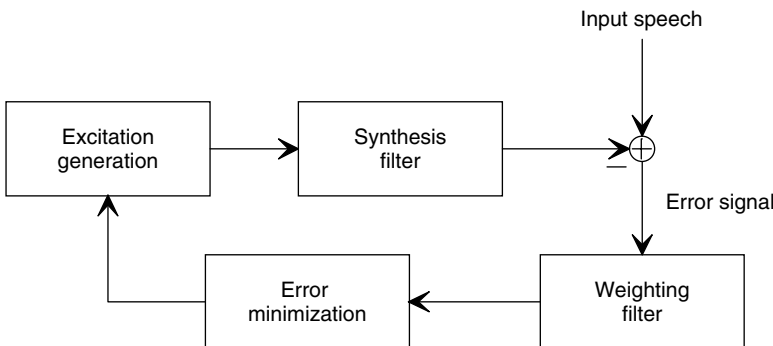


Figure 2.47 Analysis by ABS encoder principle.

The ABS speech coder optimization algorithm finds the ‘best’ vector of configuration parameters for the excitation generator. This best vector is searched by using an error minimization loop based on the perceptual error between the original speech and the synthesized signal. The synthesis filter is a cascade of the inverse LPC filter and inverse LTP filter. ABS coders can be considered both as synthesis filters (LPC/LTP approach) and waveform coders (minimization of a waveform error); they are also called hybrid waveform speech coders. An ABS decoder is very simple as shown on Figure 2.48.

2.6.2 The GSM full-rate RPE-LTP speech coder (GSM 06.10)

The most widely used ABS speech coder is the GSM full-rate codec, standardized by the ETSI in 1988 for the cellular digital mobile system. This coding scheme was proposed by PKI, IBM France, and France Telecom. It uses **regular pulse excitation (RPE)** with **long-term prediction (LTP)**, or **RPE-LTP**, at a bitrate of 13 kbit/s [A16]. The GSM coder feeds the inverse ABS filter with an excitation signal that is optimized to minimize the error signal. GSM uses a series of regular pulses, special cases of ‘multi-pulse’ excitation signals that will be studied later. The choice of RPE to ‘encode’ the residual signal allows for lower complexity implementation compared with general multi-pulse optimization.

In the GSM full-rate coder, the signal is first buffered into a frame of 20 ms (160 samples), then classical LPC analysis finds the eight coefficients that model the vocal tract. These coefficients (also called *parcours* for *partial correlation*) are encoded and transmitted in the bitstream. The entire input buffer is inverse-filtered by the inverse LPC filter, resulting in 160 residual (LPC) samples.

These 160 residual samples are subdivided in four subframes of 40 samples. In each subframe, the algorithm seeks the optimal LTP filter gain and delay. The LTP filter was described in Section 2.4.3. The use of subframes reflects the fact that pitch (which is between 75 Hz and 400 Hz depending on the age and gender of the speaker) varies more rapidly than vocal tract characteristics. The LTP lag and gain are encoded and transmitted for each subframe.

The LTP contribution is then subtracted from the residual signal for each subframe of 40 samples.

This difference signal is then encoded using the RPE procedure, which splits the original 40 samples of the difference signal into four subseries of samples:

- The first starts with the value of sample index 0, then picks one sample value out of 4, from index 3 up to index 36.
- The second starts with index 1, then picks one sample value out of 4, from index 4 up to 37.

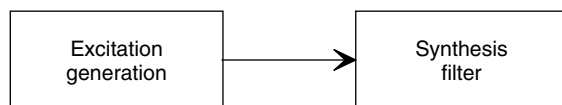


Figure 2.48 ABS decoder principle.

- The third starts with index 2, then picks one sample value out of 4, from index 5 up to 38.
- The last starts with index 3, then picks one sample value out of 4, from index 6 up to the last index of the subframe.

Of the four series, the one that best approaches the original 40 residual samples is chosen; two bits per subframe are required to indicate the choice to the receiver. The maximum energy of the samples in the selected subsequences is also encoded, using 6 bits. All the samples of the subsequence are normalized by this quantized energy, then scalar-quantized with 3 bits. Each series consists of a subsampled process which is a hard low-pass filter with a frequency cutting around 1,300 Hz. This privileges the male voice over female or child voices.

The bit allocation for one frame of the GSM RPE-LTP speech coder is given in Table 2.8. The GSM RPE-LTP encoder principle is shown in Figure 2.49 and the decoder on Figure 2.50.

Although the RPE-LTP yields a speech quality slightly lower than standard telephony it is well suited for mobile communications systems because it resists transmission errors rather well. The MOS figure of the RPE-LTP is around 3.8 compared with 4.2 of the G.711 PCM.

The ETSI 06–10 GSM RPE-LTP recommendation includes a detailed description in fixed point arithmetics relying on the use of ‘basic operators’. Digital test sequences are also given to check conformity to the standard. Although some floating versions of this standard exist and are used in VoIP software, some subtle issues may arise in interoperability with the genuine fixed point version.

In addition to basic speech encoding, a **VAD (voice activity detection)**, **DTX (discontinuous transmission)**, and **CNG (comfort noise generation)** scheme was added to the coder. VAD detects whether valid speech is present and otherwise transmits (less frequently) parameters containing the noise information. In the case of GSM, these parameters are based on the LPC parameters and on the energy of the noise. They are packed in a SID (silence description) frame which is sent every 80 ms (four frames compared with

Table 2.8 GSM full-rate bit allocation

<i>RPE-LTP frame length = 160 samples = 20 ms</i>	
Vocal tract: LPC coefficients; 8 parcors = 36 bits	36
<i>Subframe length = 40 samples = 5 ms (4 subframes)</i>	
Grid selection = 2 bits	8
Maximum of energy of selected series = 6 bits	24
Scalar quantization of 13 samples = $13 * 3 = 39$ bits	156
LTP lag = 7 bits	28
LTP gain = 2 bits	8
Total	260
<i>Bit rate = $260 / 20 \text{ ms} = 13 \text{ kbit/s}$</i>	

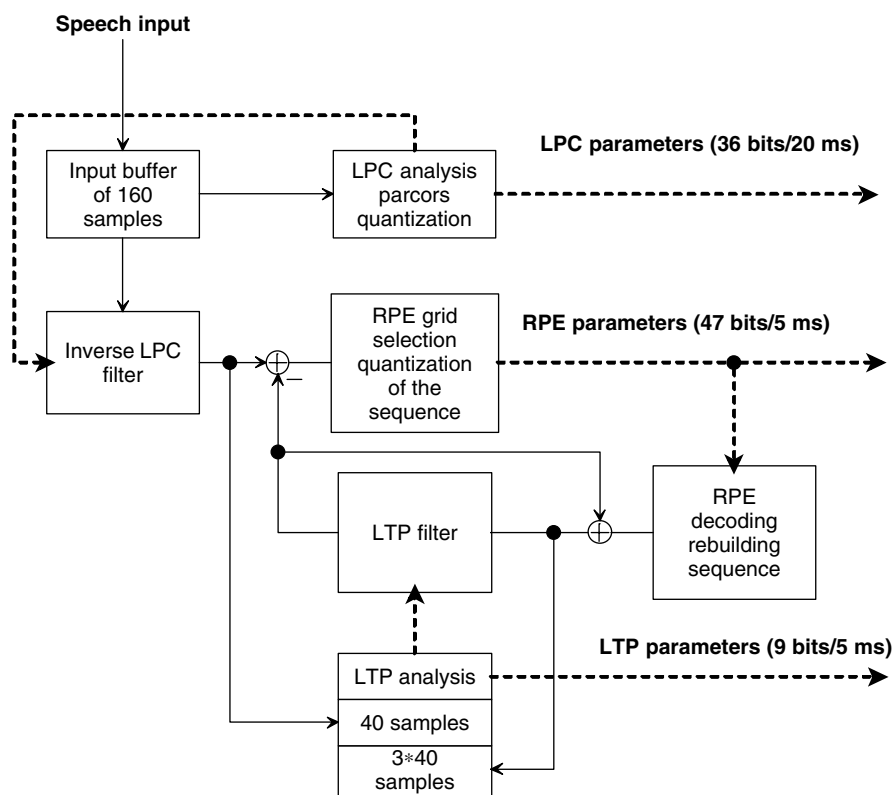


Figure 2.49 Basic principle of the RPE-LTP full-rate (13-kbit/s) GSM speech coder.

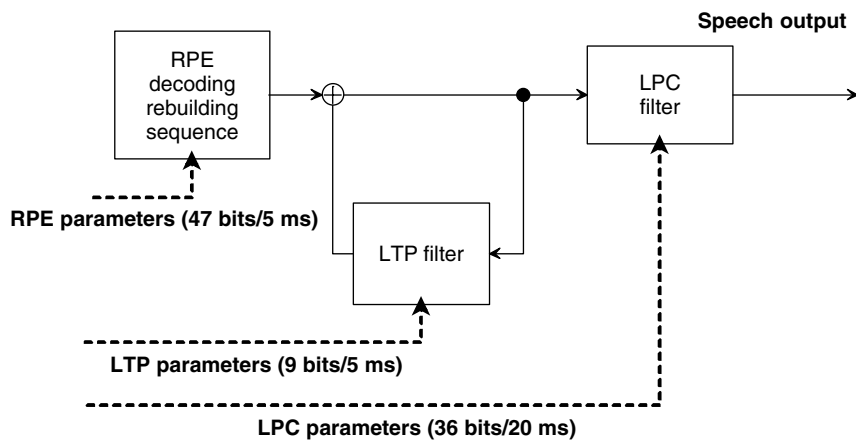


Figure 2.50 Basic principle of the RPE-LTP full-rate (13-kbit/s) GSM speech decoder.

the 20-ms speech frame). It must be pointed out that the design of a good and efficient VAD algorithm is almost as complex as the design of good speech coder.

The GSM 6.10 coder reflects the constraints of the processing power commonly available in 1988; it is being progressively replaced by GSM 6.60.

The GSM 6.60 coder is based on the ACELP technology proposed by Nokia and the University of Sherbrooke. It only uses 12.2 kbits/s (less than the 13 kbit/s of GSM 6.10, leaving some capacity for error protection). When there are no errors on the transmission channel, the voice quality is equivalent to G.726 at 32 kbit/s (toll quality).

2.7 Codebook-excited linear predictive (CELP) coders

CELP coders are in essence linear predictive coders equipped with an ABS search procedure. They were invented in the 1980s by Bell Labs (under the supervision of B.S. Atal and M.R. Schroöder). As we have already seen, once the short-term correlation in the signal has been removed by the LPC filter and the long-term correlation (or pitch contribution) has been removed by the LTP filter, the quality of reproduction depends essentially on the selection of an optimal excitation signal.

A possible choice is a multi-pulse excitation signal. The position and amplitude of each pulse are searched iteratively using an ABS algorithm. The main pulse position is searched first, then the algorithm locates the optimal second pulse, and so on. The coder bitstream must encode the position and amplitude of each pulse modeling the excitation. Note that the regular pulse solution used in the GSM full rate is a particular case of the multi-pulse excitation signal, which significantly decreases the computing power required for computation of a general multi-pulse excitation signal.

The optimization of a multi-pulse signal is very complex in general, because the number of candidate vectors is very large. In CELP coders, a codebook based on vector quantization is built, trained, and optimized off-line on a large ‘speech’ database. Only these vectors are used as candidates for the excitation generator that feeds the LTP and LPC synthesis filters. The excitation signal (index in the codebook and value of gain) that best approximates the original speech input signal is selected according to a perceptual error criterion.

The role of the perceptual filter is to redistribute noise in frequency ranges where it will be less audible due to the higher energy of the main signal: the noise will be masked by the signal itself. Significant improvements of the subjective quality [A3] are observed when using this **perceptual weighting filter**. The filter $W(z) = A(z)/A(z/\gamma)$, with a bandwidth expansion coefficient γ less than 1, forces the noise to be reinforced in the neighborhood of the formants and to be lowered in the region where the signal is weak. Although absolute noise power is generally increased, listeners generally prefer this situation.

One big issue with CELP coders is the difficulty of finding the best index and associated gain in the codebook, as the codebook is very large. For a long time, this has been a barrier to practical implementation in real time. Algorithmic simplifications brought to the initial design (efficient codebook search or algebraic codebooks) and the growth

of available MIPS (million instructions per second) in modern DSPs have finally made it possible to implement CELP coders in real time.

The basic scheme of a CELP coder is shown on Figure 2.51.

LPCs are first computed and quantized for an entire frame of speech (10–30 ms). Vector quantization and line spectrum pairs are increasingly used due to their efficiency. LTP lag and gain are searched and quantized on a subframe basis as well as the codebook index and associated gain G_i .

The decoder is much less complex than the encoder (there is no ABS search procedure) and can include an optional post-filter as shown in Figure 2.52.

In order to improve perceived quality, the post-filter aims at reducing the noise level in frequency bands located between the maxima of the spectrum (located near the harmonics). A typical implementation is a short-term post-filter which is derived from LPCs in a similar way as the perceptual weighting filter in the encoder. Modern post-filters can also include a long-term prediction post-filter and a tilt compensation post-filter. The introduction of the post-filter can significantly increase the MOS rating of CELP decoders; nevertheless, it may affect the fidelity of decoded speech if its action is exaggerated.

The basic scheme for a CELP encoder relies on an open-loop search for the long-term correlation coefficients of the LTP filter. A more advanced implementation refines this procedure by first conducting an open-loop search for an LTP lag, then testing fractional lags in the neighborhood of this initial lag in an adaptive codebook. The chosen value is selected by an **ABS-MSE (mean square error)** procedure.

The remaining components (called innovations) of the residual signal are nonpredictable, and a best matching excitation vector is searched in another codebook, called the stochastic codebook. The design of the stochastic codebook, which models samples that more or less resemble noise, is complex. There are two main approaches. The first

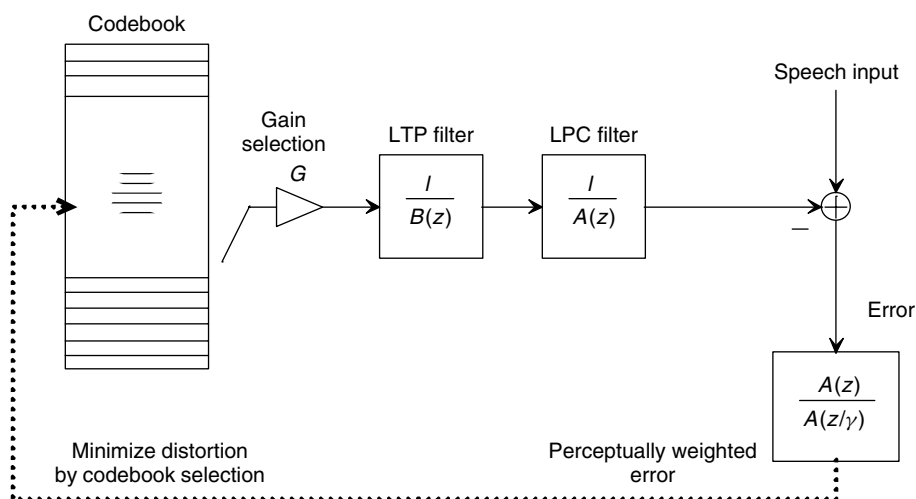


Figure 2.51 Basic concept of a CELP coding algorithm. The quantized LTP and LPC parameters are transmitted on a frame basis. The quantized gain G and the codebook index are transmitted (sometimes on a subframe basis).

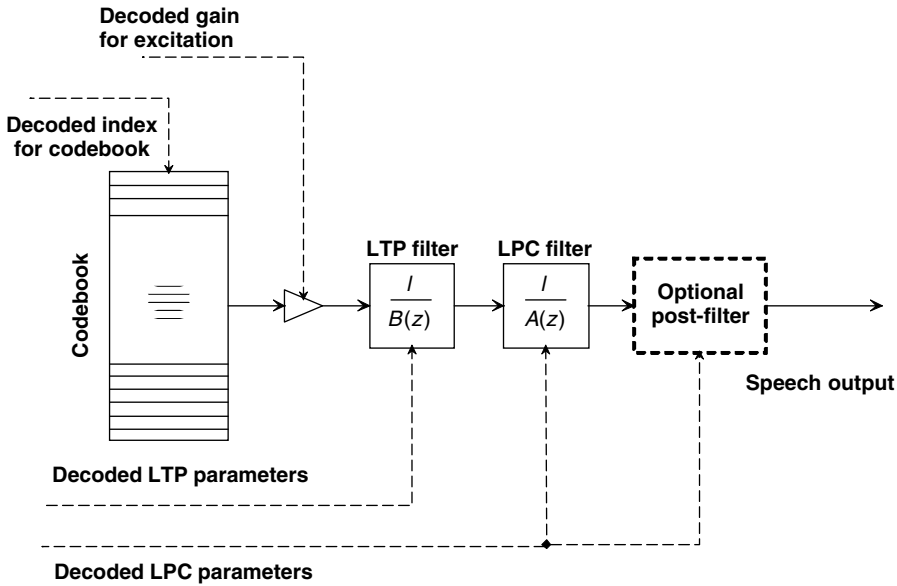


Figure 2.52 Basic concept of a CELP decoding algorithm.

is to build the codebook before the execution phase of the encoder by using training and optimization on large speech databases. The second is based on a predetermined set of patterns, which are combined, resulting in the optimal excitation vector (see Section 2.7.1 on G.729 for an example). The optimal combination is computed during the ABS mean square error procedure (e.g., selection of the pulse location and associated gain). The latter method is used, for example, in ACELP (algebraic CELP) or MP-MLQ (multipulse maximum likelihood quantization).

The algorithm is therefore based on a closed-loop search in two codebooks:

- The adaptive codebook which is devoted to long-term prediction.
- The stochastic codebook which deals with those components in the residual signal that are nonpredictable.

The closed-loop search selects four parameters:

- (1) An index in the stochastic codebook.
- (2) An optimal gain corresponding to the index selected in the stochastic codebook.
- (3) A lag (integer or fractional) in the adaptive codebook.
- (4) An optimal gain corresponding to the selected lag value.

The optimal excitation search for the LPC synthesis filter is therefore modified as shown in Figure 2.53.

tenth-order linear prediction filters. Due to the short frame length of 10 ms, LSPs (line spectral pairs) are quantized by using a fourth-order moving average (MA) prediction. The residue of linear prediction is quantized by an efficient two-stage vector quantization procedure (the CS used in the coder name refers to this). An open-loop search for the lag of the LTP analysis is made to select the initialization value for the closed-loop search in each subframe. Pitch predictor gain is close to unity, but the fixed codebook gain varies much more. This gain is estimated by a fourth-order MA gain predictor with fixed coefficients, from the sequence of previous, fixed codebook excitation vectors. This is the main difference between the G.729 encoder scheme and the one described on Figure 2.53; this gain predictor appears in the decoder scheme in Figure 2.54.

The lag and gain of the LTP filter, the optimal algebraic codebook and the fixed algebraic excitations are jointly vector-quantized using 7 bits.

The innovation codebook is built by combining four pulses of amplitudes $+1$ or -1 . The locations of the four pulses are picked from a predetermined set as shown in Table 2.9.

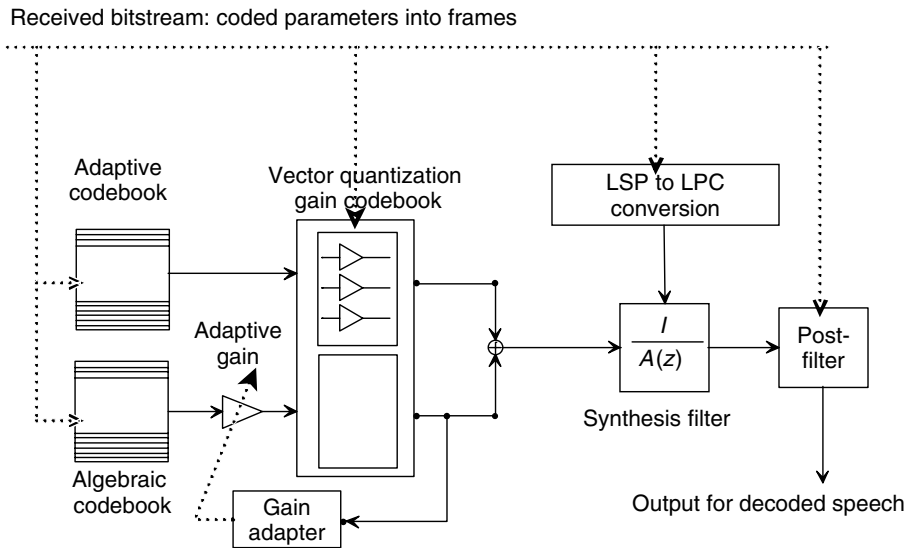


Figure 2.54 Basic principle of the ITU-T G.729 CS-ACELP 8-kbit/s speech decoder.

Table 2.9 Predetermined set of pulses of the innovation codebook used by G.729

Amplitude	Positions of pulses
± 1	0, 5, 10, 15, 20, 25, 30, 35
± 1	1, 6, 11, 21, 26, 31, 36
± 1	2, 7, 12, 17, 22, 27, 32, 37
± 1	3, 8, 13, 18, 23, 28, 33, 38
	4, 9, 14, 19, 24, 29, 34, 39

Table 2.10 G.729 bit allocation

Parameter	Subframe of 40 samples		Frame of 80 samples
	1st	2nd	
LSP	—	—	18
Pitch delay	8	5	13
Pitch parity	1	—	1
Algebraic code	13 + 4	13 + 4	34
Gain codebook	4 + 3	4 + 3	14
Total	—	—	80

The pulse positions of the first three pulses are encoded with 3 bits (eight possibilities) and the position of the fourth pulse is encoded with 4 bits (16 possibilities). Each pulse also requires 1 bit to encode the amplitude (± 1). This gives a total of 17 bits for the algebraic codebook in each subframe. Since only four nonzero pulses are in the innovation vector, very fast search procedures are made possible. Four nested loops corresponding to each pulse are used.

The structure of the final bitstream at 8 kbit/s is given in Table 2.10.

The G.729 decoder includes a post-filter consisting of three filters: a long-term post-filter, a short-term post-filter and a **tilt** compensation post-filter. The structure of the G.729 decoder is shown in Figure 2.54.

The ITU-T G.729 includes a detailed description in both fixed and floating point (annex C) with associated digital test vectors. Annex B describes a VAD/DTX/CNG scheme similar to G.723.1 (which was designed before G.729).

G.729 is recommended for use in voice over frame relay systems under the name clear voice. G.729 uses 16 MIPS. G.729 annex A is a lower complexity version (10 MIPS for the encoder compared with 18 MIPS) which was initially designed and recommended for **DSVD (digital simultaneous voice and data systems)**, but is now widely used in VoIP systems. G.729 also defines extensions at 6.4 kbit/s (annex D) and 11.8 kbit/s (annex E) which target **DCME** and **PCME** applications.

2.7.2 ITU-T G.723.1: dual-rate speech coder for multimedia communications transmitting at 5.3 kbit/s and 6.3 kbit/s

2.7.2.1 Speech encoding

G.723.1 is the result of an ITU competition for an efficient speech-coding scheme at a low bitrate for videoconferencing applications using a 28.8-kbit/s or 33.4-kbit/s V.34 voice band modem; this resulted in a compromise between the two best candidates (Audiocodes and DSP Group on one side and France Telecom on the other). This explains the two models of innovation codebooks found in the standard: the MP-MLQ (Audiocodes) for the higher bitrate and the ACELP (University of Sherbrooke) for the lower bitrate. There are some subtle differences between the general, advanced, CELP speech-coding scheme presented previously and the G.723.1 general structure, but the

basic principles and algorithmic tools are the same. The excitation signal for the high-rate coder is multi-pulse maximum likelihood quantization (MP-MLQ) and for the low-rate coder it is algebraic code-excited linear prediction (ACELP, the principle used in G.729 and GSM EFR).

The frame size is 30 ms and there is an additional look-ahead of 7.5 ms, resulting in a total algorithmic delay of 37.5 ms. Subframe duration is 7.5 ms. The MP-MLQ block vector quantization resembles the algebraic vector quantization procedure: six pulses with sign ± 1 for even subframes and five pulses with sign ± 1 for odd subframes are searched with an ABS MSE procedure. There is also a restriction on pulse positions: the positions can either be all odd or all even (indicated by a 'grid bit'). For the lower bitrate, the ACELP codebook was tuned to fit 5.3 kbit/s.

Tables 2.11 and 2.12 give the bit allocation for the two bitrates. The 189 bits of the higher bitrate are packed in 24 bytes and the 158 bits of the lower bitrate are packed in 20 bytes. Depending on the selected rate, either 24 or 20 bytes must be sent every 30 ms. Two bits in the first byte are used for signaling the bitrate and for the VAD/DTX/CNG operations described in Section 2.7.2.2.

The ITU-T recommendation includes a 16-bit, fixed point, detailed description and a floating point reference program (annex B). Both are provided as ANSI C programs. For the floating point version, software tools were designed to allow implementers to check their realizations. Conformance to the standard can be checked by undertaking all the digital test sequences. The complexity in fixed point for the encoder and both bitrates is around 16 MIPS. Annex C, devoted to mobile application, includes some mobile channel error-coding schemes.

G.723.1 is—together with G.729—one of the most well known coders used in VoIP networks and is predominantly used in PC-based systems. While most embedded systems (such as network gateways) support both G.729 and G.723.1, some of the leading IP phone vendors unfortunately recently decided to stop supporting G.723.1. This situation makes the lives of network administrators difficult, since many PC to IP phone calls can only negotiate G.711 as the common coder.

Table 2.11 Bit allocation table for the 6.3-kbit/s G.723.1 encoder (MP-MLQ)

Parameters coded	Subframe 0	Subframe 1	Subframe 2	Subframe 3	Total
LPC indices					24
Adaptive codebook lags	7	2	7	2	18
All the gains combined	12	12	12	12	48
Pulse positions	20	18	20	18	73(Note)
Pulse signs	6	5	6	5	22
Grid index	1	1	1	1	4
Total:					189

Note: By using the fact that the number of code words in the fixed codebook is not a power of 2, three additional bits are saved by combining the four MSBs of each pulse position index into a single 13-bit word.

Table 2.12 Bit allocation table for the 5.3-kbit/s G.723.1 (ACELP)

Parameters coded	Subframe 0	Subframe 1	Subframe 2	Subframe 3	Total
LPC indices					24
Adaptive codebook lags	7	2	7	2	18
All the gains combined	12	12	12	12	48
Pulse positions	12	12	12	12	48
Pulse signs	4	4	4	4	16
Grid index	1	1	1	1	4
Total:					158

2.7.2.2 Discontinuous transmission and comfort noise generation (annex A)

In order to reduce the transmitted bitrate during silent periods in-between speech, silence compression schemes have to be designed. They are typically based on the voice activity detection (VAD) algorithm and a comfort noise generator (CNG) that reproduces an artificial noise at the decoder side. The VAD must precisely detect the presence of speech and send this information to the decoder side. The G.723.1 VAD operates on a speech frame of 30 ms, and includes some spectral and energy computations.

One interesting feature of the VAD/DTX/CNG scheme of the G.723.1 coding scheme is that, when the characteristics of environmental noise do not change, nothing at all is transmitted. When needed, only the spectral shape and the energy of the comfort noise to be reproduced at the decoder side are sent. The spectral shape of the noise is encoded by LSP coefficients quantized with 24 bits and its energy with 6 bits. With the two mode-signaling bits, this fits in 4 bytes. The two signaling bits in each packet of 24, 20, or 4 bytes indicates either a 24-byte 6.3-kbit/s speech frame, a 20-byte 5.3-kbit/s speech frame, or a 4-byte CNG frame. The G.723.1 can switch from one bitrate to the other on a frame-by-frame basis (each 30 ms). At the decoder side, four situations can appear:

- (1) Receiving a 6.3-kbit/s frame (24 bytes).
- (2) Receiving a 5.3-kbit/s frame (20 bytes).
- (3) Receiving a CNG frame (4 bytes).
- (4) Receiving nothing at all (untransmitted frame).

In situations (1) to (3), the decoder reproduces the speech frame or generates the comfort noise signal with parameters indicated in the CNG frame. In situation (4), the decoder incorporates some special procedures to reproduce a comfort noise based on previously received CNG parameters. Similar VAD/DTX/CNG schemes have been included in G.729 and its annexes.

2.7.3 The low-delay CELP coding scheme: ITU-T G.728

In general, CELP coders cannot be used when there is a requirement for a low-encoding algorithmic delay. This is due to the LPC modeling principle, which requires a frame length of 10–30 ms (average stationary period of the speech signal) to compute the LPC.

Traditional low-delay encoders, such as PCM and ADPCM waveform speech coders, introduce a very low delay and do not significantly impact network planning (introduction of electrical echo cancellers). Unfortunately, they do not work at low bitrates.

The ITU was looking for a relatively low-bitrate encoder (16 kbit/s), with a low algorithmic delay (maximum 5 ms).

The G.728 low-delay coding scheme was designed by AT&T [A20], which efficiently merged the two concepts of stochastic codebook excitation (CELP) and backward prediction. In that scheme, there is no need to transmit the LPCs' which are computed in both the encoder and decoder, in a backward loop. Since backward prediction works on the current frame of samples from data of the previous samples, a relatively long set of samples can be analyzed to optimize the LPC filter without requiring a long frame to be accumulated before transmission.

The synthesis filter used in the ABS-MSE loop procedure does not include any LTP filter, but, in order to correctly represent high pitch values (and to efficiently encode generic signals such as music), its length is extended to 50 backward coefficients, updated every 20 samples. The coefficients are not transmitted but adapted (computed) in a backward manner by using the reconstructed signal in the encoder and decoder.

The frame length for the innovative codebooks is equal to only 5 samples (0.625 ms). For each set of 5 samples, an index found in the stochastic codebook of 128 entries is transmitted with a sign bit and a gain coded on 2 bits. In order to obtain an optimized codebook structure (the vectors), a very long and time-consuming training sequence on a large speech signal database was necessary. The gain is not directly encoded on 2 bits: a linear predictor is used to predict the gain, and the error of the optimal gain versus the predicted gain is encoded and transmitted. This leads to Table 2.13, the bit allocation table for the LD-CELP G.728. The LD-CELP speech encoder principle is shown on Figure 2.55.

In order to increase resistance to transmission errors, the index of the codebooks is transmitted using Gray encoding. Unlike a normal binary system, Gray encoding ensures that adjacent integers only have a single bit of difference: while a bit error can result in a large error on the integer value, a bit error in Gray encoding minimizes the error in the

Table 2.13 G.728 bit allocation

	Bit allocation per frame		Bitrate (bit/s)
	Parameters	Numbers of bits	
Excitation	Index	7	11,200
	Gain	2	3,200
	Sign	1	1,600
Frame length: 0.625 ms (5 samples)			16,000

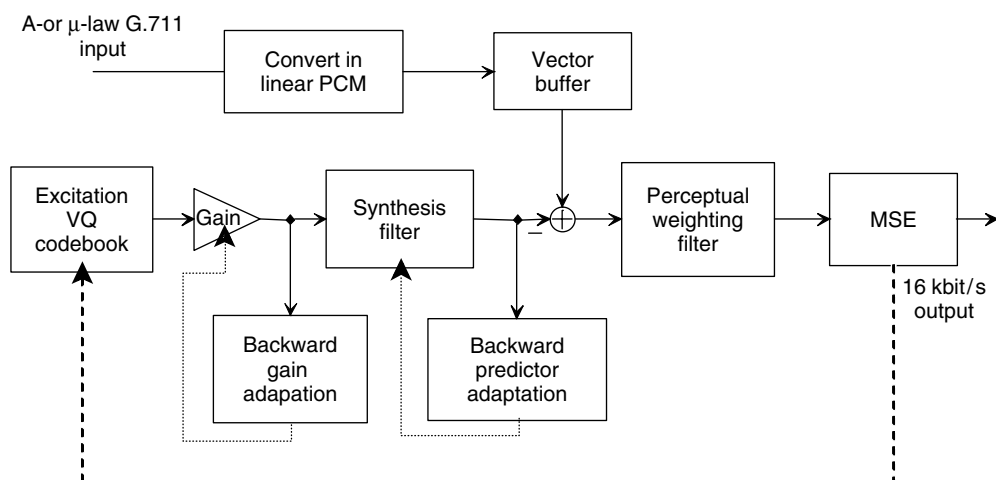


Figure 2.55 Low-delay CELP ITU-T G.728 encoder principle.

encoded value. For instance, integers 0 to 15 are encoded as 00, 01, 11, 10, 110, 111, 101, 100, 1100, 1101, 1111, 1110, 1010, 1011, 1001, 1000.

The introduction of the post-filter in the decoder shown in Figure 2.56 significantly improves the quality of decoded speech (this has allowed the AT&T proposal to fulfill the ITU-T requirements). G.728 has a very good score on the MOS scale (around 4) and is used in the H.320 videoconference system to replace the G.711 64 kbit/s with an identical quality bitstream of 16 kbit/s, leaving almost 48 kbit/s for the video on a single ISDN B channel. G.728 is also used in some modern DCME (digital circuit multiplication equipment), with extensions to 9.6 kbit/s and 12.8 kbit/s (replacing G.726 at 16 kbit/s, 24 kbit/s, and 32 kbit/s).

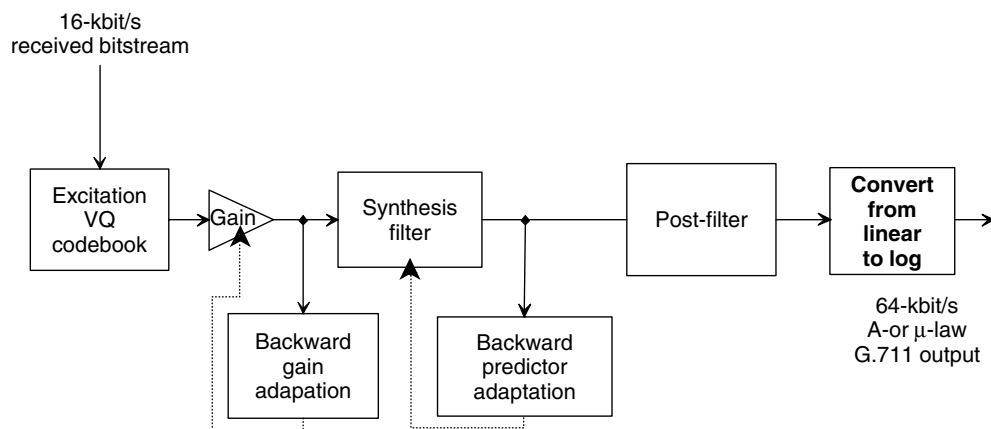


Figure 2.56 Low-delay CELP ITU-T G.728 decoder principle.

The major weaknesses of the original LD-CELP coding scheme are the difficulty to handle voice-band modem signals (an extension to 40 kbit/s is defined in annex I to solve this problem) and the high sensitivity to frame erasure due to the very long backward LPC filter and the use of a gain adaptation predictor. Recent work has significantly improved robustness and led to a new annex in the ITU-T G.728 suite of recommendations.

Another issue that G.728 shares with the G.729 coder (as opposed to the G.723.1 coder) is that there is no framing information in the transmitted bitstream. G.723.1 uses 2 bits in the first transmitted byte to indicate the type of packet. G.728 produces a 10-bit code for each 5-sample frame, but the decoder must precisely know which is the first, second, third, and fourth packet of 10 bits in order to synchronize the backward LPC filter adaptation procedure (although speech remains intelligible with G.728 even if desynchronization occurs). Strictly speaking, the use of G.728 requires a delay of 4 frames ($4 \times 0.625 \text{ ms} = 2.5 \text{ ms}$).

In the H.320 suite of recommendations, the H.221 framing procedure specifies a positioning mechanism for four packets of 10 bits of G.728 (2 bits per byte of a 64-kbit/s stream) or 80 bits of the G.729 8-kbit/s stream (1 bit per byte of a 64-kbit/s stream).

The first detailed description introduced in 1992 was for a floating point DSP and two additional years of work were needed to finalize a fixed point (16-bit) description in annex G. Unfortunately, the description is not in the form of ANSI C code, but extensive documentation.

The complexity of G.728 in fixed point is around 20 MIPS for the encoder and 13 MIPS for the decoder.

2.7.4 The AMR and AMR-WB coders

The adaptive multi-rate (AMR) coder is the result of ongoing work by ETSI (European Telecommunications Standards Institute) and 3GPP (3rd Generation Partnership Project, founded in December 1998), in collaboration with T1 in the US, TTC (Telecommunication Technologies Committee) and ARIB (Association of Radio Industries and Businesses) in Japan, TTA (Telecommunication Technologies Association) in Korea, and CWTS (China Wireless Telecommunication Standard group) in China, for the third generation of cellular telephony systems. In the current generation of cellular systems, three voice coders are used:

- GSM-FR, standardized in 1987, produces a 13-kbit/s bitstream and provides relatively good quality, with good immunity to background noise.
- GSM-HR, standardized in 1994, reduces the bitrate to 5.6 kbit/s, but is much more sensitive to background noise, which prevented any significant deployment.
- GSM-EFR, standardized in 1996, enhances the voice quality of GSM-FR in the presence of background noise with a similar bitrate (12.2 kbit/s), but the enhancement is perceptible only on error-free transmission channels.

While most voice coders seek to optimize the bitrate for a given quality of transmission channel for a desired voice quality level, so far little work has been done to take into

account the variable quality of a transmission channel. On most wire lines, it is true that the quality of transmission lines does not vary significantly, but obviously this is not the case for wireless transmission channels. With VoIP, the network conditions experienced by a PC-based phone, for instance, may also vary widely depending on whether the connection is via Ethernet, WiFi, at the office, or at a hotel. When the quality of the transmission channel varies, the optimal allocation of bits between source encoding and channel encoding varies: as the quality of the transmission link decreases, it becomes more efficient to allocate more bits to error protection schemes and fewer bits to the source encoding algorithm. Instead of optimizing the next generation coder for a given bitrate or transmission quality, it was decided that the new coder should be able to adapt to variable conditions (a 'multi-rate' coder) and provide optimal behavior under all these conditions ('adaptive'). The goals were:

- To improve the quality of GSM-FR on a channel with transmission errors, for a similar bitrate.
- To provide acceptable quality even on half-rate transmissions, in order to enhance transmission density in case of congestion.
- To adapt dynamically to the conditions of the radio channel.

The narrow-band AMR coder was standardized in March 1999; in addition, it was decided to study a version of the AMR coder for wide-band audio encoding (**AMR-WB**), which was finally standardized in March 2001.

2.7.4.1.1 Narrow-band AMR (GSM 6.90 ACELP AMR)

Narrow-band AMR provides eight bitrates (kbit/s):

- 7.95; 7.4 (IS 136); 6.7 (PDC-EFR); 5.9; 5.15 and 4.75 for half-rate transmission (similar to GSM-HR).
- 12.2 (GSM-EFR) and 10.2 for full-rate transmission (similar to GSM-FR and for UMTS).

Three of these modes interwork with existing equipment:

- GSM-EFR in 12.2-kbit/s mode.
- DAMPS in 7.4-kbit/s mode.
- PDC-EFR in 6.7-kbit/s mode.

Each mode is associated with a channel encoder which adds redundancy and interlacing in order to fill the available channel capacity (22.8 kbit/s for full rate, 11.2 for half-rate). On GSM networks, only four modes can be signaled (which can change dynamically at each even frame) and each service provider must select which modes are optimal for his network. While the network decides which mode to use depending on the conditions, the mobile terminal can signal its preferences. On UMTS networks, mobile terminals have to implement all eight modes.

The AMR coder is a CELP coder using ten LPCs. The various bitrate modes differ essentially in the number of bits allocated to quantization of the post-LPC residual signal: 38 bits for 12.2-kbit/s mode, 26 bits for 7.4-, 6.7-, and 5.9-kbit/s modes, and 23 bits for 5.15- and 4.75-kbit/s modes. All modes use an LTP filter to remove the pitch contribution, with some small precision differences depending on the mode (one-third precision for most modes). All AMR modes also use a post-filter to enhance the perceptual quality of the reproduced signal. Manufacturers of AMR devices have a choice of two algorithms for the VAD (one from Ericsson and Nokia, the other from Motorola); both reformed similarly during testing. The algorithm for the correction of erased frames was left out of the normative standard, although one example algorithm is provided. This provides some room for implementers to improve the quality of their algorithms and differentiate.

Besides the dynamic mode switching that optimizes bit allocation between source coding and channel encoding, the AMR also supports **unequal bit error detection and protection (UED/UEP)**. UED/UEP allows the loss of fewer frames over a network with a high bit error rate. Obviously, this has no impact on VoIP, since all errors are frame erasures.

2.7.4.1.2 AMR-WB (ITU G.722.2, UMTS 26171)

AMR-WB has been selected by 3GPP (TS 26.171) for UMTS phase 5 and was standardized by ITU as G.722.2 in January 2002. (A coder G.722.1 proposed by Picturitel was also standardized with similar characteristics, but it did not meet all the desired criteria for a 3G wideband codec). The AMR-WB algorithm was proposed by VoiceAge, Ericsson, and Nokia. It mainly targets three types of applications:

- GSM with a full-rate channel with a source-encoding rate limited to 14.4 kbit/s (TRAU frame).
- GSM-FR and EDGE with a full-rate channel with a source + channel-encoding rate limited to 22.8 kbit/s.
- UMTS with a source rate limited to 32 kbit/s.

The design goals of AMR-WB included:

- A voice quality at 16 kbit/s equal or superior to G.722 at 48 kbit/s.
- A voice quality at 24 kbit/s equal or superior to G.722 at 56 kbit/s.

AMR-WB provides nine bitrates (kbit/s):

- 14.25, 12.65, 8.85, and 6.6 for GSM-TRAU applications.
- 19.85, 18.25, and 15.85 are also available for GSM-FR applications.
- 23.85 and 23.05 are also available for EDGE and UMTS applications.

The AMR-WB coder jointly encodes the 0–6,400-Hz subband and the 6,400–7,000-Hz subband. The lower subband is processed by a CELP algorithm using a 16-coefficient LPC

filter, with the residual signal encoded using 46 bits for all modes except 6.6-kbit/s mode (36 bits). LTP filter analysis is extended to the full band or limited to the lower subband depending on the mode. The higher subband of the signal is regenerated by filtering a white noise signal with an LPC filter deduced from transmitted LPCs. One VAD algorithm has been standardized (annex A). As in the case of the AMR narrow-band coder, a frame erasure correction algorithm is provided, but is not part of the normative standard.

While AMR is mandatory for all terminals, the AMR-WB coder is mandatory only for terminals capable of sampling voice at 16 kHz; this will be introduced in UTMS phase 5. In multimedia communications, only AMR can be used during circuit switching, while both AMR and AMR-WB can be used for packet-switched communications (phase 5).

2.8 Quality of speech coders

Most speech coders have been designed to achieve the best possible level of speech reproduction quality, within the constraints of a given source-encoding bitrate. For narrow-band coders, the reference is ‘toll quality’, or the quality of speech encoded by the G.711 coder. For wide-band coders (transmitting the full 50–7,000-Hz band), the reference is the G.722 coder.

In fact, assessing the quality of a speech coder is a complex task which depends on multiple parameters:

- The absolute quality of the reproduced speech signal. This is the most used figure, but does not take into account interactivity (i.e., the delay introduced by the speech coder in a conversation). Several methods exist to assess the absolute, noninteractive speech quality of a coder. We will describe the MOSs which are the result of the ACR (absolute category rating) method and the DMOSs obtained with the CCR (comparative category rating) method. Several environmental conditions may influence speech degradation and need to be taken into account, such as speech input level, the type and level of background noise (bubble noise, hall noise, etc.).
- The delay introduced by the coder algorithm (algorithmic delay). This delay is due to the size of the speech signal frames that are encoded and to the additional signal samples that the coder needs to accumulate before encoding the current frame (look-ahead). Obviously, delay is only relevant for interactive communications, not for voice storage applications or noninteractive streaming applications.
- The complexity of the coder, which will result in additional processing delay on a given processor.
- The behavior of the coder for music signals, modem signals (maximum transmission speed that can be obtained), and DTMF transmission.
- Tandeming properties (i.e., the number of times voice can be encoded and decoded before voice quality becomes unacceptable). This can be assessed with the same coder used repeatedly or with other well known coders (e.g., the GSM coders used in cellular phones).

- Sensibility to errors (bit errors for cellular or DCME applications, or for VoIP frame erasures).
- The flexibility of the coder to dynamically adapt bit allocation to congestion and degradation of the transmission channel. Some coders provide only a fixed bitrate, while others can switch between bitrates dynamically (**embedded** coders). Hierarchical coders like G.722 can generate several simultaneous streams of encoded speech data: a core stream that needs to be transmitted as reliably as possible through the transmission channel (either on a high QoS level or using an efficient redundancy mechanism), and one or more ‘enhancement’ streams that can be transmitted on lower quality channels.

The importance of these parameters depends on the final application and the target transmission network (fixed network, wireless network, serial transmission links that generate bit errors, packet transmission links that generate frame erasure errors, etc.). For most applications, the shortlist of key parameters includes the bitrate, complexity of the coder, delay, and quality.

When a standard body decides to standardize a new voice coder, the first step is to specify the quality acceptance criteria for the future coder. As an example, Table 2.14 [A19] is a summary of the terms of reference set to specify the ITU 8-kbit/s coder (the future G.729). This new coder was intended to ‘replace’ the G.726 at 32 kbit/s or the G.728 at 16 kbit/s.

2.8.1 Speech quality assessment

In order to assess the level of quality of a speech coder, **objective measurements** (computed from a set of measurements on the original signal and the reproduced signal) are not reliable for new coders. In fact, most objective, automated measurement tools can only be used on well-known coders and well-known networks, and simply perform some form of interpolation using quality scores in known degradation conditions obtained using a subjective method. In the early days of VoIP, people tended to apply the known, objective measurement tools, calibrated on fixed TDM networks, without realizing that transmission link properties were completely different: frame erasure as opposed to random bit errors, correlated packet loss, degradations due to the dynamic adaptation of jitter buffers, etc. Needless to say, many of these ‘objective’ tests were in fact designed as a marketing tool for this or that voice coder.

Subjective measurements are therefore indispensable. What can assess the quality of a voice coder better than a human being? Unfortunately, subjective measurements of speech quality require a substantial effort and are time-consuming. In order to obtain reliable and reproducible results, a number of precise guidelines must be followed:

- Ensure that the total number of listeners is sufficient for statistically reliable results.
- Ensure that the auditory perception of listeners is normal (medical tests may be necessary).
- Instruct the listeners of the methodology of the tests.
- Ensure that the speech material is diversified: gender of talkers, pronunciation, age of the talkers (child).

Table 2.14 Terms of reference for the 8 kbit/s ITU-T speech coder

Items	Parameters	Requirements	Objectives
Quality for speech		No worse than that of ITU-T G.726 at 32 kbit/s	
Performance in the presence of transmission errors (bit error)	Random bit error: BER ≤ 0.1	No worse than that of ITU-T G.726 at 32 kbit/s under similar conditions	Equivalent to ITU-T G.728
Performance in the presence of frame erasure	Indication of frame erasure (random and burst)	Less than 0.5 MOS when there are 3% missing frames	As small as possible
Input-level dependence	- 36 dB, -16 dB	No worse than that of ITU-T G.726 at 32 kbit/s	As small as possible
Algorithmic delay		≤ 16 ms	≤ 5 ms
Total codec delay		≤ 32 ms	≤ 10 ms
Cascading		2 asynchronous coding ≤ 4 asynchronous ITU-T G.726 at 32 kbit/s	3 asynchronous coding ≤ 4 asynchronous ITU-T G.726 at 32 kbit/s
Tandeming with other ITU-T standards		≤ 4 asynchronous ITU-T G.726 at 32 kbit/s	3 asynchronous coding ≤ 4 asynchronous ITU-T G.726 at 32 kbit/s
Sensibility to background noise	Car noise	No worse than that of ITU-T G.726 at 32 kbit/s	
	Bubble noise		
	Multiple speakers		

- Ensure that the test is performed in several languages by a number of experienced organizations (problems may occur with languages other than English, Japanese, German, Italian, Spanish, or French even on a well-standardized speech coder),
- Ensure that all the environmental conditions of use of the candidate coder are tested (such as level dependencies, sensibility to ambient noise and type of noise, error conditions, etc.).
- Appropriate choice of pertinent listening conditions: choice of equipment (headphones, telephone handsets, loudspeakers) and loudness of the samples.

These tests are fully specified in ITU-T recommendations ITU-T P.800 and P.830 [A1], [A4]. Obviously, these tests are very time-consuming, expensive (dedicated rooms and studios, high-quality audio equipment), and require well-trained and experienced organizations.

The following subsections provide an overview of these methods. We will focus on listening opinion tests, although other tests, such as conversation opinion tests, also exist.

2.8.2 ACR subjective test, mean opinion score (MOS)

For low-bitrate telephone speech coders (between 4 kbit/s and 32 kbit/s), the **absolute category rating (ACR)** is the most commonly used subjective measurement method. It is the method that produces the well-known **MOS** figure.

In ACR subjective tests, listeners are asked to rate the ‘absolute’ quality of speech samples, without knowing what the reference audio sample is. Listening quality is generally assessed using the scale in Table 2.15.

An MOS is an absolute judgment without references, but in order to insure coherence and calibration between successive tests, some reference is needed. For this purpose, a reference audio sample is inserted among the samples given to listeners (without any notice). Very often, the **modulated noise reference unit (MNRU)** is used: this device simulates voice degradation and noise level equivalent to that produced by the A- or μ -law PCM coding scheme. It is still common to read press articles or conference presentations that present ‘the’ MOS of a new coder without also presenting the MOS obtained in the test by a reference coder. Such values should be considered with skepticism: some vendors choose to give an MOS of ‘5’ to G.711, shifting all MOSs up by almost one full MOS point, while others do not even have such a reference coder as part of their test.

Table 2.15 Listening quality scale for absolute category rating

Excellent	5
Good	4
Fair	3
Poor	2
Bad	1

The MOS figure is calculated statistically from the marks given to each audio sample by listeners. The relevance of MOS and the confidence interval of the results must be determined by statistical analysis, requiring a lot of experiments. Generally, an ACR subjective test requires an average of 24 listeners (3 groups of 8). The typical test sample consists in a double sentence: 0.5 s of silence, 2 s for sentence #1, 0.5 s of silence, 2 s for sentence #2.

Figure 2.57 provides an overview of typical MOS values for various categories of speech coders as a function of bitrate [A6]. More precisely, Table 2.16 gives the MOS figure and type of well-known, ITU-T standardized speech coders. For mobile standards see Table 2.17 and for DOD standards see Table 2.18.

Table 2.18 clearly shows the magnitude of the improvements in speech coders in ten years: the speech quality that can be obtained at 2.4 kbit/s goes from synthetic to 3.2 (fair)!

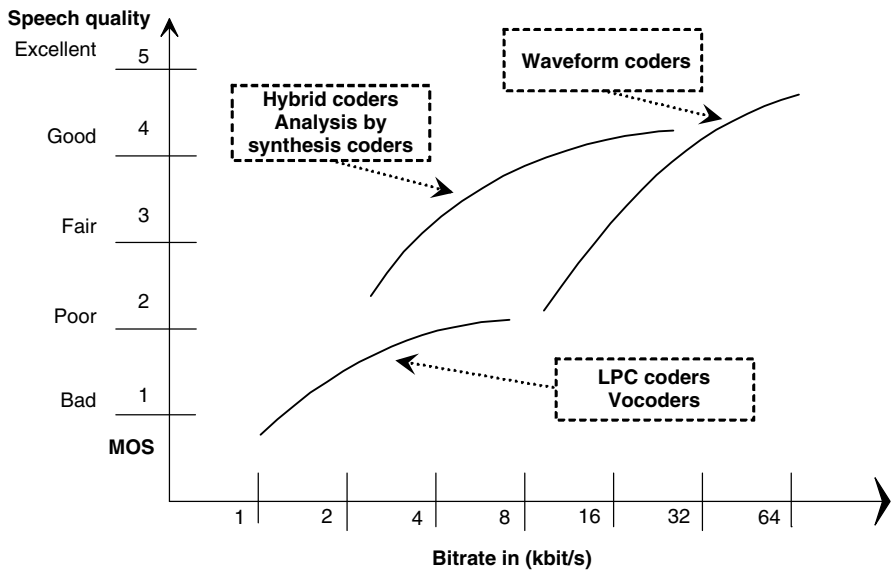


Figure 2.57 MOSs as a function of the bitrate and coder technology.

Table 2.16 MOSs of some ITU coders

Standard	G.711	G.726 or G.721	G.728	G.729	G.723.1
Date of approbation	1972	1990 (1984)	1992	1995	1995
Bitrate (kbit/s)	64	16/24/32/40	16	8	6.3–5.3
Type of coder	Waveform: PCM	Waveform: ADPCM	ABS: LD-CELP	ABS: CS-ACELP	ABS: MP-MLQ, CS-ACELP
Speech quality (MOS)	4.2	2/3.2/4/4.2	4.0	4.0	3.9/3.7

Table 2.17 MOSs of coders used in mobile telephony

Standard	ETSI-GSM 06.10	ETSI-GSM 06.20	ETSI-GSM-EFR	ETSI-TETRA	USA IS54 TDMA	USA IS96 CDMA	JAPAN JDC 1	JAPAN JDC 2
Date of approbation	1988	1994	1995	1994	1989	1992	1990	1994
Bitrate (kbit/s)	13	5.6	13	4.56	7.95	8/4/2/1	6.7	3.6
Type of coder	ABS: RPE-LTP	ABS: VSELP	ABS: ACELP	ABS: ACELP	ABS: VSELP	ABS: Qualcomm CELP	ABS: VSELP	ABS: PSI-CELP
Speech quality (MOS)	3.6–3.8	3.5–3.7	4	3.3–3.5	3.5–3.7	3.3–3.5	3.4–3.6	3.4–3.6

Table 2.18 MOS scores of military coders

Standard	American DOD FS1015	American DOD FS1016	American DOD
Date of approbation	1984	1990	1995
Bitrate (kbit/s)	2.4	4.8	2.4
Type of coder	Vocoder: LPC 10	ABS: CELP	ABS: MELP
Speech quality (MOS)	Synthetic quality	3	3.2

2.8.3 Other methods of assessing speech quality

ACR is not the only method available for speech quality assessments. The **degradation category rating (DCR)** and the **comparison category rating (CCR)** are also used, mostly for high-quality coders.

The DCR method is preferred when good-quality speech samples are to be compared. The DCR method produces a **degradation mean opinion score (DMOS)**. The range of degradation is presented Table 2.19.

DCR methodology is similar to ACR, except that the reference sample is known to the listener and presented first: pairs of samples (A–B) or repeated pairs (A–B, A–B) are presented with A being the quality reference.

CCR is similar to DCR, but the order of the reference sample and the evaluated coder sample is chosen at random: this method is interesting mostly for speech enhancement systems. The result is a **comparison mean opinion score (CMOS)**.

For all interactive communication systems, especially VoIP, conversational tests are also very instructive because they try to reproduce the real service conditions experienced by final users. The degradations introduced by echo and delays, not present in MOS tests, can also be taken into account. The test panel is asked to communicate using the system under test (e.g., DCME or VoIP) and is instructed to follow some scenario or to play some game and finally give their opinion on the communication quality and on other parameters, such as clarity, level of noise, perception of echoes, delays, interactivity, etc. Once again, each participant gives a score from 1 to 5 (as described in the ITU-T P.800 recommendation), and statistical methods are used to compute the test result (**MOS_c**, ‘c’ for communication) and the confidence interval. Interactive tests are very difficult to control, and consistency and repeatability are very hard to obtain.

An example of the sample conditions used in international experiments conducted by ITU-T when selecting a 8-kbit/s candidate is given in Table 2.20.

Table 2.19 DMOS table

Degradation is inaudible	5
Degradation is audible but not annoying	4
Degradation is slightly annoying	3
Degradation is annoying	2
Degradation is very annoying	1

Table 2.20 Typical ITU experiments for coder selection

Experiment #	Description
Experiment 1	Clean speech quality and random bit error performance
Experiment 2	Tandem connection and input-level dependence
Experiment 3	Frame erasure: random and burst
Experiment 4	Car noise, bubble noise, multiple speakers, and music
Experiment 5	Signaling tones: DTMF, tones, etc.
Experiment 6	Speaker dependence: male, female, child

2.8.4 Usage of MOS

As MOSs represent a mean value, extreme care must be taken to select or promote a speech coder for a specific application. It must be checked that all the candidate coders are evaluated under the same conditions (clean speech, level dependence, background noise of several types and level of noise, sensibility to bit errors, frame erasure, etc.) and that the test conditions actually represent the real conditions of the communication channels used by the application. International bodies, such as the ITU-T, TIA, ETSI, JDC, are well aware of the situation and evaluate each coder according to a rigorous and thorough methodology. Too often, manufacturers publish and promote MOS results that have no scientific value. A few examples of common tricks are:

- Publishing good MOS results with a high network loss rate (10%!), but with a carefully engineered loss pattern that does not represent the real situation (e.g., exactly one packet out of 33 is lost, as opposed to the correlated packet loss in a real network).
- Taking a higher MOS value for the reference coder, but omitting this detail in the final test documentation.
- Using test samples free from background noise.
- Using listening equipment of low quality that smooths the perception of all coders and therefore boosts the results of the tested coder after normalization.

2.9 Conclusion on speech-coding techniques and their near future

2.9.1 The race for low-bitrate coders

Many coding schemes have not been described in this chapter:

- The MELP (mixed excitation LPC) coder, retained in the new 2,400-bit/s US Federal standard.
- The VSELP (vector sum excited LP) coder, used in the half-rate 5.6-kbit/s GSM system.

- The multi-rate Q-CELP (Qualcomm CELP) at 1, 2, 4 and 8 kbit/s, used in the cellular US IS96 CDMA system.
- Multi-band excitation (MBE) coders.
- Sinusoidal transform coders (STCs).

The number of coding schemes reflects the constant progress of speech-coding technology. This progress has been driven by major telecommunication applications.

The first application of voice coding was the optimization of submarine cables and expensive long-distance links. The focus was on reducing bitrate while preserving good voice quality, and on providing reasonable support for modem and fax transmission. This led to relatively simple voice coders like the ITU G.726 at 32 kbit/s (1990).

Since 1990 the bitrate required to reach toll quality has decreased to about 8 kbit/s, or one bit per sample!

2.9.2 Optimization of source encoding and channel encoding

After 1999, the priority was no longer the absolute reduction of the bitrate, because the price of bandwidth continuously decreased on fixed lines. The driving application for voice-coding technology became wireless telephony. Wireless telephony offers a limited transmission bandwidth, which can be addressed by existing algorithms, but more importantly the transmission quality of the transport channel varies continuously. The best voice quality depends not only on how good the source encoding of the voice coder is, but also and, just as importantly, on how well channel encoding can correct transmission errors.

The priority of coder research became the optimal combination of source-encoding and channel-encoding methods in a given envelope. Both compete for the available bitrate on the channel:

- If the number of errors is low, the channel-encoding algorithm is not necessary and does not generate any redundancy information, and the full available bitrate can be used for the voice coder (source encoding).
- If the number of errors is high, the channel-encoding algorithm will generate a lot of redundancy information to protect voice coder information. As a consequence, the source-encoding algorithm needs to reduce its bitrate.

Dynamic optimization of the source-encoding and channel-encoding allocation within the available bitrate is a complex problem. The AMR and AMR-WB coders are the result of research carried out on this problem: both use multiple source-encoding algorithms, each combined with a channel-encoding algorithm, and the optimal mode is switched dynamically as transmission conditions change.

This new generation of voice coders provides a much more homogenous experience over a varying quality radio channel: voice quality does degrade as the radio conditions of the transmission channel get worse, but does so progressively, without the catastrophic

degradation experienced with single-mode codecs. Dynamic selection of the optimal source coder and channel coder makes the best possible use of the transport link under any conditions.

To a large extent, the enhancements of voice coders that were originally designed for radio channels are also valid for VoIP. The only significant difference, on is that radio channels create bit errors in the data stream (characterized by a **bit error rate, or BER**), while IP networks create frame-level (packet-level) errors. For a given bitrate, VoIP can also benefit from an optimal combination of source encoding and channel encoding, but the optimal channel-encoding method for VoIP differs from the optimal channel-encoding method for wireless applications, as it must protect against frame erasures.

2.9.3 The future

2.9.3.1 VoIP

What should we expect next? Perhaps the most important feedback from early VoIP trials was that there was no market for sub-toll-quality voice. Users are not only not prepared to pay less for such voice quality, they are not prepared to pay at all. As a consequence there are no big incentives to continue to decrease the bitrate of a pure voice coder, and IP overheads would make such progress irrelevant anyway. Although there is still some progress for voice coders to make at 4 kbit/s and below, none of these coders achieves toll quality, and therefore they can only be used in degraded conditions, in combination with a high-redundancy channel-encoding method, or in military applications.

One of the issues about current coders is their poor performance for the transport of music, another is the degradation of voice encoding when there are multiple speakers or background noise. It seems that most of the efforts in the coming years will be to improve these weaknesses, while keeping a bitrate of 8 kbit/s or even above.

Unlike wireless networks, which will always have a tight bandwidth constraint (shared medium), VoIP applications benefit from the constant progress of wired transmission links. As the cost of bandwidth decreases, it becomes more interesting to provide users with a better telephony experience. Some VoIP systems already support wide-band voice coders, such as G.722, which make it easier to recognize the speaker and provide a more natural sound. Beyond wide band, multichannel coders (stereo, 5.1) can provide spatialized sound, which can be useful for audio- or videoconferences. Since 2002, it seems the focus of voice encoding for VoIP systems has shifted from low-bitrate encoders to these high-quality wide-band encoders.

We believe that in the coming years wideband encoders will become increasingly common in VoIP systems.

2.9.3.2 Broadcast systems

Both wireless telephony and VoIP are interactive, one-to-one systems, where there is only one transmission channel. Audio broadcast systems on the Internet pose a different problem. For such systems there are many transmission channels, each with different degradation levels. If a separate information stream is sent on each channel (multi-unicast),

then dynamic mode selection works, but for multicast systems, where everyone receives the same information, it would not be optimal to use the bit allocation between source encoding and channel encoding that is optimal for the worst channel, as even listeners using the best transmission channels would experience poor audio quality.

For such links, the focus is on hierarchical coders, which produce several streams of information: a core stream, providing low quality, is transmitted with the highest possible redundancy (and an above-average QoS level is available), one or more enhancement streams that provide additional information on top of the core stream information, allowing receivers to improve playback quality. ISO-MPEG 4 is an example of a hierarchical encoder. These systems are mainly useful for broadcast of multicast systems, and their use for VoIP (e.g., with H.332) mainly depends on the deployment of multicast-capable IP networks.

2.10 References

2.10.1 Articles

- [A1] ITU-T. G.7xx, P.3xx, H.3xx Recommendations, ETS ETSI xxxx, IS ISO-MPEG yy.
- [A2] P. Combescure and M. Mathieu. Codage des signaux sonores. *L'Écho des recherches*, **121**(3), 1985, pp. 13–22.
- [A3] P. Combescure, A. Le Guyader, D. Jouvét, and C. Sorin. Le traitement du signal vocal. *Annales des Télécommunications*, **50**(1), 1995, pp. 142–164.
- [A4] A. Gersho. Advances in speech and audio compression. *Proceedings of the IEEE*, June 1994, pp. 900–918.
- [A5] S. Dimoultsas. Standardizing speech coding technology for network application. *IEEE Communications Magazine*, November 1993, pp. 26–33.
- [A6] R.V. Cox. Speech coders: From idea to product. *AT&T Technical Journal*, March/April 1995, pp. 14–21.
- [A7] L.M. Supplee, R.P. Cohn, and J.S. Collura. MELP: The new federal standard at 2400 BPS. *Proceedings of ICASSP Conference*, 1997, pp. 1591–1594.
- [A8] T.E. Tremain. The government standard linear predictive coding algorithm: LPC 10. *Speech Technology*, April 1982, pp. 40–49.
- [A9] Y. Mahieux, J-P. Petit, and A. Charbonnier. Codage pour le transport du son de haute qualité sur le réseau de télécommunications. *L'Écho des recherches*, **146**(4), 1991, pp. 25–35.
- [A10] K.H. Brandenburg, G. Stoll, Y.F. Dehéry, J.D. Johnston, L.D. Kerkof, and E.F. Schröder. ISO-MPEG-1 audio: A generic standard for coding of high quality digital audio. *Journal of the Audio Engineering Society*, **42**, October 1994, pp. 780–792.
- [A11] M. Bosi, K.H. Brandenburg, S. Quackenbush, L. Fielder, K. Akagiri, H. Fuchs, M. Dietz, J. Herre, G. Davidson, and Y. Oikawa. ISO/IEC MPEG-2 advanced audio coding. *Journal of Audio Engineering Society*, **45**(10), October 1997, pp. 789–814.
- [A12] L.D. Fielder, M. Bossi, G. Davidson, M. Davis, C. Todd, and S. Vernon. AC-2 and AC-3: Low-complexity transformed-based audio coding. *Collected Papers on Digital Audio Bit-Rate Reduction*, pp. 54–72. Editors N. Gilchrist and C. Grewin. Audio Engineering Society.
- [A13] W.R. Daumer, P. Mermelstein, X. Maître, and I. Tokizawa. Overview of the ADPCM coding algorithm. *Proceedings of GLOBECOM*, 1984, pp. 23.1.1–23.1.4.

- [A14] P. Noll. Wide band speech and audio coding. *IEEE Communications Magazine*, November 1993.
- [A15] X. Maitre. 7 kHz audio coding within 64 kbit/s. *IEEE Journal on Selected Areas on Communications*, **6**(2), February 1988, pp. 283–298.
- [A16] K. Hellwig, P. Vary, D. Massaloux, J.P. Petit, C. Galand, and M. Rosso. Speech codec for the European mobile radio system. *GLOBECOM Conference*, 1989, pp. 1065–1069.
- [A17] J.P. Campbell, T.E. Tremain, and V.C. Welsh. The federal standard 1016 4800 bps CELP voice coder. *Digital Signal Processing*, **1**, 1991, pp. 145–155.
- [A18] R. Salami, C. Laflamme, J.P. Adoul, A. Kataoka, S. Hayashi, T. Moriya, et al. Design and description of CS-ACELP: A toll quality 8 kbit/s speech coder. *IEEE Transactions on Speech and Audio Processing*, **6**(2), March 1998, pp. 116–130.
- [A19] Special features on ITU-T standard algorithm for 8 kbit/s speech coding. *NTT Review*, **8**(4), July 1996.
- [A20] J.H. Chen, R.V. Cox, Y.C. Lin, N. Jayant, and M.J. Melchner. A low-delay CELP coder for the CCITT 16 kb/s speech coding standard. *IEEE Journal on Selected Areas on Communications*, **10**(5), June 1992, pp. 830–849.
- [A21] D.A. Huffman. A method for the construction of minimum-redundancy codes. *Proc. Inst. Radio Eng.*, **40**, 1952, pp. 1098–1101.

2.10.2 Books

- [B1] A.M. Kowdoz. *Digital Speech: Coding for Low Bit Rate Communications Systems*. Wiley.
- [B2] N. Moreau. *Techniques de compression des signaux*. Masson, CNET-ENST, 1995.

2.11 Annexes

2.11.1 Main characteristics of ITU-T standardized speech coders

Standard	G.711	G.721	G.726	G.727	G.728	G.729	G.723.1	G.722	G.722	G.722.2 (AMR-WB)
Approval date	1972	1984	1990	1990	1992	1995/99/99	1995	1988	1999	2002
Bitrate (kbit/s)	64	32	16/24/32/40	16/24/32/40	16	8/6.4/11.8	6.3/5.3	48/56/64	32/24	23.85; 23.05; 19.85; 18.25; 15.85; 14.25; 12.65; 8.85; 6.6
Transmitted bandwidth	300 Hz–3.4 kHz	300 Hz–3.4 kHz	300 Hz–3.4 kHz	300 Hz–3.4 kHz	300 Hz–3.4 kHz	300 Hz–3.4 kHz	300 Hz–3.4 kHz	100 Hz–7 kHz	100 Hz–7 kHz	100 Hz–7 kHz
Complexity	0.1	10	12	12	33	22 (12 G.729A)	16/18	10	14	40
MIPS						2,500	2,100	256	3,600	6.5 kw
RAM (words)	2	256	256	256	3,400	9,500	7,000	4,000	? kw	16 kw
ROM (words)	50	4,000	5,000	5,000	8,000	10	30	0.125	20	20
Frame length (ms)	0.125	0.125	0.125	0.125	0.625	10	30	0.125	20	20
Look-ahead (ms)	0	0	0	0	0	5	7.5	1.5	20	5
Speech quality (MOS)	4.2	4.0	4.0'	4.0'	4.0	4.0	3.9/3.7	+64, 56 kbit/s	+ (32 kbit/s)	+ (23.85 kbit/s)

2.11.2 Main characteristics of cellular mobile standardized speech coders

Country	ETSI EUROPE				TTA USA				CRC JAPAN	
	GSM 06.10 (GSM-FR)	GSM 06.20 (GSM-FR)	GSM 06.71 (AMR)	TETRA	GSM 06.60 (GSM-FR)	IS54 TDMA	IS96 CDMA	IS136 TDMA (PCS)	JDC 1	JDC 2
Approval date	1988	1994	1999	1994	1996	1990	1992	1996	1990	1993
Bitrate (kb/s)	13	5.6	4.75, 5.15, 5.9, 6.7, 7.4, 7.95, 10.2, 12.2	4.56	12.2	7.9	8.4/2/1	7.4	6.7	3.6
Bitrate for channels existing	9.8	5.8	Up to total of 22.8 (HR) or 11.2 (LR)	2.633	10.6	5.1		5.6	4.5	2
Transmitted bandwidth	300 Hz	300 Hz	300 Hz	300 Hz	300 Hz	300 Hz	300 Hz	300 Hz	300 Hz	300 Hz
Complexity	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4
MIPS	2.5	1.5	15	15	15.1	14	10-12	16.1	14/18	30-40
RAM (words)	800	3,200	4,800	4,000	4,700	?	?	2,400	?	?
ROM (words)	2,000	18,000	15,000	7,000	5,900	?	?	5,400	?	?
Frame length (ms)	20	20	20	20	20	20	20	20	20	40
Overhead (ms)	0	?	5	?	0	5	5	5	5	10
Speech quality	3.6-3.8	3.5-3.7	Var	3.3-3.5	4.1	3.5-3.7	3.3-3.5	4	3.4/3.6	3.4/3.6
Noise << quality >>	-	-	Var	-	-	-	-	-	-	-
Capability to encode music	-	-	-	-	-	-	-	-	-	-
Robustness to errors	+/-	+/-	+/-	+/-	+/-	+/-	+/-	+/-	+/-	+/-
Maximum speed for modems (kb/s)										
Acceptable number of tandem	2	2	2/1	2/1	2	2	2/1	2	2	2/1
Coder type (all AAS)	RPE LTP	VSELP	ACELP	ACELP	ACELP	VSELP	CELP	ACELP	VSELP	PS

3

Voice Quality

3.1 Introduction

A common joke among IP telephony engineers is to say that if they had proposed to carry voice over IP a couple years ago, they would have been fired. This remains a private joke until you make your first IP phone call to someone on an old PC without a headset, to find out that the only person you heard was yourself (this is no longer true today ... even PCs have software echo cancelers). Another way to find out why there really is a problem with IP telephony is to try a simple game: “collaborative counting”.

Collaborative counting has a simple rule: if you hear the person you talk to say ‘ n ’, you immediately say ‘ $n + 1$ ’. In order to compare ‘classic’ telephony with IP telephony, you first make a regular phone call to someone you know and say ‘1’, he goes ‘2’, etc. Keep an eye on your watch and measure how long it takes to count to 25.

Then you make an IP phone call and play the same game. In all cases, it will take much longer ...

The problems we have just emphasized, echo and delay, have been well known to telephone network planners since the early days of telephony, and today’s telephone networks have been designed to keep these impairments imperceptible to most customers.

When carrying voice over IP, it becomes much more difficult to control echo, delay, and other degradations that may occur on a telephone line. As we will see, it will require state-of-the-art technology and optimization of all components to make the service acceptable to all customers.

However, once echo and delay are maintained within acceptable limits by proper network engineering, VoIP can use voice coders, such as G.722 (‘wide-band’ coder), which provide an absolute voice quality beyond that of current PSTN networks. It is frequently heard that VoIP can reach ‘toll quality’; in fact, in the future VoIP will provide a voice quality that exceeds ‘toll quality’ through rigorous planning and design.

3.2 Reference VoIP media path

The media path of IP telephony calls can be modeled as shown in Figure 3.1, when there is no POTS or ISDN terminal involved. The situation gets a little more complex when interworking with an ISDN phone through a gateway (Figure 3.2):

When the gateway interfaces with an analog network, the user–network interface is in most cases using only 2 wires (incoming and outgoing signals share the same pair), and a 4-wire/2-wire hybrid is required (Figure 3.3). The model includes the most significant sources of voice quality degradation:

- The IP network introduces packet loss, delay, and jitter.

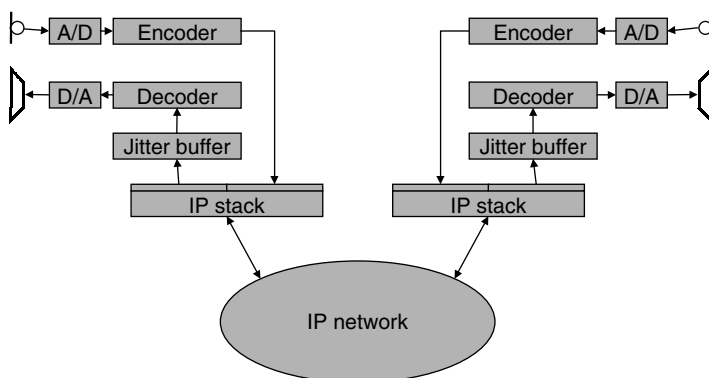


Figure 3.1 Reference VoIP media path.

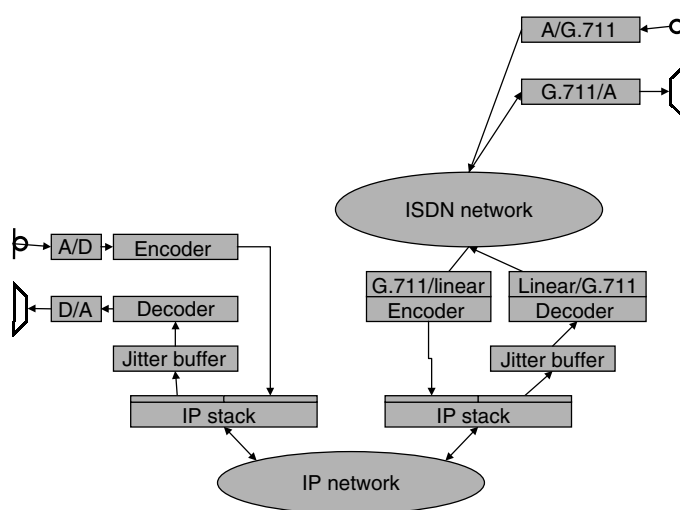


Figure 3.2 Reference VoIP to ISDN path.

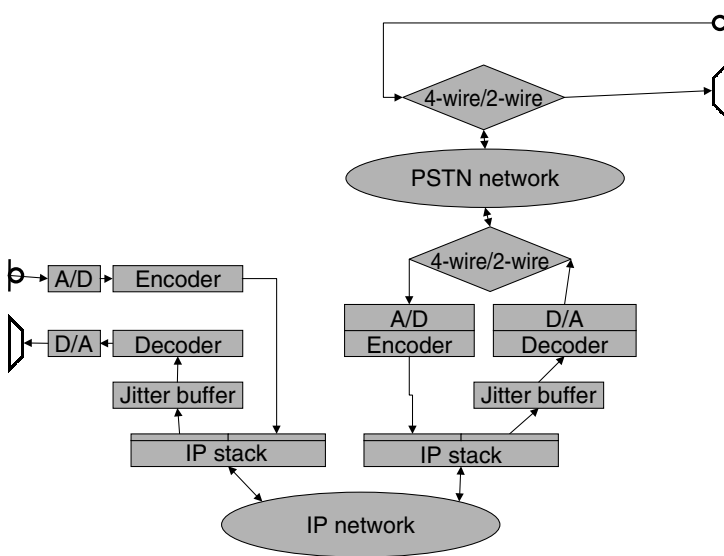


Figure 3.3 Reference VoIP to POTS path.

- The jitter buffers (JBs) influence end-to-end delay and frame loss.
- The acoustic interfaces introduce acoustic echo.
- The 2-wire/4-wire interfaces introduce electric echo.
- The PSTN network, which potentially introduces further delays.

In this chapter we describe the main factors influencing end-user perception of voice quality. Most of those factors are common to switched circuit telephony and IP telephony. However, IP telephony has some unique characteristics, such as long delays, jitter, and packet loss, and therefore requires a new framework for assessing voice quality.

3.3 Echo in a telephone network

3.3.1 Talker echo, listener echo

The most important echo is **talker echo**, the perception by the talker of his own voice but delayed. It can be caused by electric (**hybrid**) echo or acoustic echo picked up at the listener side.

If talker echo is reflected twice it can also affect the listener. In this unusual case the listener hears the talker's voice twice: a loud signal first, and then attenuated and much delayed. This is **listener echo**.

These two types of echo are illustrated on Figure 3.4.

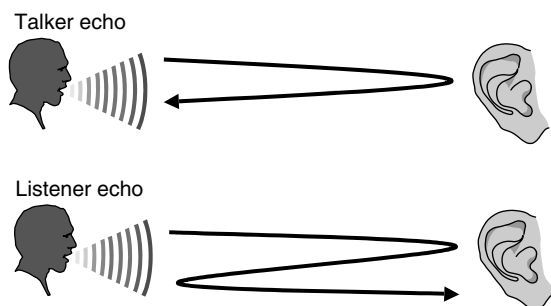


Figure 3.4 Talker and listener echo.

3.3.2 Electric echo

3.3.2.1 What is a hybrid?

The simplest telephone system would look like Figure 3.5. However, to use fewer wires, the phone system was designed to use just two wires. The first 2-wire phones looked like Figure 3.6. Because of parasitic capacities on the line, most microphone signals were dissipated in the talker's loudspeaker (who then tended to speak lower), and almost nothing reached the listener.

The final design arrived at is as shown on Figure 3.7, where Z_{ref} matches the characteristic impedance of the line. Now, the microphone signal is split equally between Z_{ref} and the line, and the speaker hardly hears himself in his own loudspeaker (a small unbalance is kept for him not to have the impression that he is talking in the air). In the ETSI standard Z_{ref} is a 270- Ω resistor connected to a 750- Ω resistor in parallel with a 150-nF capacitor. In France, for instance, Z_{ref} is a 150-nF capacitor in parallel with a 880- Ω resistor, wired to a 210- Ω resistor (complex impedance), but some older phones are also equipped with a real impedance of 600 Ω .

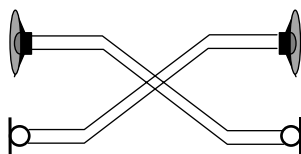


Figure 3.5 Simplest phone network.

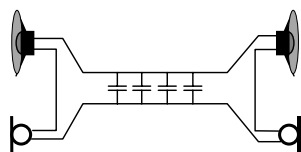


Figure 3.6 Basic phone connection over a single pair.

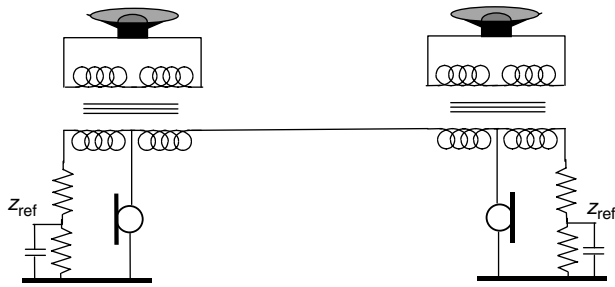


Figure 3.7 Improved design using a hybrid.

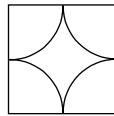


Figure 3.8 Hybrid symbol.

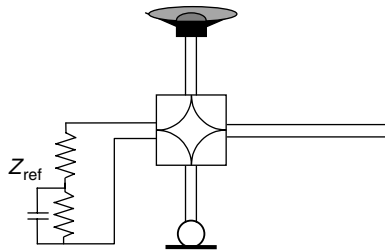


Figure 3.9 Simplified representation of an analog phone.

These values were found to be a good average for a typical line. The actual impedance of a given line will vary according to its length (between 0 km and 9 km, typically), so there is *always* some mismatch.

The common way to symbolize this impedance adaptation device is illustrated in Figure 3.8, where each corner represents 2 wires. It is called a duplexer, or a hybrid. Each half of the circuit of Figure 3.7 can be represented as in Figure 3.9. A hybrid can be integrated easily, a possible circuit is shown in Figure 3.10.

The hybrid is also commonly used in an analog telephone network to allow line signal amplification using the configuration of Figure 3.11.

3.3.2.2 Electric echo

In Figure 3.7 or Figure 3.11, Z_{ref} never matches exactly the characteristic impedance of the 2-wire line, so a portion of the incoming signal is fed back in the outgoing signal. This parasitic signal is the hybrid echo and has all sorts of consequences:

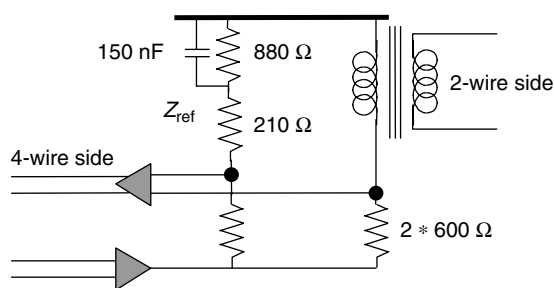


Figure 3.10 Emulating a hybrid with operational amplifiers.

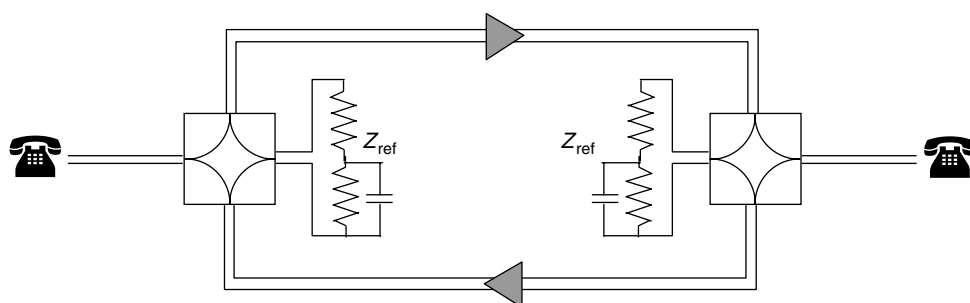


Figure 3.11 Line amplification in the 4-wire path.

- For instance, in Figure 3.11 the signals will loop between the two amplifiers and generate a ‘cathedral effect’ if the one-way delay is about 20 ms. To avoid instability in the network, a loss of 6 dB at least is introduced in the 4-wire path.
- The talker at the other end of the line will hear himself after a round trip delay (talker echo).

In many countries, the transit network is entirely built using 4 wires (any digital link is a virtual 4-wire link). Two- to 4-wire separation occurs at the local switch where the analog phone is connected. Because the echo generated at the switch end comes back to the phone undelayed, it has no effect. On the other hand, the echo generated at the phone end travels back to the other phone through the network (Figure 3.12) and is noticed as soon as the round trip time is above 50 ms (without echo cancellation in the 4-wire path).

ITU Recommendation G.165 provides more details on the handling of hybrid echo.

3.3.3 Acoustic echo

Note: In the following text we will term ‘loudspeaker phone’ an amplified phone without acoustic echo cancellation and ‘hands-free phone’ as amplified phone with acoustic echo cancellation.

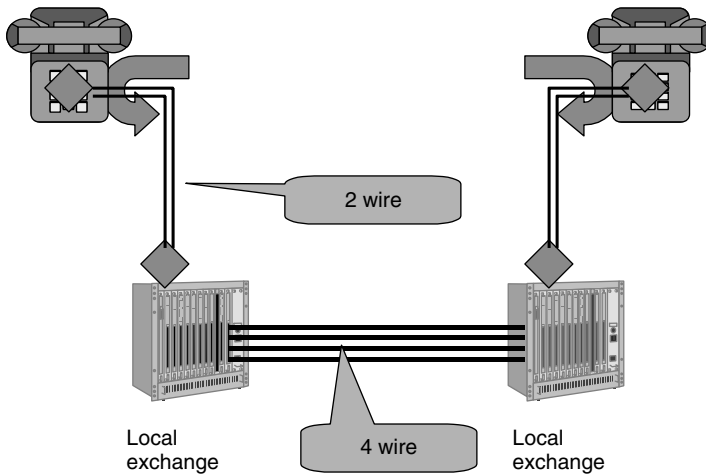


Figure 3.12 Hybrid echo.

Acoustic echo is simply that part of the acoustic signal that is fed back from the loudspeaker of a device to the microphone of that same device. Typically, acoustic echo is a parasitic signal about 10–15 dB (in the case of a loudspeaker phone) below the acoustic signal of the person actually talking into the microphone. Just like hybrid echo, such a level of acoustic echo goes unnoticed if the round trip delay is below 50 ms. After 50 ms the person at the other end of the line gets the impression of talking inside a deep well and then begins to distinctly perceive the echo of his own voice.

An easy way to suppress acoustic echo is to use a headset. However, with appropriate echo-canceling devices it is possible to reduce the power of parasitic echo to about 45 dB below the speaker's signal, even using a loudspeaker phone.

ITU recommendations G.161, G.167, and P.330 focus on acoustic echo and give some values for the typical echo path to use during the testing of echo cancelers:

- for teleconference systems, the reverberation time [time after which the sound energy of an impulse has decayed below 60 dB of the original power] averaged over the transmission bandwidth shall be typically 400 ms. The reverberation time in the highest octave shall be no more than twice this average; the reverberation time in the highest octave shall be no less than half this value. The volume of the typical test room shall be of the order of 90 m³.
- for hands free telephones and videophones, the reverberation time averaged over the transmission bandwidth shall be typically 500 ms; the reverberation time in the highest octave shall be no more than twice this average; the reverberation time in the highest octave shall be no less than half this value. The volume of the typical test room shall be of the order of 50 m³.
- for mobile radio telephones an enclosure simulating the interior of a car can be used.[...] A typical average reverberation time is 60 ms. The volume of the test room shall be 2.5 m³.

Echo cancelers usually do not work as well with acoustic echo as with electric echo, because the acoustic echo path varies much more, which makes it more difficult to dynamically adapt the synthesized echo to the real one. In particular, echo cancelers compliant with ITU recommendations G.165 performance are likely to be insufficient. Even the newer recommendation, G.168, already implemented by most vendors, may not be sufficient in some cases. Both recommendations also provide the ability to stop echo cancellation when detecting the phase reversal tone of high-speed modems.

Typical values for acoustic echo attenuation in current devices are:

- Loudspeaker phones (80% of the market): 10–15 dB.
- Hands-free phones: 35–40 dB.
- Phones with good-quality handsets: 35–40 dB.

3.3.4 How to limit echo

Two types of devices are commonly used to limit echo: echo cancelers and echo suppressors. Electric and acoustic echo reduction is measured in the 4-wire path with the reference points indicated in Figure 3.13.

3.3.4.1 Echo suppressors

Echo suppressors were introduced in the 1970s. The idea is to introduce a large loss in the send path when the distant party is talking. This technique is widely used in low-end hands-free phones, but tends to attenuate the talker when the distant party talks at the same time. It is very noticeable because the background noise that was perceived over the talker's voice by the listener suddenly disappears when he stops speaking or when the listener starts talking. It sometimes creates the impression that the line has been cut, prompting the response: 'Are you still there?'

3.3.4.2 Echo cancelers

The echo canceler functional model is shown in Figure 3.14. An echo canceler is much more complex than an echo suppressor, because it actually builds an estimate of the shape

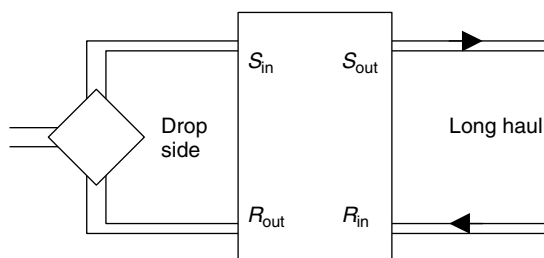


Figure 3.13 Reference points for echo measurement.

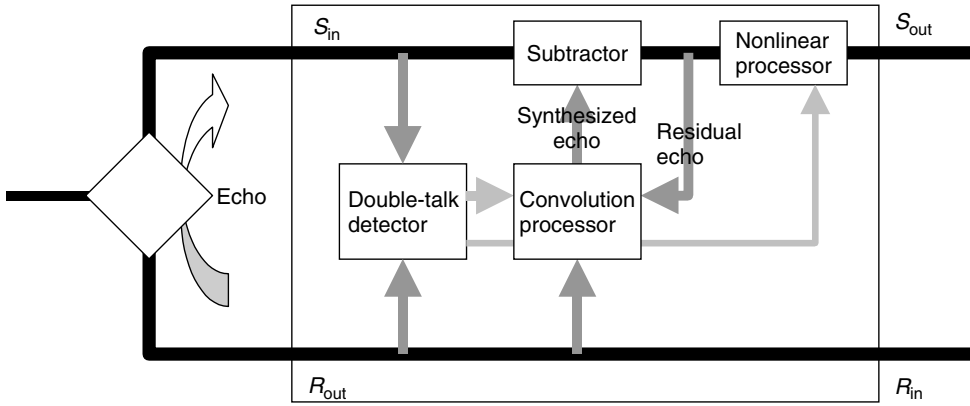


Figure 3.14 Echo canceler block diagram.

of the echo to remove it from the incoming signal. The echo is modeled as a sum of signals similar to the incoming signal, but delayed and with a lower amplitude (a convolution of the incoming signal); therefore, it only work with linear modifications of the signal between R_{in} and S_{in} (e.g., clipping will ruin the performance of an echo canceler). The error signal is measured and minimized only when the distant party is talking, which is what the double-talk detector is used for.

Echo cancelers need to store the amplitude of every input signal sample for each possible delay between 0 and the biggest reverberation delay (the impulse response of the hybrid). Therefore, echo cancelers that can handle large delays (e.g., 128 ms) on the drop side are more expensive than echo cancelers that only handle small delays: it is always best to place the echo canceler as close as possible to the source of echo.

Technically, echo cancelers are FIR (finite impulse response) adaptive digital filters placed in the network (e.g., in an international switching center for a satellite link or in the mobile switching center for MSC, for digital cellular applications). The filter (Figure 3.15) tries to get an output signal $y(k)$ that closely matches the echo signal from delayed input signal samples $x(n - k)$:

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n - k)$$

Note that the input signal must be linear and therefore must be decoded if it arrives as a G.711 encoded signal. Note also that the model of echo is linear (each $h(k)$ coefficient models a possible delay and attenuation), and therefore any nonlinearity in a network (e.g., clipping due to too high signals or VAD devices) will ruin the performance of the echo canceler.

If the echo signal is $d(n)$, then the algorithm will seek to minimize the sum of squared errors E :

$$E = \sum_{n=0}^M e(n)^2 = \sum_{n=0}^M \left[d(n) - \sum_{k=0}^{N-1} h(k)x(n - k) \right]^2$$

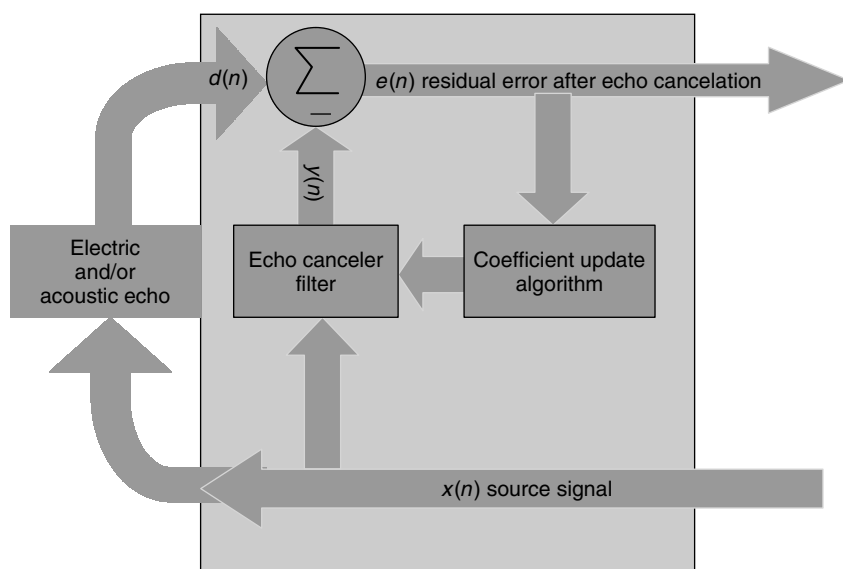


Figure 3.15 Principle behind an echo canceler.

One of the most common algorithms is the recursive least mean squares (LMS) algorithm, which computes the optimal $h(k)$ using a descent algorithm. After each new sample $x(n)$, the $h(k)$ coefficients are updated as follows, where α is the descent algorithm step size parameter:

$$h_p(k) = h_{p-1}(k) + \alpha e(n)x(n - k)$$

A larger step size accelerates convergence while slightly decreasing the quality of echo cancellation. The step size should be smaller than $1/(10 \cdot N \cdot \text{Signal power})$ to ensure stability. Signal power can be approximated by:

$$\frac{1}{M} * \sum_{n=0}^{M-1} x(n)^2$$

3.3.4.3 Usage of echo cancelers

Electric (hybrid) echo cancelers (EECs) are also called line echo cancelers. They are inserted right after the hybrid, located between the 4-wire section of the network (the packetized network in the case of VoIP) and the 2-wire portion (Figure 3.16).

Acoustic echo cancelers are usually implemented in the phone itself.

Many national PSTN networks do not have line echo cancelers due to the relatively small transmission delays. Telephony networks that introduce longer delays can be connected to such PSTNs only through line echo cancelers. For example, in the GSM system, one-way delay is around 100 ms due to:

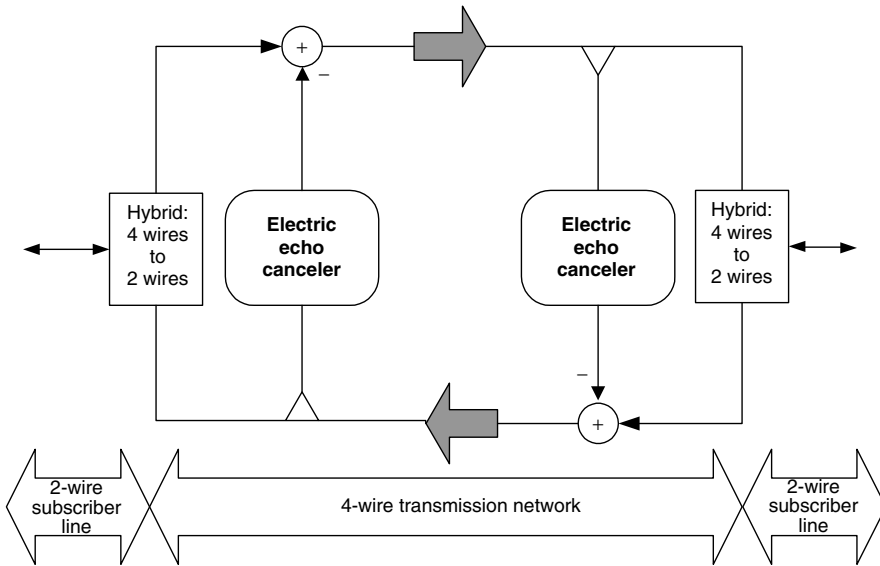


Figure 3.16 Insertion of echo cancelers in a network.

- A frame length of 20 ms.
- A processing delay of about 20 ms (depending on the handset's DSP).
- Interleaving for channel protection.
- Buffering and decoding.

So, an EEC must be included in the mobile switching center (MSC) as shown in Figure 3.17.

The situation is quite similar to that of a VoIP network, where line echo cancellation must be done in the VoIP gateways connected to the PSTN. If line echo cancellation is of insufficient quality, the user on the IP side will hear echo.

VoIP devices can also introduce acoustic echo. The worst examples are PCs with older VoIP software (without acoustic echo cancellation, or AEC). These PCs must be used with headsets in order to reduce echo as much as possible. Note that many headsets are not designed for this (e.g., many headsets have a microphone attached to one of the side speakers, allowing mechanical transmission of speaker vibrations to introduce echo). Some high-end active headsets, as well as dedicated soundboards, now include an AEC module, but the more recent PC VoIP software is now capable of performing the AEC algorithm, making it possible to use standard headsets or even have a hands-free conversation (Figure 3.18).

In a VoIP to PSTN call, if the AEC of the IP phone or the PC is insufficient, echo will be heard at the PSTN end.

The performance of an echo canceler involves many parameters (see G.168 for more details). The most important are echo return loss enhancement, or ERLE (in dB), the amount by which the echo level between the S_{in} and S_{out} port is reduced (see Figure 3.13),

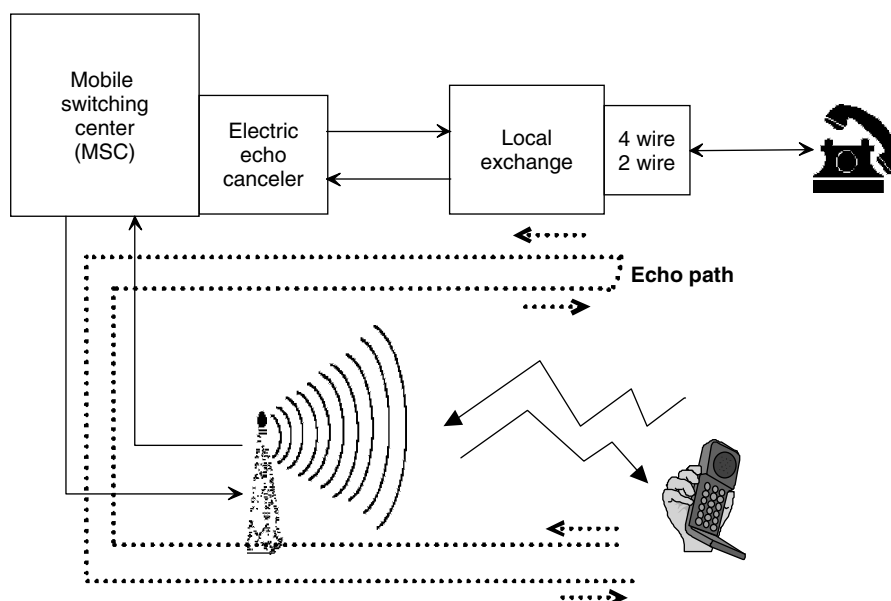


Figure 3.17 EEC required at the interface with the cellular network.

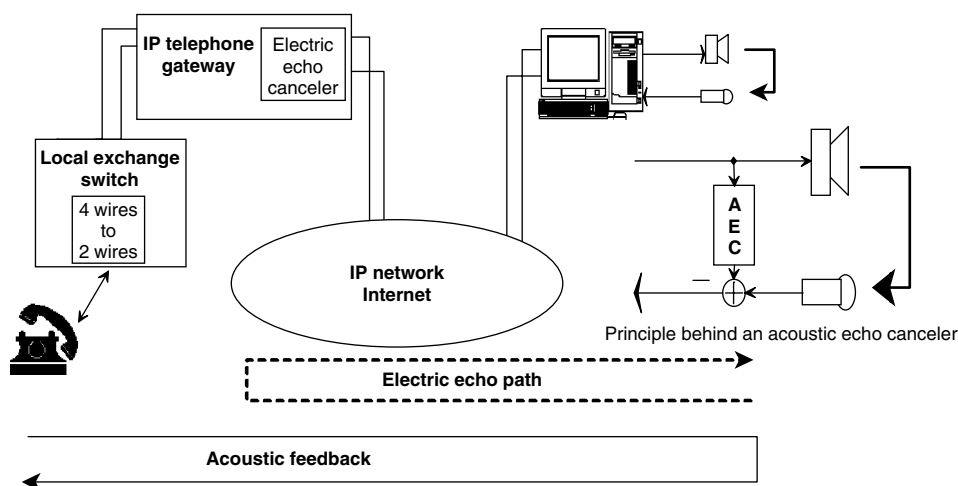


Figure 3.18 Softphones have to provide acoustic echo cancellation.

and the size of the window modeling the impulse response (some echo cancelers are optimized to cancel all echoes coming with a delay of 0 to T_{\max} , some echo cancelers are optimized to model only echoes coming with a delay of T_{\min} to T_{\max}). Other parameters include convergence time and quality of double-talk detection.

3.4 Delay

3.4.1 Influence of the operating system

Most IP phone applications are just regular programs running on top of an operating system, such as Windows. They access sound peripherals through an API (e.g., the Wave API for Windows), and they access the network through the socket API.

As you speak the sound card samples the microphone signals and accumulates samples in a memory buffer (it may also perform some basic compression, such as G.711 encoding). When a buffer is full the sound card tells the operating system, using an interrupt, that it can retrieve the buffer, and stores the next samples in a new buffer, etc.

Interrupts stop the regular activities of the operating system and trigger a very small program called an interrupt handler which in our case may simply store a pointer to the sound buffer for the program that has opened the microphone.

The program itself, in the case of the Wave API, registered a callback function when it opened the microphone to receive the new sample buffers, and the operating system will simply call this function to pass the buffer to our IP phone application.

When the callback function is enacted, it will check that there are enough samples to form a full frame for a compression algorithm, such as G.723.1, and if so put the resulting compressed frame (wrapped with the appropriate RTP information) on the network using the socket API.

The fact that samples from the microphone are sent to the operating system in chunks using an interrupt introduces a small accumulation delay, because most operating systems cannot accommodate too many interrupts per second. For Windows many drivers try not to generate more than one interrupt every 60 ms. This means that on such systems the samples come in chunks of more than 60 ms, independent of the codec used by the program. For instance, a program using G.729 could generate six G.729 frames and a program using G.723.1 could generate two G.723.1 frames for each chunk, but in both cases the delay at this stage is 60 ms, due only to the operating system's maximum interrupt rate.

The self same situation occurs when playing back the samples, resulting in further delays because of socket implementation.

The primary conclusion of this subsection is that the operating system is a major parameter that must be taken into account when trying to reduce end-to-end delays for IP telephony applications. To overcome these limitations most IP telephony gateways and IP phone vendors use real-time operating systems, such as VxWorks (by Wind River Systems) or Nucleus, which are optimized to handle as many interrupts as needed to reduce this accumulation delay.

Another way of bypassing operating system limits is to carry out all the real-time functions (sample acquisition, compression, and RTP) using dedicated hardware and only carry out control functions using the non-real-time operating system. IP telephony board vendors, such as Natural Microsystems, Intel, or Audiocodes, use this type of approach to allow third parties to build low-latency gateways with Unix or Windows on top of their equipment.

3.4.2 The influence of the jitter buffer policy on delay

An IP packet needs some time to get from A to B through a packet network. This delay $t_{AB} = t_{\text{arrival}} - t_{\text{departure}}$ is composed of a fixed part L characteristic of average queuing and propagation delays and a variable part characterizing jitter as caused by the variable queue length in routers and other factors (Figure 3.19).

Terminals use jitter buffer to compensate for jitter effects. Jitter buffer will hold packets in memory until $t_{\text{unbuffer}} - t_{\text{departure}} = L + J$. The time of departure of each packet is known by using the time stamp information provided by RTP. By increasing the value of J , the terminal is able to resynchronize more packets. Packets arriving too late ($t_{\text{arrival}} > t_{\text{unbuffer}}$) are dropped.

Terminals use heuristics to tune J to the best value: if J is too small too many packets will be dropped, if J is too large the additional delay will be unacceptable to the user. These heuristics may take some time to converge because the terminal needs to evaluate jitter in the network (e.g., the terminal can choose to start initially with a very small buffer and progressively increase it until the average percentage of packets arriving too late drops below 1%). For some terminals, configuration of the size of jitter buffer is static, which is not optimal when network conditions are not stable.

Usually endpoints with dynamic jitter buffers use the silence periods of received speech to dynamically adapt the buffer size: silence periods are extended during playback, giving more time to accumulate more packets and increase jitter buffer size, and vice-versa. Most endpoints now perform dynamic jitter buffer adaptation, by increments of 5–10 ms.

A related issue is clock skew, or clock drift. The clocks of the sender and the receiver may drift over time, causing an effect very similar to jitter in the network. Therefore, IP phones and gateways should occasionally compensate for clock drift in the same way they compensate for network jitter.

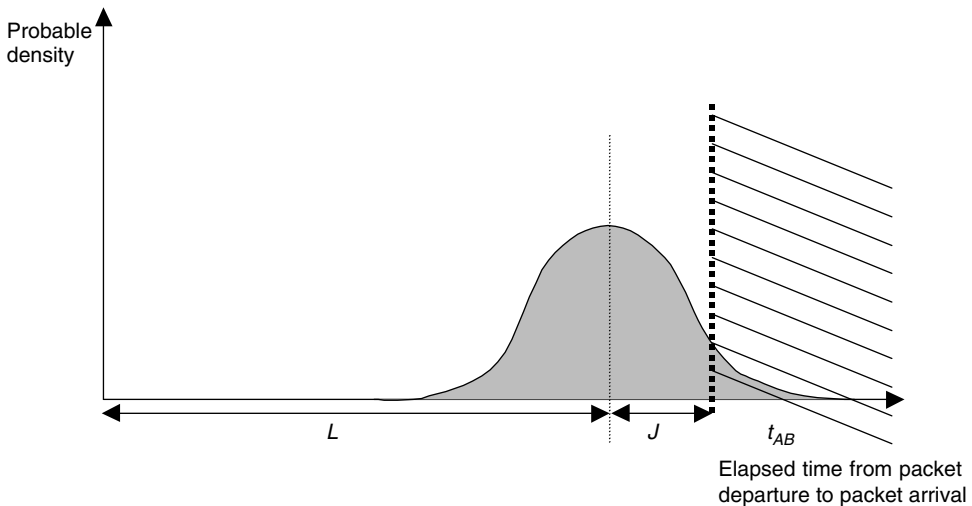


Figure 3.19 Influence of jitter buffer size on packet loss.

3.4.3 The influence of the codec, frame grouping, and redundancy

3.4.3.1 Frame size, number of frames per packet

Most voice coders are frame-oriented; this means that they compress fixed size chunks of linear samples, rather than sample per sample (Figure 3.20). Therefore, the audio data stream needs to be accumulated until it reaches chunk size, before being processed by the coder. This sample accumulation takes time and therefore adds to end-to-end delay. In addition, some coders need to know more samples than those already contained in the frame they will be coding (this is called **look-ahead**).

Therefore, in principle, the codec chosen should have a short frame length in order to reduce delays on the network.

However, many other factors should be taken in consideration. Primarily, coders with larger frame sizes tend to be more efficient, and have better compression rates (the more you know about something the easier it is to model it efficiently). Another factor is that each frame is not transmitted ‘as is’ through the network: a lot of overhead is added by the transport protocols themselves for each packet transmitted through the network. If each compressed voice frame is transmitted in a packet of its own, then this overhead is added for each frame, and for some coders the overhead will be comparable if not greater than the useful data! In order to lower the overhead to an acceptable level, most implementations choose to transmit multiple frames in each packet; this is called ‘bundling’ (Figures 3.21).

If all the frames accumulated in the packet belong to the same audio stream, this will add more accumulation delay. In fact, using a coder with a frame size of f and three frames per packet is absolutely equivalent, in terms of overhead and accumulation

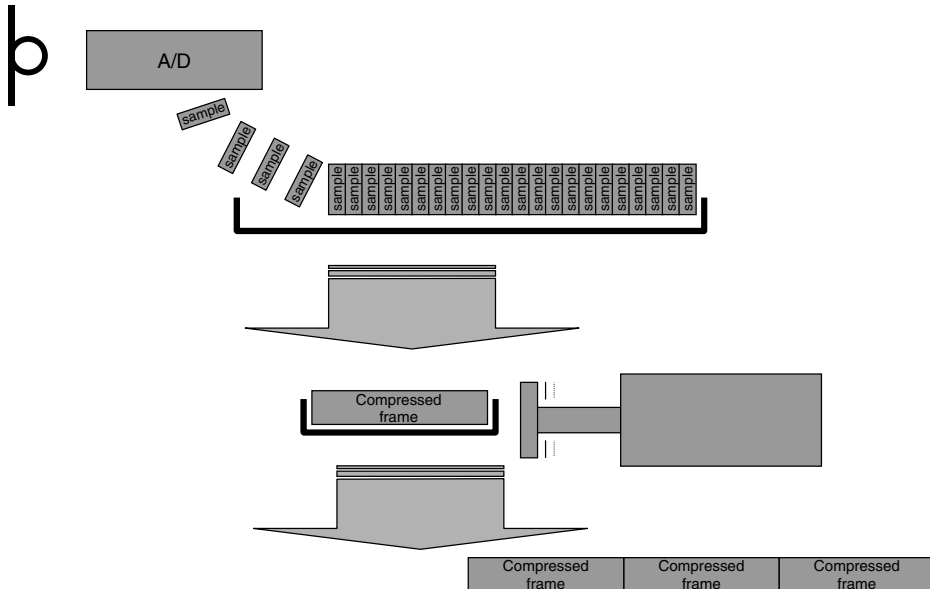


Figure 3.20 Concept of audio codec ‘frames’.

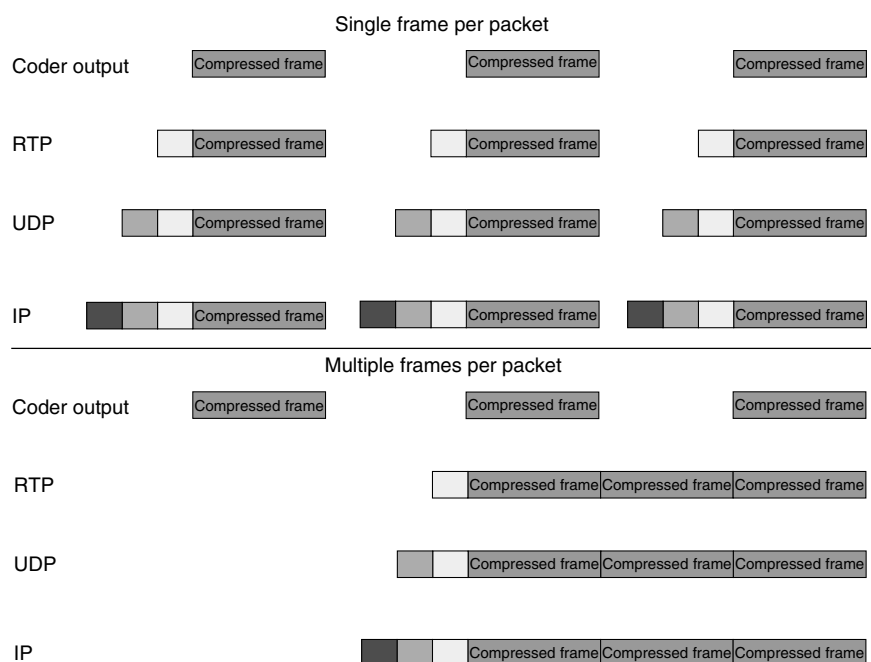


Figure 3.21 Influence of bundling on overhead.

delay, to using a coder with a frame size of $3f$ and one frame per packet. Since a coder with a larger frame size is usually more efficient, the latter solution is likely to be more efficient also.

Note that if the operating system gives access to the audio stream in chunks of size C ms rather than sample per sample (see Section 3.4.1), then samples have already been accumulated and using a coder with a larger frame size f introduces no additional overhead as long as $f < C$.

A much more intelligent way of stacking multiple frames per packet in order to reduce overhead without any impact on delay is to concatenate frames from different audio streams, but with the same network destination, in each packet. This situation occurs frequently between corporate sites or between gateways inside a VoIP network. Unfortunately, the way to do this RTP-multiplexing (or RTP-mux) has not yet been standardized in H.323, SIP, or other VoIP protocols. The recommended practice is to use TCRTP (tunneling-multiplexed compressed RTP), an IETF work-in-progress which combines L2TP (Layer 2 Tunneling Protocol, RFC 2661), multiplexed PPP (RFC 3153), and compressed RTP (RFC 2508, see ch. 4 p. 176).

3.4.3.2 Redundancy, interleaving

Another parameter that needs to be taken into account when assessing the end-to-end delay of an implementation is the redundancy policy. A real network introduces packet

loss, and a terminal may use redundancy to be able to reconstruct lost frames. This can be as simple as copying a frame twice in consecutive packets (this method can be generalized by generating, after each group of N packets, a packet containing the XORed value of the previous N packets, in which case it is called FEC, for forward error correction) or more complex (e.g., interleaving can be used to reduce sensitivity to burst packet loss).

Note that redundancy should be used with care: if packet loss is due to congestion (the most frequent case), redundancy is likely to increase the volume of traffic and as a consequence increase congestion and network packet loss rate. On the other hand, if packet loss was due to an insufficient switching capacity (this is decreasingly likely in recent networks, but may still occur with some firewalls), adding redundancy by stacking multiple redundant frames in each packet will not increase the number of packets per second and will improve the situation.

Redundancy influences end-to-end delay because the receiver needs to adjust its jitter buffer in order to receive all redundant frames before it transfers the frame to the decoder (Figure 3.22). Otherwise, if the first frame got lost, jitter buffer would be unable to wait until it has received the redundant copies, and they would be useless! This can contribute significantly to end-to-end delay, especially if the redundant frames are stored in noncontiguous packets (interleaving) in order to resist correlated packet loss. For this reason this type of redundancy is generally not used for voice, but rather for fax transmissions which are less sensitive to delays.

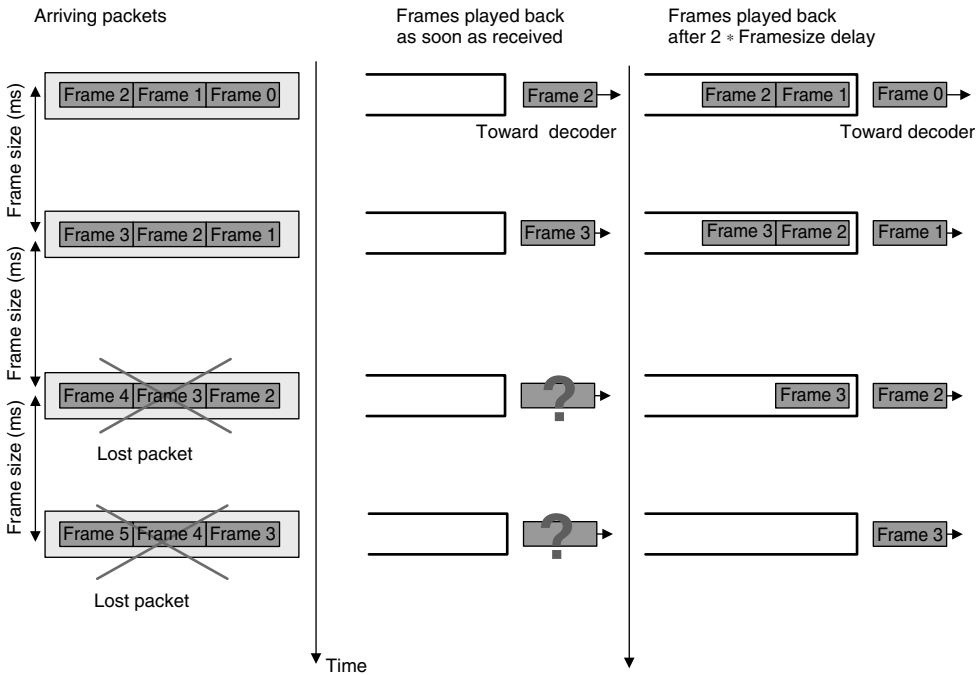


Figure 3.22 FEC or interleaving type of redundancy only works when there is an additional delay at the receiver buffer.

3.4.4 Measuring end-to-end delay

In order to assess the delay performance of IP telephony hardware or software, it is necessary to simulate various network conditions, characterized by such parameters as average transit delay, jitter, and packet loss. Because of the many heuristics used by IP telephony devices to adapt to the network, it is necessary to perform the end-to-end delay measurement after allowing a short convergence time. A simple measurement can be done according to the following method:

- (1) IP telephony devices' IP network interfaces are connected back to back through a network simulator.
- (2) The network simulator is set to the proper settings for the reference network condition considered for the measurement; this includes setting the average end-to-end delay L , the amount of jitter and the jitter statistical profile, the amount of packet loss and the loss profile, and possibly other factors, such as packet desequencing. Good network simulators are available on Linux and FreeBSD (e.g., NS2: www.isi.edu).
- (3) A speech file is fed to the first VoIP device with active talk during the first 15 s (Talk1), then these follows a silence period of 5 s, then active talk again for 30 s (Talk2), then a silence period of 10 s.
- (4) The speech file is recorded at the output of the second VoIP device.
- (5) Only the Talk2 part of the initial file and the recorded file is kept. This gives the endpoint some time to adapt during the silence period if a dynamic jitter buffer algorithm is used.
- (6) The average level of both files is equalized.
- (7) If the amplitude of the input signal is $IN(t)$, and the amplitude of the recorded signal is $OUT(t)$, the value of D maximizing the correlation of $IN(t)$ and $OUT(t + D)$ is the delay introduced by the VoIP network and the tested devices under measurement conditions. The correlation can be done manually with an oscilloscope and a delay line, adjusting the delay until input and output speech samples coincide (similar envelopes and correlation of energy), or with some basic computing on the recorded files (for ISDN gateways the files can be input and recorded directly in G.711 format).

The delay introduced by the sending and receiving devices is $D - L$, since L is the delay that was introduced by the network simulator. With this method it is impossible to know the respective contributions to the delay from the sending and the receiving VoIP devices.

Note that a very crude, but efficient way of quickly evaluating end-to-end delay is to use the Windows sound recorder and clap your hands (Figure 3.23). The typical mouth-to-ear delay for an IP phone over a direct LAN connection is between 45 ms and 90 ms, while VoIP softphones are in the 120 ms (Windows XP Messenger) to 400 ms range (NetMeeting and most low-end VoIP software without optimized drivers).

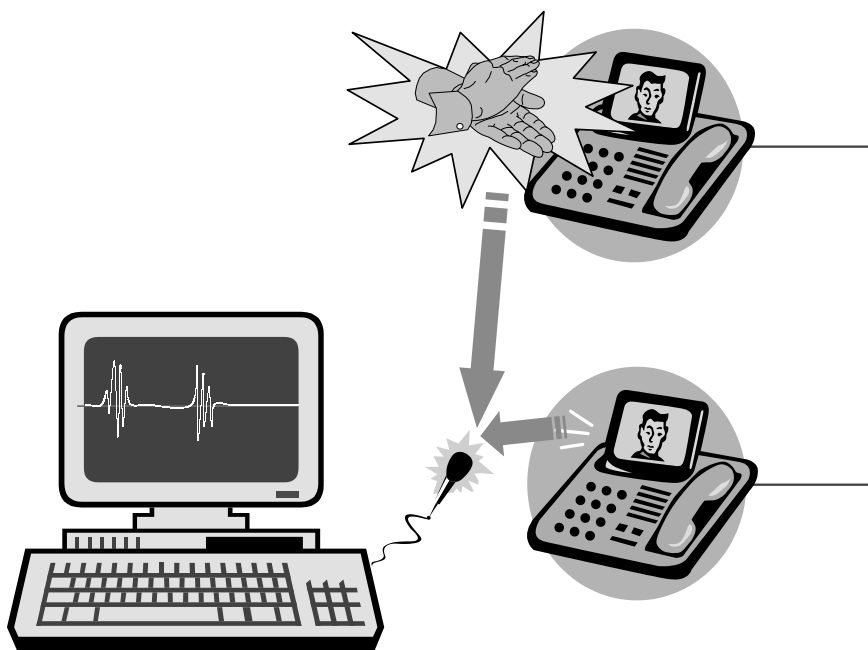


Figure 3.23 Poor man's delay evaluation lab.

3.5 Acceptability of a phone call with echo and delay

3.5.1 The G.131 curve

The degree of annoyance of talker echo depends on the amount of delay between the original signal and the echo, and on the attenuation of the echo signal compared with the original. This attenuation is characterized by the 'talker echo loudness rating' (TELR) as described in G.122 and annex A/G.111. A higher value of TELR represents better echo cancellation. Note that, because of quantization noise on the original signal, it is impossible to achieve a perfect echo cancellation (typically about 55 dB at best).

G.131 provides the minimum requirements on TELR as a function of the mean one-way transmission time T . According to G.131 (Figure 3.24), conditions are generally acceptable when less than 1% of the users complain about an echo problem. The second curve, where 10% of users complain, is an extreme limit that should be allowed only exceptionally.

Figure 3.24 clearly shows that echo becomes more audible as delay increases. This is the reason echo is such a problem in all telephony technologies that introduce high delays. This is the case for most packet voice technologies, for networks that use interleaving for error protection (e.g., cellular phones), and for satellite transmissions in general.

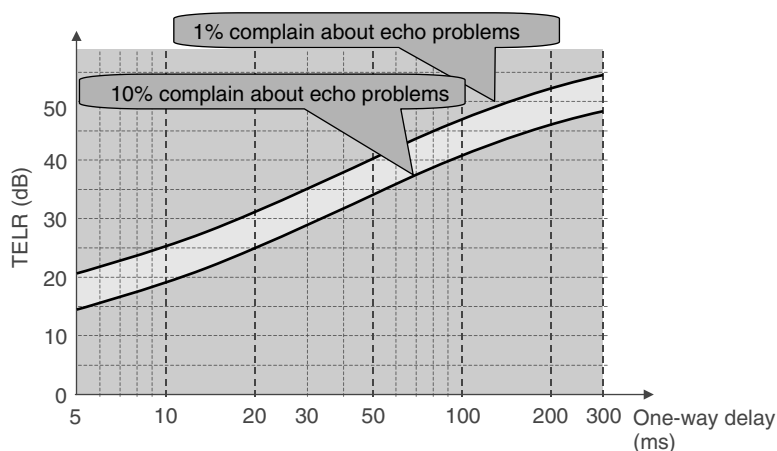


Figure 3.24 G.131 one-way delay versus echo perception.

3.5.2 Evaluation of echo attenuation (TELR)

3.5.2.1 Overview of signal level measurement (dB, dBr, dBm0, etc.)

A discussion of units can be found in G.100 annex A. Here is a short summary:

- **Relative power** is measured in *dB*. A signal of P_1 mW is at level L dB compared with a signal of P_2 mW if $L = 10 \log_{10}(P_1/P_2)$. For relative voltages, currents, or acoustic pressure, the formula uses a multiplicative factor of 20 instead of 10 (power depends on the square of voltage/current or pressure).
- **dBm** refers to a power measurement in dB relative to 1 mW.
- **dBr** is used to measure the level of a reference 1,020-Hz signal at a point compared with the level of that same reference signal at the reference point (the 0-dBr point). For instance, if the entrance of an *2 amplifier (Figure 3.25) is the 0-dBr point, the output is a +3-dBr point. Digital parts of the network are by convention at 0 dBr (unless digital gain or loss is introduced). To determine the dBr level at the analog end of a

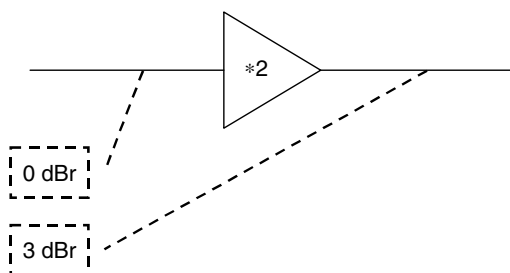


Figure 3.25 dBr levels at the input and output of a *2 amplifier.

coder or decoder, G.101 defines a digital reference sequence (DRS). When decoding the DRS, if the output of the decoder is at R dBm, then it is an R -dBr point.

- **dBm0** is used to measure the absolute level in dBm which a signal would have when passing through the reference point (0-dBr point). For instance, if the power of a signal at the output of the Figure 3.25 amplifier is 10 dBm, then it is a 7-dBm0 signal.

3.5.2.2 TELR for analog and digital termination lines

Recommendation G.131 uses the reference circuit of Figure 3.26 to evaluate talker echo attenuation. The send loudness rating (SLR) and receive loudness rating (RLR) model the acoustic-to-electric efficiency of the emitter and the receiver, respectively (see ITU recommendation P.79). For typical phone sets G.121 states that $SLR_{\text{target}} = 7$ dB, $SLR_{\text{min}} = 2$ dB, $RLR_{\text{target}} = 3$ dB, $RLR_{\text{min}} = 1$ dB. For digital phones, the recommended values are $SLR = 8$ dB and $RLR = 2$ dB.

For an analog phone at the distant side $TELR = SLR + RLR + R + T + L_r$, where R and T stand for additional loss introduced in the analog circuit in order to have a 0-dBr point at the exchange. Most analog phone connections have an $L_r > 17$ dB for an average length of subscriber cable; however, in some networks it can be 14 dB with a standard deviation of 3 dB. In many networks $R + T = 6$ dB.

For a digital phone at the distant side $TELR = SLR + RLR + TCL$, where TCL is terminal coupling loss. IP phones are digital phones. For software phones the values of SLR and RLR can be affected by sound card settings (microphone volume, speaker volume), and properly implemented software should apply digital attenuation to make sure that the resulting SLR and RLR provide the recommended values for the voice level in the VoIP network. Most digital handsets have a TCL of 40–46 dB, although lower end phones may have a TCL as low as 35–40 dB.

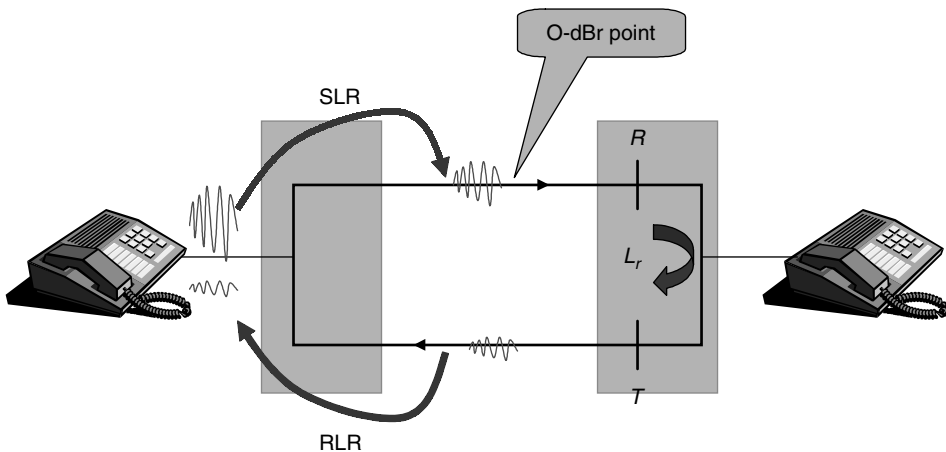


Figure 3.26 Parameters influencing TELR.

When the intrinsic TELR value of the network is too low for the expected network delay, an echo canceler must be added (in the handset for acoustic echo, in the network for line echo) in order to increase the resulting TELR to a value that is acceptable.

3.5.2.2.1 Examples

For $SLR = 7$ dB, $RLR = 3$ dB, $L_r = 14$ dB, $R + T = 6$ we get a TELR of 30 which leads to an acceptable limit for the one-way delay of 18 ms (33 ms in the limiting case). For a 'loud' telephone set with $SLR = 2$ dB, $RLR = 1$ dB, and an L_r of 8 dB we get a TELR of 17 and the limiting case is now 7 ms!

When ringing a digital handset ($TCL = 45$ dB) with the talker's phone at $SLR = 7$ dB, $RLR = 3$ dB we get a TELR of 55 dB and the one-way delay is 'acceptable' up to 400 ms regarding echo perception (but such a one-way delay is already impacting the interactivity of the conversation).

3.5.2.3 VoIP circuits

The IP telephony circuit is subject to the same echo/delay constraints as any other telephony technology. With current IP technology, delays of 200–300 ms for one-way transmission are still common over wide area networks. Other delay factors (encoding delay, jitter buffers, etc.) may add as much as 100 ms. Therefore, all VoIP networks require state-of-the-art echo cancellation, with a TELR value of at least 55 dB. Note that this is close to the highest achievable value for G.711 encoded voice signals, due to the quantization noise that is introduced by G.711. With most echo cancelers, this echo cancellation level can be reduced to about 30 dB under double-talk conditions.

3.5.2.3.1 IP software phone to IP phone or IP software phone

In this case if we assume $SLR + RLR = 10$ dB, then the echo loss of the distant IP phone must be at least $TCL = 45$ dB. On a software phone, this might be implemented in the audio peripherals (soundboard, headset) or by the IP telephony software itself.

3.5.2.3.2 IP phone to a regular phone

3.5.2.3.2.1 IP phone to digital or cellular phone At the ISDN phone end, only acoustic echo is generated since there is no hybrid. Most ISDN phones have a TCL value of 45 dB, so the IP telephony gateway does not need to perform echo cancellation at the ISDN phone end if the connection is digital end to end (this is rarely the case, except in Germany).

Obviously, the IP phone needs to have an echo canceler as well, otherwise the digital phone user will hear echo.

In the early days of VoIP, many gateway demonstrations made phone calls to ISDN or cellular phones. In the case of cellular phones, some vendors even explained that this was the worst case scenario because, after all, you were calling a cellular phone. In fact, this was done on purpose to hide the lack of an echo-canceling algorithm in the IP telephony gateway! The cellular phone itself is a 4-wire device (no electric echo) and includes a

powerful acoustic echo canceler. The cellular phone network interface with the regular phone network is also made via echo cancelers.

3.5.2.3.2.2 IP phone to PSTN user In this case the PSTN phone will generate hybrid echo and acoustic echo. Since propagation time in the PSTN is usually low, many national links may not implement sufficient echo cancellation (if implemented at all). Therefore, the gateway must implement echo cancellation. In some cases there will already be an echo canceler in the PSTN path, which may cause some signal degradation (e.g., voice clipping), but even such degradation is preferable to the risk of not having any echo cancellation.

Canceling electric and acoustic echo is difficult because their characteristics in terms of attenuation and, more importantly, delay are very different. Acoustic echo signal components are typically spread over about 50 ms (office environment), while electric echo signals are typically spread over 13 ms. Echo cancelers are often characterized by the maximum skew between the signals that compose the echo. This signal is a superposition of signals s_i that are a copy of the original signal but attenuated by a factor A_i and delayed by a factor of $D + d_i$. D is introduced by the voice transport network between the echo canceler and the source of echo. If a gateway is implemented in a country the size of France, for instance, D is below 64 ms in 90% of the cases, including call rerouting.

Some echo cancelers optimized for use in corporate equipment only work with $D = 0$ and $0 < d_i < \text{Max skew}$ (e.g., 18 ms). Some network echo cancelers can work with D as large as 500 ms and $0 < d_i < \text{Max skew}$. Only variation of the delay (maximum skew) requires memory in the echo canceler. Since most echo cancelers are implemented as FIR filters on signals that were originally G.711 signals, the memory (therefore, variation of the delay) supported by the echo canceler is sometimes mentioned as ‘taps’ (i.e., a memory cell for one sample, or 0.125 ms). An acoustic echo canceler requires more than 400 ‘taps’ (50 ms), while a line echo canceler requires about 100 ‘taps’ (12.5 ms). Most VoIP gateways have an echo canceler with a memory of at least 32 ms, (many go up to 64 or even 128 ms), and most of them only cancel hybrid echo, which explains why some echo can still be heard sometimes when talking to people with low-quality loudspeaker phones.

Note that it should be possible to disable this echo canceler, either statically (if the gateway is connected to a network already performing echo cancellation) or dynamically (if a modem connection is detected, because modems perform their own echo cancellation as required by recommendation G.168).

3.5.3 Interactivity

In the previous examples the term ‘acceptable’ only considered echo. Interactivity is another factor that must also be considered. Usually, a delay below 150 ms one-way provides good interactivity. A one-way delay between 150 ms and 300 ms provides acceptable interactivity (satellite hop). A one-way delay in excess of 400 ms should be exceptional (in the case of two satellite hops it is about 540 ms) and is the limit after

Table 3.1 ITU interactivity classes

Class	One-way delay (ms)	
1	0–150	Acceptable for most conversations. Only the most interactive tasks will perceive a substantial degradation
2	150–300	Acceptable for communications with low interactivity (communication over satellite link)
3	300–700	Conversation becomes practically half-duplex
4	Over 700	Conversation impossible without some training to half-duplex communications (military communication)

Table 3.2 Communication impairment caused by one-way delay

One-way delay (ms)	Index of communication impairment (%)
200	28
450	35
700	46

which the conversation can be considered half-duplex. ITU recommendation G.114 lists classes of interactivity and quality as a function of delay (Tables 3.1 and 3.2).

When there are large delays on the line, the talker tends to think that the listener has not heard or paid attention. He will repeat what he said and be interrupted by the delayed response of the called party. Both will stop talking . . . and restart simultaneously! With some training it is quite possible to communicate correctly, but the conversation is not natural.

3.5.4 Other requirements

3.5.4.1 Average level, clipping

Gateways and transcoding functions to the PSTN should implement automatic-level control to respect ITU recommendation G.223: ‘The long term average level of an active circuit is expected to be –15 dBm0 including silences. The average level during active speaking periods is expected to be –11 dBm0.’ The methodology for measuring active speech levels can be found in ITU recommendation P.56.

Note that PCM coding is capable of handling a maximum level of +3.14 dBm0 in the A law (+3.17 dBm0 in the μ law). The gateways should absolutely avoid clipping, since this would adversely disturb the echo cancelers in the network (introduction of nonlinearities). As the average-to-peak power ratio of voice signals is about 18 dB, this imposes an average level not exceeding –15 dB.

Even IP phones and software phones should respect the average levels expected on a transmission line. Since microphone sensitivity can be adjusted on most operating systems, software phones should adjust accordingly to avoid sending too high or too low signals over the VoIP network.

3.5.4.2 Voice activity detection

VAD algorithms are responsible for switching the coder from ‘active speech mode’ to ‘background noise transmission mode’ (this can also be ‘transmit nothing mode’). If they are not implemented properly these algorithms may clip parts of active speech periods: beginning of sentences, first syllables of words, etc.

A general guideline for a good VAD algorithm is to keep the duration of clipped segments below 64 ms and have no more than 0.2% of the active speech clipped. These guidelines are part of ITU recommendation G.116. More detailed information is available in ‘Subjective effects of variable delay and speech loss in dynamically managed systems’, J. Gruber and L. Strawczynski, *IEEE GLOBECOM* ’82, 2, pp. 7.3.1–7.3.5.

3.5.5 Example of a speech quality prediction tool: the E-model

The E-model was originally developed in ETSI for the needs of network planning and later adopted by the ITU as recommendation G.107. It allows the subjective quality of a conversation as perceived by the user to be evaluated. The E-model appraises each degradation factor on perceived voice quality by a value called an ‘impairment factor’. Impairment factors are then processed by the E-model which outputs a rating R between 0 and 100. The R value can be mapped to a mean opinion score, conversational quality evaluation (MOS_{CQE}) value between 1 and 5, or to percent good or better (%GoB), or to percent poor or worse (%PoW) values using tables. An R value of 50 is very bad, while an R value of 90 is very good.

The E-model takes into account parameters that are not considered in the G.131 curve (Figure 3.24), such as the quality of the voice coder (degradation factor I_e) and frame loss (degradation factor B_{p1}). Most voice coders have been rated for their impairment factor without frame loss, and consequently the E-model (available as commercial software from various vendors) can readily be used to evaluate perceived voice quality through an IP telephony network with no packet loss and low jitter. This work was published by the T1A1.7 committee in January 1999.

Impairment factor parameters are evaluated from real subjective tests to calibrate the model (e.g., see G.113). Therefore, the usability of the E-model for a particular technology depends on how much calibration has been done previously on this technology. The E-model is only useful if it is used correctly. An impairment factor for a coder measured under specific loss profiles is not valid for other loss profiles (e.g., if there is correlated loss). IP telephony introduces many new types of perturbations that do not exist on traditional networks, such as the delay variation that may be introduced by endpoints trying to dynamically adjust the size of jitter buffers, voice clipping introduced by VAD algorithms, or correlated loss introduced by frame grouping. Some R&D labs that specialize

in voice quality and network planning have released new versions of the E-model with specific support for IP telephony degradations. This should lead to an update of the ITU specification in 2005 (currently known as P.VTQ).

One of the interesting aspects of the E-model is that it also takes into account psychological parameters that do not influence absolute voice quality, but the *perception* of the user. For instance, the ‘expectation’ impairment factor takes into account the fact that most users expect to have degraded voice quality when using a cellular phone, and therefore will be more indulgent ... and complain less, for the same level of quality, than if they had been using a normal phone and vice versa: if a cellular phone achieves similar voice quality to a normal fixed phone, many users will actually find it better than the normal phone. IP phone manufacturers may have to find a recognizable design if they want to benefit from the ‘VoIP expectation factor’!

3.6 Conclusion

In many ways, IP phone networks and mobile phone networks (such as GSM) face similar constraints regarding voice quality. The main issue in both systems is to correctly control echo, minimize the degradations introduced by packet loss, and preserve good interactivity of speech.

In early VoIP systems, all three factors were a problem and, unfortunately, this brought about a bad perception of VoIP which still persists today:

- Packet loss and delay: in 1998, the Internet was still perceived as a gadget by most incumbent carriers; as a result there was typically too little capacity installed for IP traffic. In addition, IP networks were frequently built on frame relay networks, which introduced long delays (frame relay switches often have large buffers). This situation was reversed after the Internet bubble. The backbones carrying the IP traffic today use state-of-the-art technologies, such as MPLS, and are capable of handling large volumes of traffic with minimal delay. As IP traffic dominates any other kind of data, fewer and fewer encapsulation layers are used to transport IP packets. The frame relay transport layer has been completely abandoned in core backbones, unable to offer the required capacity and performance. Even the intermediary ATM transport layer is becoming a problem at the speed at which many backbones operate today. Many backbones now carry IP packets directly on top of a layer 2 technology, such as SDH. Finally, most VoIP gateways and IP phones now implement sophisticated packet loss concealment methods which can efficiently mask up to two consecutive frames lost, making the occasional lost packet less perceptible.
- Echo control: early VoIP gateway implementations often had low-quality echo canceler implementations. With the consolidation of the telecom market, most gateway manufacturers are either previous DCME equipment vendors (compression equipment for submarine cables) with extensive know-how in signal processing and echo cancellation, or companies using the reference algorithms proposed by their DSP vendors (during the telecom bubble, most DSP vendors either developed internally or acquired companies

that had a lot of know-how in signal-processing algorithms). As a result the quality of echo cancelation in VoIP gateways and IP phones is now much better, reaching the level of echo attenuation required for long-delay networks.

In early VoIP systems, mostly targeted for the prepaid market, there was a lot of attention paid to proprietary redundancy schemes or supposedly high-performance proprietary coders. This trend has now come to an end for many reasons. Many redundancy schemes were promoted for marketing reasons, but didn't perform as advertised in real networks. More fundamentally, as VoIP developed beyond the prepaid market, into business trunking (connection of corporate PBXs to VoIP backbones), residential telephony or IP Centrex, the key requirement became interoperability. VoIP networks using proprietary schemes were unable to evolve and offer new applications, and many disappeared.

Whether or not VoIP networks need to pay attention once more to redundancy algorithms and high-performance low-bitrate coders is debatable. Most wired IP networks now support differentiated quality of service, which means VoIP transmissions enjoy very low packet loss even without any redundancy in the RTP stream. For wireless networks (e.g., UMTS), the trend is to provide various levels of error protection directly at the physical level, dynamically for each type of stream (for IP streams, desired transport layer behavior may be signaled via DiffServ marks).

Future work on voice quality on IP networks will focus on providing better than toll quality on wired lines (such as wide-band voice or stereo/spatialized voice) and improving the quality of voice over IP on wireless networks through tight integration with the QoS mechanisms of the physical layers of UMTS or WiFi networks.

3.7 Standards

[G.100, ITU]	Definitions used in recommendations on general characteristics of international telephone connections and circuits
[G.107, ITU]	The E-model, a computational model for use in transmission planning.
[G.113, ITU]	Transmission impairments.
[G.114, ITU]	One-way transmission time.
[G.116, ITU]	Transmission performance objectives applicable to end-to-end international transmissions
[G.122, ITU]	Influence of national systems on stability and talker echo in international connections.
[G.131, ITU]	Control of talker echo.

[G.122, ITU]	Influence of national systems on stability and talker echo in international connections
[G.111, ITU]	Loudness ratings in an international connection.
[G.168, ITU]	Digital network echo cancelers.
[G.167, ITU]	Acoustic echo controllers.
[G.165, ITU]	Echo cancelers.
[G.174, ITU]	Transmission performance objectives for terrestrial digital wireless systems using portable terminals to access the PSTN
[P.310, ITU]	Transmission characteristics for telephone band (300–3,400 Hz) digital telephones.
[P.79, ITU]	Calculation of loudness ratings for telephone sets.
[P.56, ITU]	Objective measurement of active speech levels.
[P.11, ITU]	Effect of transmission impairment.
[G.175, ITU]	Transmission planning for private/public network interconnection of voice traffic
[G.173, ITU]	Transmission planning aspects of the speech service in digital public land mobile networks
[G.111, ITU]	Loudness ratings in an international connection
[IEEE]	Subjective effects of variable delay and speech loss in dynamically managed systems”, J. Gruber and L. Strawczynski, IEEE GLOBECOM '82, Vol 2: F.7.3.1-F.7.3.5
[ETSI, TR 101329 V 1.2.5]	Telecommunications and Internet Protocol harmonization over networks (TIPHON) : General Aspects of Quality Of Service (QoS)
[Technical Report 56, T1A1.7]	Performance guidelines for voiceband services over hybrid IP/PSTN connections.

4

Quality of Service

4.1 Introduction: What is QoS?

Quality of service, or QoS, has been largely ignored in the initial design of IP networks. IP, like other packet network technologies, was built and optimized to transport data files, not voice or video. In this context, the only ‘quality of service’ that was required was that the data should not be corrupted or get lost. Today the improvement of networking technology makes it feasible to transport real-time data, such as voice or video, over an IP network. Therefore, it becomes extremely important to be able to control and characterize the QoS provided by an IP network.

Figure 4.1 summarizes the main parameters characterizing QoS in a packet network:

- The available capacity, also called ‘bandwidth’ in this context (peak, sustained).
- The end-to-end transmission delay (latency) and its variation (jitter).
- Packet loss and desequencing (older packets arriving first).

Bandwidth seems an easy issue to tackle ... just throw more leased line capacity at the problem! In fact, there is more than just providing overall bandwidth: a provider must also ensure that each user of the network gets a fair share of it. This is not a trivial problem, and it is only recently that efficient fair sharing techniques have been deployed.

Latency is by far the most difficult issue of all. A common opinion is to say that IP is simply unsuitable for the transport of latency-controlled data. This is *not* true. Parekh and Gallager found a very useful approach in 1993, leading to a family of queuing algorithms called ‘weighted fair queuing’. These algorithms, although difficult to implement in practice, can guarantee an upper bound on latency for certain flows and enable IP to provide the same guaranteed quality of service as ATM networks.

Jitter is important mainly in real-time applications that need to maintain worst-case buffers to allow for timely delivery of the packets. If there is a lot of jitter, the jitter buffers must be bigger and, therefore, introduce more delays in the end-to-end information path.

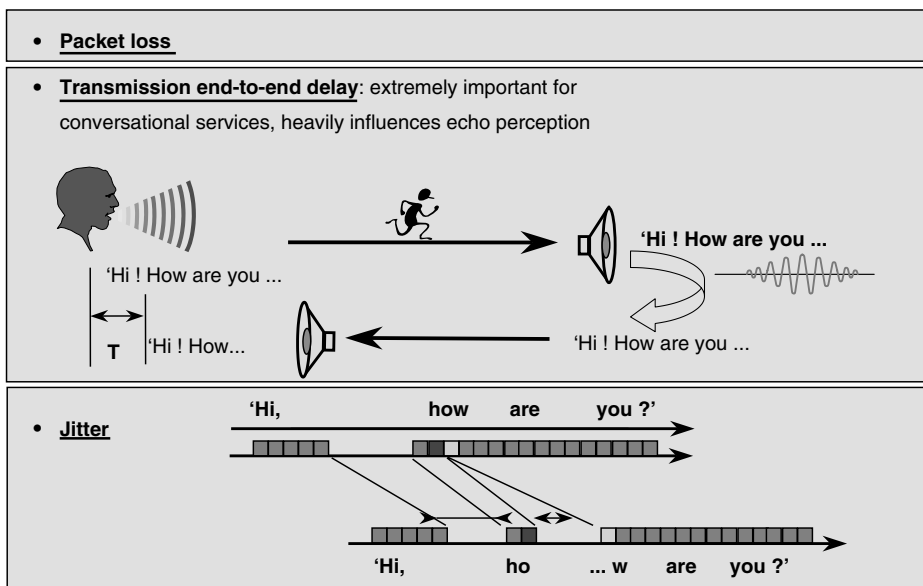


Figure 4.1 Key factors influencing Qos in an IP network.

Packet loss is closely linked to the bandwidth issue and proper use of congestion control at the edge of the network, and within the backbone. Packet loss usually occurs when there is congestion on the packet's path, causing router buffers to overflow. On TCP connections, it will cause a significant drop in the connection's throughput due to the Van Jacobson algorithm. A loss of several packets may switch TCP in slow-start mode and result in a slow connection long after the actual network congestion has stopped. On UDP connections, packet loss will also cause delay (if an acknowledgment and retransmission scheme or a forward error correction method is used) or quality degradation in multimedia applications when loss cannot be recovered due to latency constraints.

Desequencing of packets is mainly influenced by the route stability of the network and efficient queue management in routers with multiple interfaces for a given destination. For most applications desequencing is not a problem *per se* and only causes the same types of delays as jitter.

Telephony is not the only application that poses severe constraints on network QoS: Transaction applications (round trip delays will slow the set-up time of all applications using TCP and may in extreme cases slow the overall transmission speed), interactive applications, such as simulations and games, and some protocol encapsulations, such as SNA, in IP are also very sensitive to network QoS.

4.2 Describing a data stream

A data stream, or session, is a sequence of packets that can be grouped logically (e.g., packets coming from a given computer to another computer). In most applications that

use UDP or TCP packets to communicate between two computers, the packets generated by the application from one computer to the other computer all have the same source IP address and port, and destination IP address and port (software engineers will remark that these elements also characterize a connected ‘socket’). Therefore, these packet properties characterize the group of packets that form the data stream specific to the application session between the two computers.

Protocol, IP address, and port properties can be used to describe the data stream as a group of packets (these properties are often called **filters**), but for QoS engineering it is also important to characterize the timing properties of the sequence of packets: how often a packet is sent, how regularly, etc.

The simplest metaphor that can be used to model a stream is called the fluid model. In this model, packet stream granularity is ignored, just as if it was a continuous stream of bits. A popular fluid model is the **token bucket** regulated stream (or **leaky bucket**) model, as shown in Figure 4.2. The token bucket uses two parameters: the token bucket size σ (in bits) and the incoming traffic long-term average ρ (in bits/s).

The bits of the incoming traffic must remove a token from the bucket before being forwarded to the output. To regenerate tokens, new ones are created every $1/\rho$ second until the number of unused tokens stacked in the bucket reaches a depth of σ tokens. When, this limit is reached the bucket is full and the new tokens are rejected. Therefore, σ represents the size of the largest possible traffic bursts. At any point in time the total volume of traffic that gets through the leaky bucket regulator is smaller than $\sigma + \rho t$, regardless of the profile of the traffic that has been presented at the input. The original properties of incoming traffic have been ‘shaped’ and can now be represented by the σ and ρ parameters.

The token bucket model is widely used to represent the timing properties of a data stream. It captures some of the burstiness characteristics of a stream, as well as the stream’s average rate.

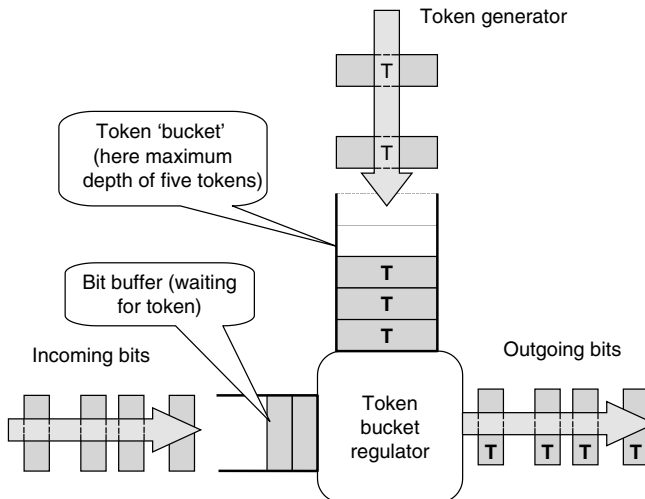


Figure 4.2 The token bucket model.

4.3 Queuing techniques for QoS

What causes so much trouble in a packet network is that, at each node, every packet can be received and processed immediately, but it can be forwarded to the next hop only when some capacity is available over the appropriate link. The delay between the reception and the emission of a packet is therefore variable (this variation is called **jitter**): it can be extremely long if the network is congested or if a very long packet is already being emitted on the interface, or very short if the packet always finds available transmission capacity along its forwarding path.

A simple way of reducing delays and jitter for a given packet stream is to prioritize it over all others, but this is not an acceptable solution, as most networks are designed to serve all users equally well. The notion of ‘fairness’ can have many interpretations. The idea is that all flows should be given the same service or a service proportional to their priority, with all flows with the same priority level treated equally. Depending on the exact interpretation of what ‘fairness’ is, several queue management techniques can be used by the nodes of the network. There has been many efforts to design a **scheduling policy** that would minimize transmission delays while still giving each stream a fair share of the available capacity. The scheduling policy decides, on each outgoing queue corresponding to a transmission link, the order of the transmission of packets or eventually their destruction, and seeks to approach for each traffic flow certain goals in terms of capacity, jitter, latency, and packet loss through each node.

Several technologies exist, which can be grouped in two categories. The first category only takes care of packet ordering in the output queues:

- **FIFO (first in first out)**, also called **first come first served (FCFS)** simply outputs the packets in the order in which they have been received.
- **Class-based queuing** (also called custom queuing by some router vendors).
- **Fair queuing** and **weighted fair queuing** algorithms. Here we will mainly present an algorithm called **PGPS (packet-generalized processor sharing)**, which is the reference fair scheduling algorithm.

These techniques can be combined with any packet loss management technique of the second category:

- Simple overflow.
- **Random early detection (RED)** and **weighted random early detection (WRED)**.

4.3.1 Class-based queuing

Class-based queuing (CBQ) sorts data flows into several logical queues according to filtering parameters (e.g., protocol). Arriving packets are sent to one of the appropriate logical queues, and each separate queue works in FIFO mode. Each queue therefore groups a ‘class’ of data streams.

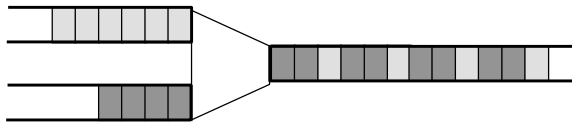


Figure 4.3 Class-based queuing.

Each queue is usually assigned a priority. A scheduler then picks candidate packets from these logical queues according to the priority of the queue and forwards them to the interface physical transmitter. A weighted round robin scheduler is shown in Figure 4.3. In the example, the scheduler picks two packets (if there are more than two packets waiting to be serviced) in the high-priority queue, then services one packet of the low-priority queue, and this goes on until no packet is left to be serviced.

Class-based queuing is extremely useful. A common configuration is to prioritize UDP (DNS, real-time applications) and interactive (TELNET) traffic. The sorting algorithm can rely on the value of the TOS field (see Section 4.4.1) protocol identifier or sort the packet according to a combination of source address, destination address, and port. It can also be controlled by RSVP (see Section 4.4.2), especially to support the controlled load mode of RSVP.

This is simple and efficient, but does not guarantee any delay to any flow. In addition, since the algorithm is not sensitive to the size of packets, the instantaneous capacity allocated to a given class may vary widely according to the size of packets in the queue at that instant.

4.3.2 Simple fair queuing: bitwise round robin fair queuing algorithm

This algorithm is a refinement of CBQ which takes into account the size of packets. The model of fairness that bitwise round robin fair queuing tries to emulate is a TDM (time division multiplex) link. If each class of flow is allocated a virtual time slot on the TDM link, the service will be equal between all classes. If some priority is needed, then one class could be allocated two or more slots. To emulate TDM on an interface, each queue keeps track of the total received byte count. Initially, all queues start with a byte count of zero, then each arriving packet is assigned a tag with the value of the byte count of the queue just after it arrived. Then the scheduler serves packets in the order of their tags. As described above, TDM emulation will allocate the same capacity share to each flow class. But it is simple to allocate different capacity shares (e.g., if queue 1 needs to be configured to use 50% of the available capacity, queue 2 30% and queue 3 20%, then the tags will not be the byte count, but the byte count divided by 5 for queue 1, divided by 3 for queue 2, and divided by 2 for queue 3).

This TDM model is quite good at allocating different shares of the capacity for each queue in an interface, but it has a major flaw: classes not using capacity accumulate the right to use this capacity later. A flow that has not sent data for a while could potentially send a huge burst and be serviced immediately. During this time, even if other flows have to send data, they will be blocked. In other words, the TDM model does not achieve stream isolation.

It is possible to limit this effect by controlling the maximum amount of data that can get in each queue in any given period (e.g., through a leaky bucket regulator).

4.3.3 GPS policy in a node

4.3.3.1 Generalized processor sharing (GPS)

Generalized processor sharing (GPS) is another view of fairness that is better than TDM. For GPS, the ideal multiplexer node should allocate a share of the available capacity to each stream, proportionally to its priority. However, this share must be *immediately* available as soon as there is some data to be sent, even if other flows happen to be in a burst period at the time. If some flows do not need to send data, then their reserved share of the output capacity is redistributed to other streams proportionally to their respective priorities.

Unfortunately, this policy is only possible if we consider that the data of each stream can be arbitrarily fragmented and that many data elements from different streams can be sent at the same time through the output link of a node. This is called fluid approximation. In the real world of packet data, a packet that is being sent takes all of the bandwidth, even if another packet arrives simultaneously and would need its share of the capacity immediately.

If for a moment we accept that packets can be arbitrarily fragmented, then the best possible multiplexer node looks like a tube of toothpaste (Figure 4.4). If the red toothpaste represents one data stream and the white toothpaste represents another data stream, the output is a mix of the red and white toothpaste, where the “red” stream and the “white” stream each get a fair share of the output.

We can estimate the worst case delay that an element of data belonging to a token bucket-regulated stream (average rate ρ , maximum burst size σ) would face going through such a node: it would be σ/R , where R is the capacity allocated to the stream through the node ($R > \rho$).

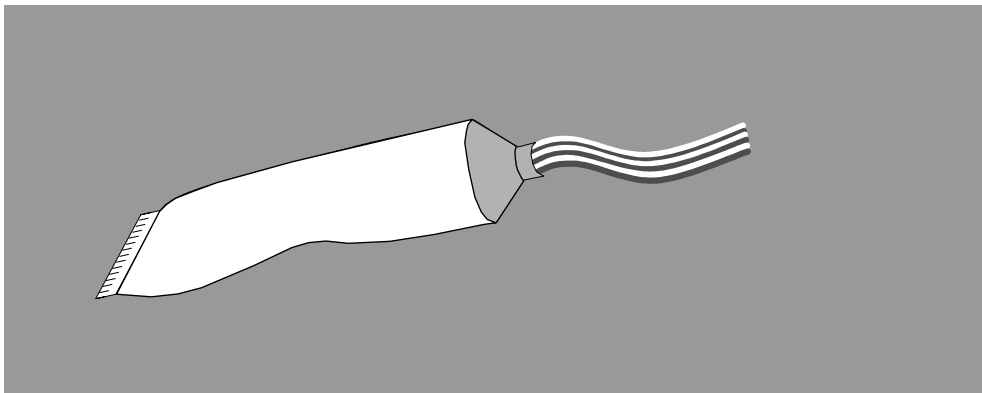


Figure 4.4 An ideal red/white multiplexer.

GPS is a policy where the processing power of the router and the output capacity on each interface are shared among the competing streams. Each stream receives at least a share ϕ_i of the resources ($\sum_{i=1}^N \phi_i = 1$).

A GPS node is non-idling as long as at least one stream is still queued (which means that the scarce resource, output capacity, is never wasted transmitting nothing). The name of this policy was chosen because there is a good analogy to the sharing of CPU cycles between threads in a multitasking operating system.

In Figure 4.5, when packet 1 arrives, it initially takes all the available output capacity. When packet 2 arrives, it shares the capacity with packet 1, which is still being transmitted. Since both streams have the same priority (25%), the output capacity is shared equally between packet 1 and packet 2. When packet 3 arrives, all packets compete for the output, and the GPS multiplexer node allocates half of the capacity to packet 3 (stream 3 has a priority of 50%) and the rest equally between packet 2 and 3. At some point the transmission of packet 3 is complete, and since there are no other packets from stream 3 in the queue the output capacity is shared again between packets 1 and 2. Note that the order in which the transmission of packets completes is not the same as the order of packet arrival, due to the higher priority of stream 3.

More precisely, if $S_i(s, t)$ denotes the volume of a stream that has gone through the node between instant s and t , then:

$$\frac{S_i(s, t)}{S_j(s, t)} \geq \frac{\phi_i}{\phi_j}$$

for each session i continuously backlogged between s and t . The backlogged active sessions share the resources of inactive sessions (not backlogged) proportionally to their ϕ_i .

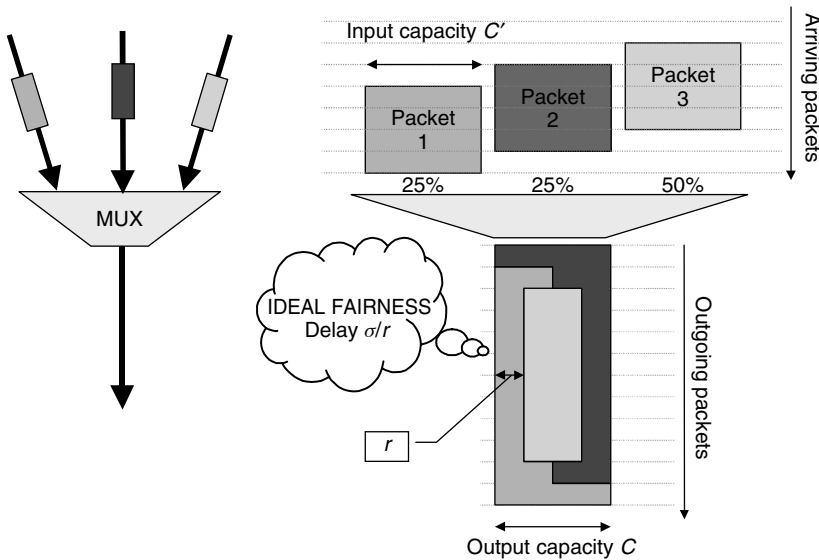


Figure 4.5 Handling three packets using a 'fluid' multiplexer.

In other words, each session i can use at least the capacity $\phi_i * C$ at any time, where C is the total capacity of the output interface considered. If *all* sessions are active, then each session i uses exactly $\phi_i * C$.

The service received by the token bucket-regulated session i (average rate ρ , maximum burst size σ) in a system where N token bucket-regulated sessions share the scarce resource is always better than the service received by session i if all other sessions start with their token bucket full, then send the longest allowed traffic burst, and finally keep sending data at the maximum long-term average rate allowed by the token bucket regulator. With this remark, we can calculate:

- The largest delay through the node $\sigma/\phi_i * C$.
- The buffer size needed for the worst backlog σ .

GPS policy also has a very important property: the relative order of departure (i.e., last bit of the packet has been output) of two packets i and j is independent of future packet arrivals. The reason for this is that if another packet arrives, the transmission speed of i and j is changed homogenously (the factor of change is the same for i and j), which preserves the departure time. The exact time of departure of i and j , though, is obviously changed.

4.3.3.2 PGPS policy in a node

4.3.3.2.1 The PGPS approximation

The fluid model used in GPS is not valid for real-world networks, where packets are received as whole entities (i.e., nothing is processed until the final packet CRC is checked) and put in the output queue as blocks of contiguous bits. Taking into account these real-life facts leads to ‘packet by packet GPS’. The idea behind PGPS is to serve the packets in the order in which they would leave under a GPS policy (see Figure 4.6). This order of departure is not changed by future arrivals, so packets that have already been ordered keep the same order, only newly arrived packets may be inserted in this arrangement.

Figure 4.7 is an illustration (PGPS left, GPS right) of the case when the transmission capacity of each input and the output link is 1. The two leftmost streams are given a weight of 0.25, whereas the rightmost stream is given a weight of 0.5. A packet arrives for each stream every $\frac{1}{4}$ time unit in our case (we take the transmission time of a packet of size 1 to be the time unit): a_i is packet i ’s arrival time. In case you wonder why packet 1 is sent first while it only finishes second under GPS, this is because at the time the PGPS node had to choose a packet for output (remember it is non-idling), packet 3 had not yet arrived and so its departure time was not known.

4.3.3.2.2 Assessment of the PGPS approximation

Under PGPS, packets are sent in the order of departure as known when the decision to select a new packet for the output queue is taken. Because we cannot guess what the future will be, this might lead to errors as in Figures 4.6 and 4.7: a new packet (packet 3, Figure 4.6, time t_3) arrives just after the selection of the new output packet (packet 1,

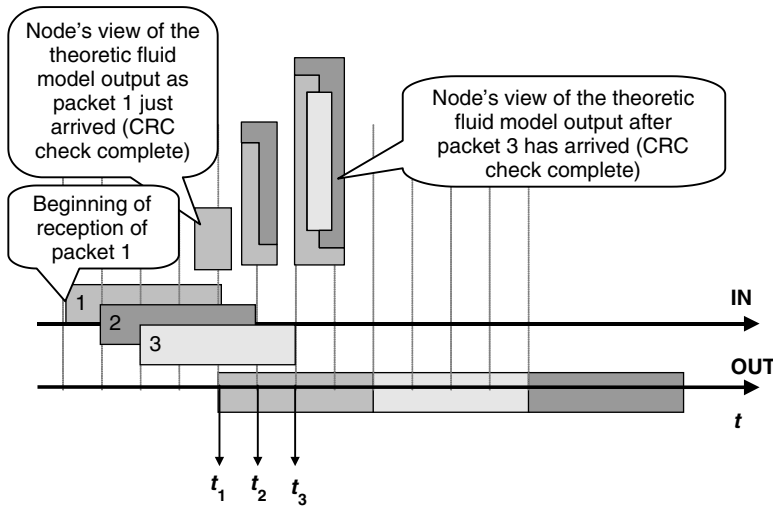


Figure 4.6 Handling three packets using PGPS, according to instantaneous ‘fluid model’ -projected completion order. The router calculates in real time the way packets would be handled under fluid approximation and GPS, in order to find in which order they would finish.

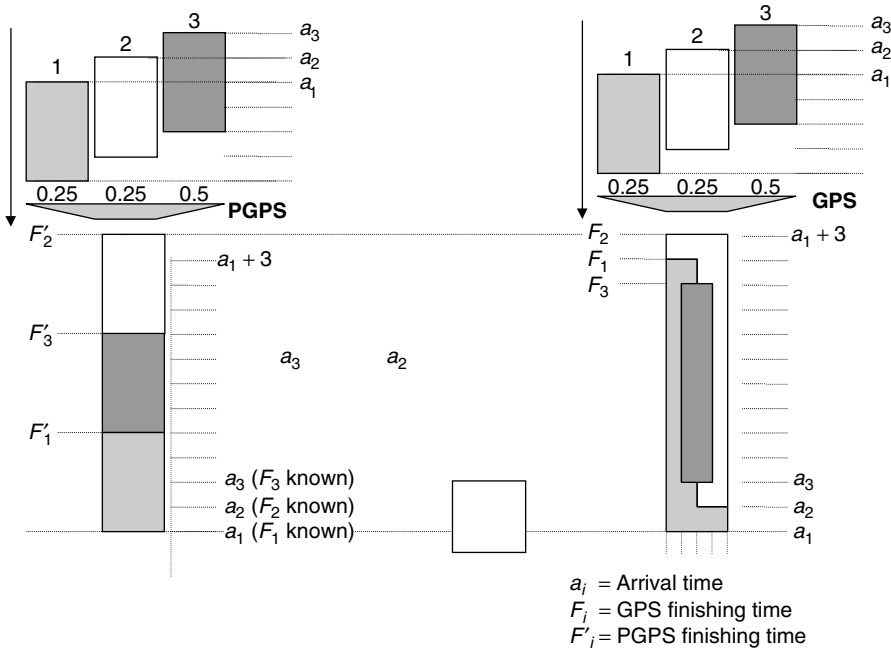


Figure 4.7 PGPS and GPS packet ordering is not always identical.

which arrived in Figure 4.6 at time t_1) and the GPS theoretical departure time F_p of packet 3 is lower than the GPS departure time of packet 1 currently being sent. The PGPS scheduler has no other choice than waiting until this one finishes. Because a PGPS multiplexer cannot guess the future, in some cases the rule ‘PGPS sends packets in the same order as the GPS finishing order’ is not followed.

Because of this, under the PGPS policy, some packets will have their departure time later than under the ideal GPS policy. Imagine, for instance, a high-priority packet arriving in the multiplexer queue directly after a low-priority packet has been sent to the output transmission line. There is an upper bound to this added delay:

$$F'_p - F_p \leq \frac{L_{\max}}{C}$$

where L_{\max} is the maximum size of a packet and C the throughput of the outgoing interface. This limit is approached in the example if we grow stream 3 weight to nearly 1 and we make packets 1, 2, and 3 arrive closer.

The worst delay through the node for stream i under PGPS becomes (L_{\max} is the maximum packet size on inbound links, which must be smaller than the **maximum transmission unit, or MTU**):

$$D_i^* \leq \frac{\sigma_i}{r_i} + \frac{L_{\max}}{C} + Tr$$

where $r_i = C * \phi_i$ is the capacity reserved for stream i , C is the total capacity of the output link, and Tr is the duration of processing in the router. This result is illustrated in Figure 4.8.

The queue buffer necessary to prevent any overflow for stream i becomes a bit larger than in GPS because packets may wait longer:

$$Q_i^* \leq \sigma_i + L_{\max}$$

under the stability condition $\rho_i \leq r_i$.

It is obvious from these formulas that it is better not to have big packets, which was one of the design goals of ATM. However, as output link capacity increases, this becomes decreasingly important. With current transmission technology in excess of 1 Mbps at the edge (xDSL) and multigigabit in core networks, this explains why ATM is quickly getting displaced by IP.

As proof of this, let us number the packets in the order they are processed by the PGPS server since the last busy session. For any packet p_k there are two cases:

- All packets p_j leaving PGPS before p_k ($j < k$) also leave the GPS server before p_k . Therefore, at the time f_k when packet p_k finishes under GPS, GPS has already served all other packets p_j ($j < k$). Because PGPS is work-conserving it will have served the same volume as the GPS server between the beginning of the busy session and f_k . All packets p_j ($j \leq k$) fit in this volume because GPS has served them, so PGPS has also served them before f_k , so $f'_k \leq f_k$.
At least one packet p_m leaving PGPS before p_k ($m < k$) leaves after packet p_k under GPS. Among those packets, let us consider the one leaving PGPS last: p_M . If a_i denotes

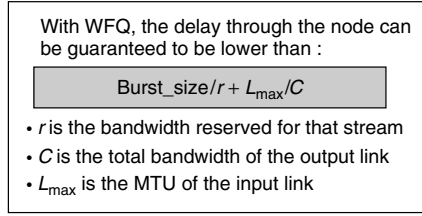


Figure 4.8 Delay guaranteed by a single PGPS node.

the arrival time of packet i , then we have:

$$\forall_i \in [M + 1, k], \quad a_i > f'_{M-1}$$

otherwise, if packet i had arrived before f'_{M-1} (this is when PGPS must choose which packet will go after p_{M-1}), because packet i finishes before packet M under GPS (otherwise M would not be the biggest integer lower than k having $f_M > f_k$) p_i would have been scheduled next, not p_M . So we now have that no packet p_i , i between $M + 1$ and k , had arrived before f_{M-1} . But, they have all been served before f_k (otherwise M would not be the biggest integer lower than k having $f_M > f_k$). So we can write:

$$f_k > f'_{M-1} + \sum_{i=M+1}^k \frac{L_i}{C}$$

where L is the size of the packet and C the bandwidth of the output. Under PGPS we have exactly:

$$f'_k = f'_{M-1} + \sum_{i=M}^k \frac{L_i}{C}$$

which finally gives:

$$f_k > f'_k - \frac{L_M}{C}$$

and because $L_M \leq L_{\max}$ we have our result.

4.3.3.2.3 Computing PGPS packet ordering

A packet has arrived when its last bit has arrived. We call a_i^k the moment of the arrival time for the k th packet arriving from flow I , and the length of the packet is L_i^k . Let s_i^k and f_i^k denote the moment at which packet k begins and finishes to be processed by the GPS server. Then:

(a) $s_i^k = \max\{f_i^{k-1}, a_i^k\}$.

(b) $f_i^k = s_i^k + t(L_i^k)$, where t is the time used by the GPS server to process L bits.

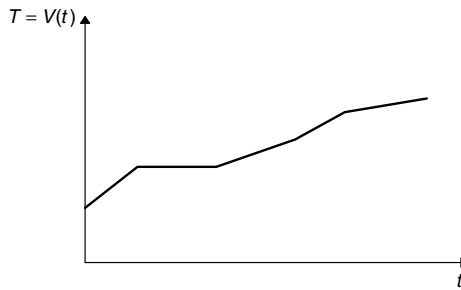


Figure 4.9 The $v(t)$ function.

The difficult part is that the processing speed of the GPS server depends on its load and changes with each packet departure and arrival (called an *event*). If t_i denotes the time of event i , the processing speed for session i is:

$$\frac{\phi_i}{\sum_{j \in B(t)} \phi_j} * r$$

where r is the total throughput of the outgoing link and B is the group of buffered sessions at this time. This reflects the fact that a GPS server redistributes unused reserved bandwidth to active sessions according to their precedence.

We call $v(t)$ the piecewise linear function of t defined by its slope $\frac{r}{\sum_{j \in B(t)} \phi_j}$ (Figure 4.9).

We can rewrite (b) as:

$$(b) \int_{s_i^k}^{f_i^k} \phi_i * v'(t) dt = L_i^k.$$

Writing $S = v(s)$ and $F = v(f)$, (a) and (b) become:

$$(a) S_i^k = \max\{F_i^{k-1}, V(a_i^k)\}$$

$$(b) F_i^k = S_i^k + \frac{L_i^k}{\phi_i} \text{ with } F_i^0 = 0, \forall i.$$

This virtual time T respects the same order relations as t because function $v(t)$ is nondecreasing. In order to build a PGPS multiplexer, for each packet it is enough to calculate F , which is possible as soon as we receive the packet since a and L are known.

With this algorithm, we can immediately classify a new packet among the queued packets according to PGPS scheduling. However, this calculation requires that the PGPS multiplexer maintains the parameters necessary for the calculation of $v(t)$: the coordinates of the last slope change, the current slope, and the number of backlogged sessions. Since each arrival and departure will change these values and packets can arrive simultaneously, this requires significant processing power on high-capacity links.

4.3.3.2.4 PGPS multiplexers in a network

Along a path going through N PGPS multiplexers, the maximal end-to-end delay for a token bucket-regulated data flow is:

$$D_i^* \leq \frac{\sigma_i + (N - 1)L_{\max}}{r_i} + \sum_{n=1}^N \left(\frac{L_{\max}}{C_n} + Tr_n \right)$$

where r_i is the smallest bandwidth amount allocated to stream i along the path. Both the propagation time and the processing time should be included in term Tr . The first term shows that the burstiness of the data flow increases through each PGPS multiplexer, due to possible data accumulation while waiting to be served.

The formula is complex, but it is interesting to note that the first term decreases as the reserved bandwidth increases (see Figure 4.10). The guaranteed delay through a set of PGPS multiplexers can be reduced by increasing the reserved bandwidth beyond the average bitrate of the data stream. This is the key result used by RSVP in guaranteed service mode. Now if the stream goes through several WFQ nodes, the end-to-end delay will be lower than shown in Figure 4.10.

There is also a more accurate version of the equation in Figure 4.10. Let us now consider the peak emission rate p_i of stream i :

$$D_i^* \leq \frac{\sum_i C_{\text{tot}}}{r_i} + D_{\text{tot}}$$

where:

$$C_{\text{tot}} = \sum_{n=1}^N L_{\max}^n$$

$$D_{\text{tot}} = \sum_{n=1}^N \left(\frac{L_{\max}^n}{C_n} + Tr_n \right)$$

$$\sum_i = L + \frac{(\sigma_i - L)(p_i - r_i)}{(p_i - \rho_i)}$$

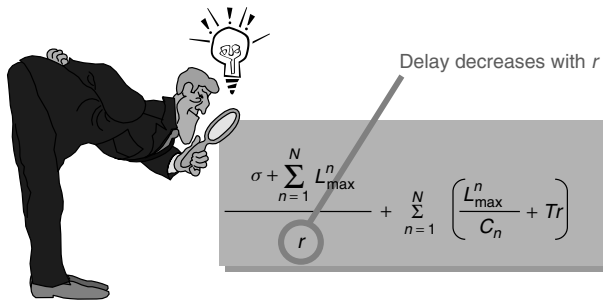


Figure 4.10 End-to-end delay through multiple PGPS nodes.

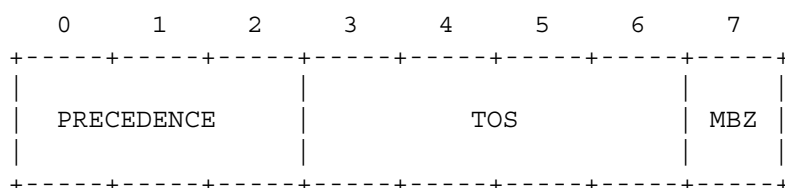
RSVP gives a data flow receiver all the parameters in this formula. The receiver then selects the rate r that he wants to reserve in order to have an acceptable, guaranteed end-to-end delay.

4.4 Signaling QoS requirements

4.4.1 The IP TOS octet

The IPv4 packet **type of service (TOS)** octet, shown in Figure 4.11, captures the parameters describing how this packet should be handled relative to quality of service. IPv6 has a similar octet called the **traffic-class** octet.

The IPv4 TOS octet was traditionally structured as follows:



The IP precedence field uses the first 3 bits, encoding a value between 0 and 7. Packets with a higher IP precedence value should have a higher priority in the network. The traditional meaning of the IP precedence values (RFC 791) values is described in Table 4.1. The vocabulary used reflects the military origin of IP, a ‘flash’ IP packet was supposed to be the electronic equivalent of a flash message.

The following 4 bits form the TOS field and were supposed to be used to mark a desired trade-off between cost, delay, throughput, and reliability, as described in RFC

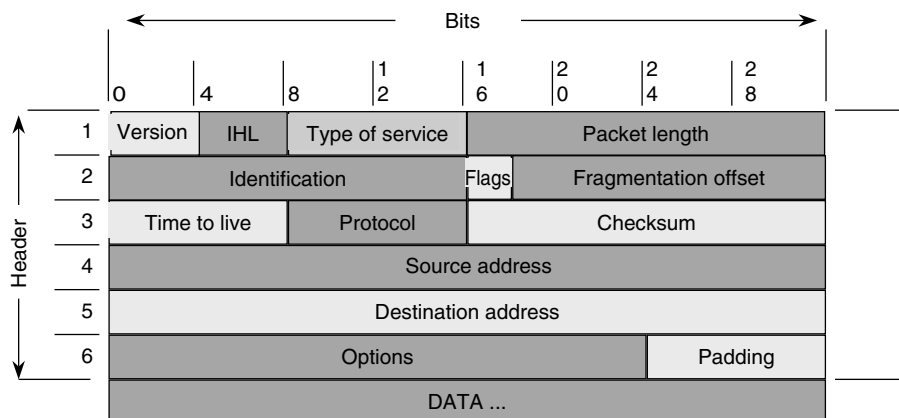


Figure 4.11 The IPv4 header.

Table 4.1 Precedence values in the original RFC 791

Value	Definition	
Network	Match packets with network control precedence	(7)
Internet	Match packets with inter-network control precedence	(6)
Critical	Match packets with critical precedence	(5)
Flash override	Match packets with flash override precedence	(4)
Flash	Match packets with flash precedence	(3)
Immediate	Match packets with immediate precedence	(2)
Priority	Match packets with priority precedence	(1)
Routine	Match packets with routine precedence	(0)

Note: Levels 110 (6) and 111 (7) are reserved for routing and some ICMP messages (see RFC 1812 for details). Therefore, only six levels (routine–critical) remain for user applications.

1349. Originally, RFC 791 used only the first 3 bits as the TOS field, and the last 2 bits were part of the MBZ field. RFC 1349 defined some extended values for 4 bits of the TOS field:

- 1000—minimize delay.
- 0100—maximize throughput.
- 0010—maximize reliability.
- 0001—minimize monetary cost.
- 0000—normal service.

The idea is that interactive services like TELNET should require TOS 1000. In the original RFC 971 it was legal to add those values to the required combined properties. RFC 1349 no longer allows this, and value 1100, for instance, could mean anything. RFC 1349 requires the last bit (MBZ) to be 0: the MBZ (‘must be zero’) field is for experimental use and is ignored by routers. RFC 1122 and 1123 (Host requirements) also defined a few rules for setting TOS values for hosts, but they were based on a 5- bit TOS field.

If you are beginning to think that things are not crystal clear, you are right. Things are not clear indeed. When sending traffic over the Internet with a specific IP precedence and TOS value, no one can be really sure of the behavior of routers along the path, unless only one provider was in control of the whole domain and had properly configured the forwarding policies of all routers.

4.4.1.1 Using the IP precedence field

The most straightforward use of the IP precedence field is to cause interface schedulers to prioritize marked packets. Beyond this, the IP precedence field can be used at the network access level in conjunction with policy routing and priority routing. Packets primarily need to be marked with the proper TOS field. If the sender of the packet does not do it directly, the TOS field can be set by a router: many routers can overwrite the TOS field based on certain filtering criteria.

For example, a router can be configured to set the IP precedence field of TCP traffic on port 80 to critical (this example is for a Cisco router):

```

interface Serial0                                /* interface facing the customer router
ip address 10.0.0.1 255.0.0.0
ip policy route-map test                        /* this activates policy routing on interface
                                                Serial 0
interface Serial1                                /* interface to low-latency backbone
ip address 192.168.1.1 255.255.255.0
interface Serial2                                /* interface to normal latency backbone
ip address 192.168.2.1 255.255.255.0
access-list 101 permit udp any any gt          /* this simple filter will match
1023
                                                RTP traffic (but not only)
route-map test permit 2                        /* defines the default path
set default int serial2
route-map test permit 1                        /* Defines route map 'test' 1
match ip address 101                          /* all ip addresses that pass filter 101
set ip next-hop 192.168.1.5                    /* will go through interface serial1
set ip precedence critical                     /* set TOS field to critical for all traffic
                                                matching access list 101

```

Once this is configured in the customer's access router, the network provider has two options:

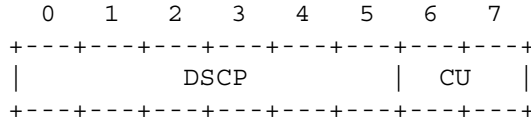
- Prioritize the IP traffic in the backbone according to the value of the TOS field. This can be done using priority queuing, class-based queuing or weighted fair queuing.
- Use a separate path in the network (e.g., avoiding satellite links) for IP traffic with critical priority. This ability to bypass the regular routing mechanism to set custom next hops or custom IP precedence to packets is called policy routing. On Cisco routers these features are also enabled by the 'route map' set of commands. In the example the low-latency network can be reached through interface serial1.

4.4.1.2 (Re)defining the values of the IP TOS octet

There has been much theoretical work on the behavior of packet networks since the creation of the IP, and people have much more experience on the sophisticated queuing mechanisms used in QoS-enabled equipment. The work done for ATM and frame relay networks has also helped us to get clearer ideas on the QoS-related information that needs to be transported in each packet.

It was high time to review the original meaning(s) of the IP TOS octet and stop wasting 8 bits per IP packet. This revision work became the charter of the IETF DiffServ group, with the intent to 'provide scalable service discrimination in the Internet without the need for per-flow state and signaling at every hop.' The Diffserv group redefined the semantics

of the IPv4 TOS octet and the IPv6 traffic-class octet in RFC 2474 (December 1998) which made both RFC 1455 and 1349 obsolete. The name TOS itself was changed; this byte is now called the **DS**, or **differentiated services**, byte. The DS byte is subdivided into a 6-bit **DSCP (differentiated services codepoint)** field and a CU (currently unused) field:



The codepoint value should be used as an index to the appropriate packet handler, or **per-hop behavior (PHB)**. A PHB applies a particular forwarding treatment to all packets with a particular DSCP field and direction. This class of packets is called a **behavior aggregate** (Figure 4.12). For instance, each behavior aggregate can be mapped to a particular queue in a weighted round robin CBQ or WFQ scheduler.

The index is based on an exact match on the 6 bits of the DSCP (the 2 CU bits being ignored). Each specified PHB should be assigned a unique default DSCP field among the 64 that could potentially be available with 6 bits. In fact, RFC 2474 has allocated three pools of codepoints:

- xxxxx0 for 'standard actions'. Eight codepoints (yyy000), called **class selector codepoints**, are already allocated for backward compatibility with the IP precedence field of RFC 791. RFC 2474 states that the set of PHBs mapped to these codepoints must offer at least two independent queues, expedite forwarding according to yyy values (the

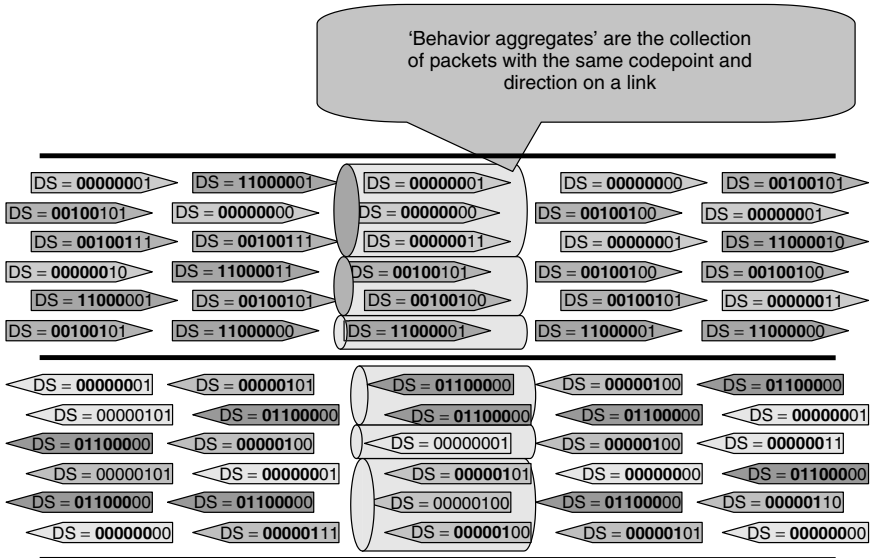


Figure 4.12 Behavior aggregates.

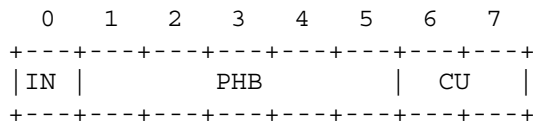
higher the *yyy*, the lower the average queuing delay), and prioritize *yyy* = 110 and 111 (routing traffic) over *yyy* = 000 (best effort traffic). Note that this set of requirements does not imply the use of one particular scheduling algorithm (WFQ, CBQ, or priority queuing could be used). This philosophy will be retained when defining other PHBs.

- xxxx11 for experimental or local use.
- xxxx01 for experimental or local use, or extension of the standard actions pool if it gets fully allocated.

Even if each PHB is allocated a default codepoint, Diffserv nodes are free to assign other codepoints to a particular PHB (except for xxx000 codepoints). In fact, DSCP fields may have a meaning that is only local to a domain. At the boundary of such DiffServ domains, it is very important to control the forwarding of IP packets according to their DSCP fields and, if necessary, to map some codepoints to other values. The configuration of PHB-to-DSCP field mapping is an administrative decision that may vary from domain to domain. Two service providers with service-level agreements must either agree on DSCP field values for each PHB or properly configure DSCP field translation at boundary nodes.

All packets with unknown codepoints (not part of the service-level agreement of a service provider) are assumed to be part of the best effort forwarding behavior. This PHB must always be present in a DS-compliant node. The default codepoint for the best effort PHB is 000000, and this value must be recognized by all DS-compliant nodes. So far only class selector PHBs have been defined (in fact, this was mostly a mapping of the IP precedence semantics of RFC 791), but in the future there could be a 'strict priority queuing PHB' with a different set of requirements.

In order to avoid mixing widely different traffic types into a single queue, providers of DiffServ networks are expected to perform some traffic shaping/regulation at the edge of the network (e.g., with a token bucket). Occasionally, there will be flows exceeding regulator settings. It is not always a good idea to immediately mark such out-of-profile packets for best effort forwarding, since this practice can introduce unnecessary desequencing. Maybe the network still has enough capacity to carry the excess packet in sequence; therefore, the best solution is to mark this packet as being 'out of profile'. A node will forward this packet as if it was 'in profile' if there is enough capacity locally or discard it otherwise (or mark it for best effort forwarding). The solution currently recommended in RFC 2475 for marking in-profile and out-of-profile packets is to use two codepoints. In a previous proposal, the first bit was allocated to mark an out-of-profile packet:



where 'IN' represents in (1) or out (0) of profile. The CU field has yet to be allocated, but could be used for forward/backward congestion notification. This has been very useful in frame relay networks.

A more complete description of the differentiated services architecture has been published in RFC 2475 (December 1998). This RFC mainly introduces a specific DiffServ

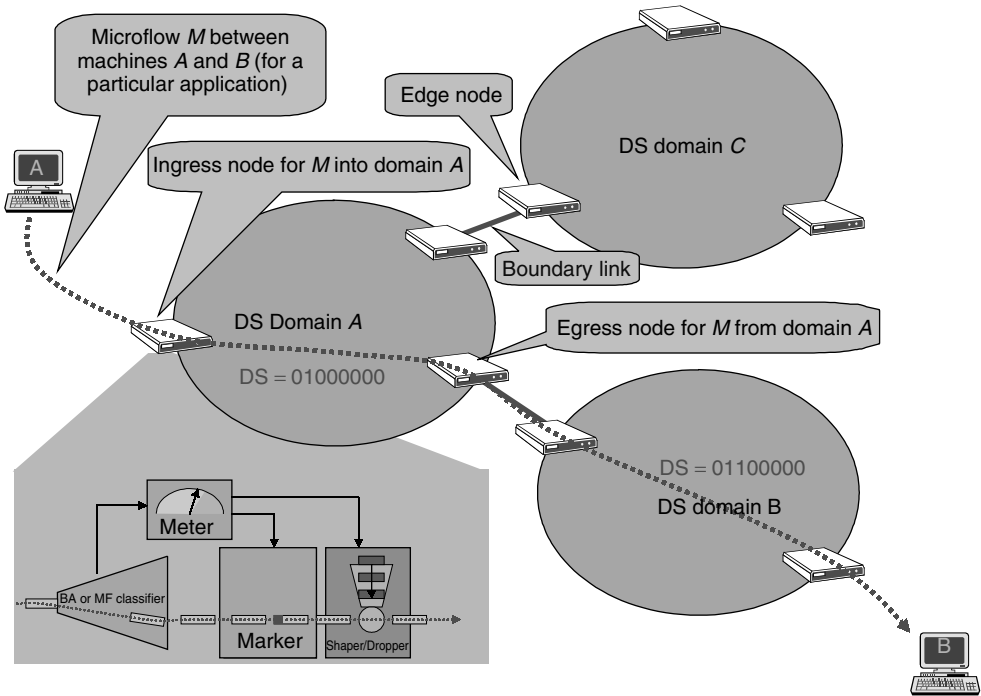


Figure 4.13 The DiffServ ecosystem. The ingress node implements the traffic-conditioning agreement (TCA) used by the provider's service level agreement (SLA).

vocabulary (some of which appears in Figure 4.13), discusses the DiffServ paradigm, and compares it with other architectures supporting service-level agreements.

4.4.1.3 Remaining issues with IP TOS/DS octet

4.4.1.3.1 Beware of layer 2!

It is important to remember that IP is used as a layer 3 protocol only and, therefore, any layer 2 multiplexer can potentially ruin the IP-level quality of service. Here are a few common examples:

- *Ethernet*: most PCs and IP phones are connected to Ethernet LANs. While shared coax cable LANs have almost disappeared, most corporations still have Ethernet LANs based on Ethernet hubs. Since hubs typically broadcast any received frame to all connected Ethernet segments, such networks frequently have a high percentage of packets lost due to Ethernet-level collisions. VoIP should never be deployed on such networks. At a minimum, VoIP should be deployed on switched LANs (switches maintain a cache of MAC Ethernet addresses connected on each link and therefore forward Ethernet

frames only to the proper link, reducing unnecessary broadcast traffic). In many cases, however, this is still insufficient if Ethernet concentration links are used where Ethernet frames can get discarded with a probability of over 1% (this can happen even on moderately loaded networks, due to the bursty nature of data communications). A possible enhancement is to use two separate LANs: one for bursty data traffic, one for IP telephony (switched LAN). On recent IP phones and Ethernet switches, the IEEE 802.1Q **VLAN** standard allows network architects to emulate two distinct Ethernet LANs on a single physical network (a 12-bit VLAN tag is appended to the MAC address, as well as 3 bits whose usage is defined in 802.1P). Many IP phones and switches also support the 802.1P QoS standard, which offers eight levels of QoS on an Ethernet LAN: these levels are offered by managing the 3 QoS bits and, in case of collision or congestion, lower precedence frames get discarded. This strict priority-based QoS is sufficient on LANs, because latency is always very low. IP precedence DSCP fields map easily to these eight-layer two-precedence levels.

- *ADSL*: when service providers began deploying ATM DSLAMs (ADSL access concentrators) around 1995, ATM was still regarded by many as the technology of the future for multi-service applications. The Internet and IP technology were still considered a toy. As a consequence, most service providers did not anticipate that *all* future multimedia applications would run exclusively on IP and planned their support for QoS only for native ATM applications. ATM has indeed an extensive support for QoS, based on a connection-per-connection negotiation that is part of the connection set-up process. Unfortunately, service providers quickly discovered that dynamic connection establishment did not scale to large networks, and most ADSL offers are built on ADSL CPEs connected to the core ATM network through a single ATM permanent virtual connection. All IP traffic is therefore routed over that single channel and all IP packets, high priority or not, are sent as fragmented cells. When congestion occurs (e.g., on the ATM concentration links from the DSLAM to the backbone), cells are dropped at random. This has a very negative impact on IP traffic: it is very inefficient, as a single lost cell will prevent the destination router from correctly reconstructing the IP packet, but all the other now-useless cells will still be conveyed by the ATM network. It also ignores IP precedence completely, and high-priority IP traffic is impacted with the same probability as the low-priority IP traffic. This problem has become the hottest issue of many ADSL service providers. All newer DSLAMs are now IP-aware and can properly discard whole packets and prioritize them according to precedence fields. For networks based on older DSLAMs, there can only be work-arounds (e.g., oversizing the ATM network to prevent cell loss) which create two ATM connections per ADSL CPE (one for high-precedence traffic, one for best effort) or use of the CPL ATM cell bit (cell loss priority).
- *Frame relay*: some international backbones still rely on frame relay networks for IP transport. Many of these frame relay networks are still relatively old and were optimized for data only. The only important quality-of-service parameter for such data networks was frame loss; in order to reduce it to the minimum, large buffers have been configured in each frame relay node, thus preventing overflows due to data bursts. Unfortunately, this also creates large, uncontrollable delays (and jitter) and IP packets transported on such layer 2 networks cannot transport voice.

- *Wireless links*: the 802.11 family of wireless standards is very popular. However, at present most vendors do not implement any quality-of-service mechanism that would enable VoIP packets to have precedence over best effort data. The 802.11e prioritization standard is required for any serious implementation of VoIP and data over 802.11 links. Until then, wireless networks can only be used by VoIP if they are dedicated to voice only.

4.4.1.3.2 *Number of traffic classes*

Sorting flows based on the old IP precedence value limits the number of queuing behaviors to eight, of which six are available to end-user applications. This can be further refined by using packet filters based on the protocol number (e.g., to prioritize UDP over TCP) or destination/source addresses and ports.

Offering six classes of service to the end-user may seem enough, when thinking only in terms of broad ‘classes’ that should be prioritized, because it is hard to think of more than six very distinct and useful behaviors. However, this is valid only if all sorts of traffic classes that require a specific forwarding behavior can be grouped in the same queues. Unfortunately, very bursty traffic and smooth traffic should not be mixed in the same queues, as this might degrade the properties of smooth traffic (e.g., voice). This requires a traffic-class value for each combination of desired per-hop behavior and category of ‘burstiness’. In addition, as we have seen above, it is useful to mark out-of-profile traffic at the edge of the network, which really requires two identifiers for each traffic class.

The more recent DiffServ framework makes things potentially much better, since up to 32 packet-handling algorithms could be indexed (with the possibility of marking out-of-profile packets, which uses two codepoints for each traffic class). For current applications, this new framework seems to provide enough traffic classes.

4.4.1.3.3 *Identifying data flows that should be mapped to traffic classes*

In an ideal world, all applications would ‘know’ the DSCP codepoint to use when sending IP packets, and no one would try to cheat by using inappropriate codepoints. Unfortunately, in real networks it is frequently necessary to either set or verify the DSCP fields before injecting the packet into the network. Packets that should use a given codepoint can be recognized using filters based on the packet protocol, IP address, etc. However, some application sessions are impossible to prioritize using static filters (e.g., all applications that use a dynamic port negotiation, such as SIP or H.323). If the router has no proxy capability for the application, it has no way of knowing which port to prioritize. The only possibility is to prioritize an entire range of ports or all packets originated from the host. Obviously, in many cases this is not enough. In a shared commercial backbone, this also creates potential security issues: since the prioritization mechanism uses static filters, a devious user can decide to design an application that ‘looks like’ an authorized application but uses many more resources (e.g., if the provider prioritizes UDP in order to speed up small DNS queries, videoconferencing users will also benefit from it, while they obviously use many more resources).

As we will see, RSVP provides a much more powerful solution to negotiate certain QoS levels for a given data stream dynamically. RSVP can also be extended to include

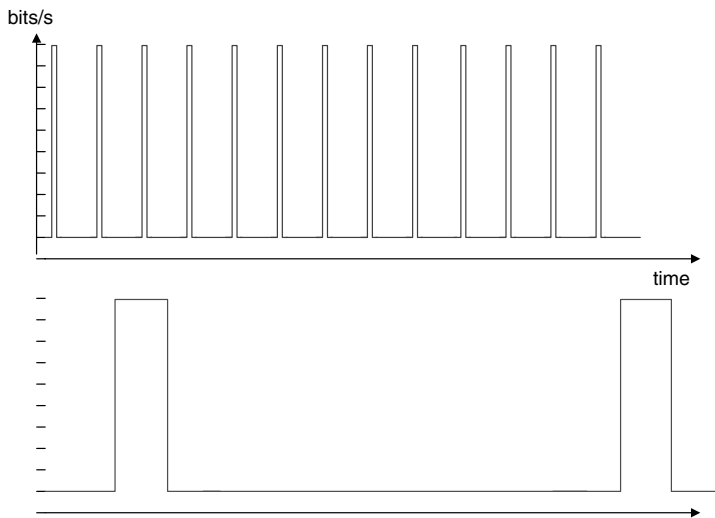


Figure 4.14 Sessions with same average rate and peak rate, but different burst size.

authentication mechanisms and, therefore, can secure access to the backbone resources for prioritized traffic.

4.4.1.3.4 *Network dimensioning and pricing*

For network dimensioning, it is very useful to know the characteristics of the data streams that are being multiplexed. For instance, let us compare the multiplexing of the sessions illustrated in Figure 4.14:

- Flows having an average rate of 20 kbit/s, a maximum burst of 1 kbit at a constant rate of 100 kbit/s, and a minimum constant bitrate of 10 kbit/s.
- Flows having an average rate of 20 kbit/s, a maximum burst of 100 kbit at a constant rate of 100 kbit/s, and a minimum constant bitrate of 10 kbit/s.

In both cases it is easy to calculate that the high bitrate occurs one-tenth of the time and the low bitrate occurs nine-tenths of the time. But, for the first type of flow the bursts are very short (1/100 s), while in the second case the bursts last 1 s. So, in the first case the provider will be able to fit 1,000 flows in just a little more than $1,000 \times 20$ kbit/s (20 Mbit/s): the excess traffic during bursts will accumulate in small router buffers, and the resulting delay will not be too large. But, in the second case the bursts last much longer: the required buffers would be too large for a 20-Mbit/s link and the resulting delays unacceptable. So the provider needs to provide significantly more than $1,000 \times 20$ kbit/s in order to keep the buffer size low in routers.

In general, bursty traffic is much more expensive to carry than smooth traffic. In the next generation IP networks providing QoS, the provider will probably try to isolate these flows and apply a special pricing (such a pricing could be hidden in a ‘right to use’ the application generating such data streams). In order to identify such flows, it would be useful to have a description of the characteristics of each data stream that a customer

sends in the backbone. Based on this description, the provider can decide which streams can be grouped and how expensive it is to carry them. The TOS octet value alone does not describe the traffic characteristics, and there is no easy way for the provider to sort similar streams together and have higher tariffs for more bursty streams.

We will see that RSVP provides the network with many parameters that characterize the properties of the data stream, which helps each router to decide in which queue the traffic should be sent and eventually to choose a tariff.

4.4.1.3.5 *Programming applications that require QoS*

The internal backbone of a provider will have well-defined TOS values/DS codepoints for each class of service. But different providers will probably use different values for the same class. Even with the current DiffServ RFCs only the relative behaviors of class selector PHBs are defined, but the quality of a particular codepoint could vary widely when changing providers. When designing a DiffServ-aware application, the programmer cannot know in advance which codepoint value must be used and, therefore, the application will probably need some configuration. The average user will have no means of deciding which codepoint is appropriate for each application: What are the implications for delay? Can this application recover from packet loss? Is it sensitive to jitter?

It is much easier for the programmer to be able to ask the network what it needs in terms of bandwidth and delay, and let the network provide the required QoS. This requires some signaling mechanism between the applications and the network, and this is where RSVP can play a key role.

4.4.2 RSVP

4.4.2.1 *RSVP is an enabler of a business-grade Internet*

Many people used to consider IP networks would never need sophisticated QoS mechanisms, because the ever-increasing capacity of backbones would always push back the time when QoS would really be useful. ‘Peer-to-peer’ applications have demonstrated that this assumption was wrong. From the point of view of service providers these applications can be considered as a new generation of viruses which do not attack PCs, but attack best effort IP networks instead: file exchange software automatically uploads files without a need for human intervention, ensuring that whatever capacity is available in the backbone will always be saturated, and peer-to-peer sessions cannot be easily identified (they are even designed to escape most classification attempts). This type of traffic is now jamming most IP networks, and the situation is likely to get much worse when users discover that they can download not only MP3 music, but also high-quality MPEG4 movies.

This situation will soon require legitimate applications to be able to use a level of service beyond ‘best effort’. We saw in Section 4.4.1.3 which difficulties are encountered when we only use IP precedence as a way to signal a need for QoS. Commercial providers need to be able to:

- Promote native support of QoS by IP applications.
- Arrange agreements to support QoS across networks managed by different entities.

- Provide QoS guarantees when needed.
- Bill for the service.

Therefore, they need to:

- Give applications a uniform way to ask for a level of QoS.
- Guarantee a level of QoS to the requesting application.
- Provide authentication.

RSVP is an appropriate answer to all these issues.

4.4.2.2 Services provided by RSVP

RSVP is the key component of the IETF **integrated services** model (**IntServ**) and offers two types of services:

- *The controlled load service*: an application requesting the controlled load service for a stream of given characteristics expects the network to behave as if it was lightly loaded for that stream. The exact meaning of this is not completely defined in RSVP, but the general understanding is that packet loss should be very low or null. The absolute delay is not specified, but jitter should be kept as low as possible since in a lightly loaded network router buffers are empty. This is typically the service that could be requested to distinguish normal web browsing or email applications from peer-to-peer traffic.
- *The guaranteed service*: the guaranteed service not only requests bandwidth, but also a maximum transit delay. RSVP's guaranteed service is built on the PGPS paradigm (see Section 4.3.3.2). In the PGPS formula of the maximum theoretical delay, parameters C and D appear as sums along the path of the stream through the network: RSVP is used to calculate these sums and propagate the intermediary results between RSVP routers. The aim is to make C and D available to the recipient of the stream, together with the traffic characteristics of the flow, such as maximum burst size σ , average rate ρ , and peak rate p . This information allows the recipient to calculate the bandwidth r he wants to reserve for that stream in order to achieve a particular delay limit: in the formula given in Section 4.3.3.2.4 the maximal delay D_i^* is a decreasing function of r_i , so by allocating a greater minimal rate r_i the recipient can make the transit delay through the network as close as possible to D_{tot} , which is the smallest value he can hope for. In our description of PGPS, we emphasized that packets would not arrive later than the calculated PGPS delay limit, but they could also arrive *much* sooner. This means that RSVP cannot be used to specify maximum jitter independently of maximum delay. The jitter guaranteed by RSVP is nothing more than the difference between minimum path latency (propagation delays) and maximum guaranteed delay. The only way to request very low jitter is to request a delay that is very close to minimum path latency: we will see later that this is not very practical since the bandwidth reservation needed to request such a delay is extremely large. Not having strict control over jitter is in fact not very important for most applications: interactive applications need very low round

trip delays and can adapt to jitter using jitter buffers and protocols, such as RTP. But this could be a problem for applications using very large bitrate streams, because they would need to allocate a lot of memory for jitter buffers.

There are no clear guidelines in IETF documents for the use of guaranteed service versus controlled load service when writing RSVP-aware applications. The main difference is that controlled load parameters do not include target end-to-end delay values. Since the guaranteed service is more complex to implement, it may not be as readily available as controlled load mode.

4.4.2.3 RSVP messages

RSVP mainly uses two types of messages:

- PATH is sent by the source of the stream. This message initially contains data describing the stream (TSPEC); in particular, the bucket parameters σ and ρ . It follows exactly the same path as the stream itself¹ (including multicast transmission, see Chapter 6), and each router updates the data elements C_{tot} and D_{tot} that are also part of that message (ADSPEC). Figure 4.15 describes the propagation of PATH messages for a multicast stream. At each hop, an RSVP router modifies the PATH message to update

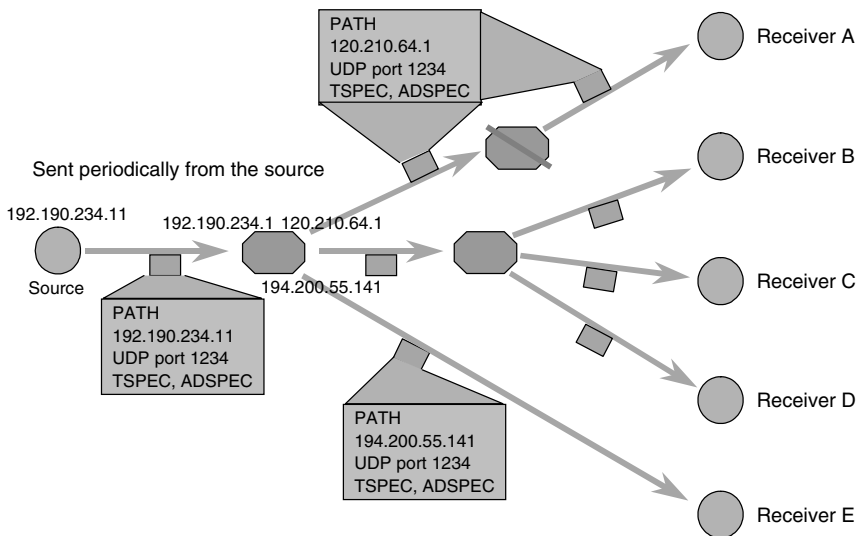


Figure 4.15 PATH message propagation (multicast example).

¹ RSVP is only useful if the data packet part of the session follows the same path as PATH messages: in order to enforce this for some complex routing algorithms, it is necessary to have a dialog between the RSVP process and the routing process; this is the role of the RSRR (routing support for resource reservation) interface.

the C and D parameters, and includes its IP address in the message before forwarding it. It also stores the last hop address that was originally in the PATH message. As illustrated on Figure 4.15, if one of the routers is not capable of handling RSVP, it simply forwards the PATH packet as it received it.

- RESV is sent by the recipient(s) of the stream toward the source following exactly the inverse path of the upstream packets and PATH messages. Each RSVP router, when receiving a RESV message for a flow, forwards it to the last hop address that was previously stored from the PATH message. The RESV message specifies the minimal bandwidth r_i required for stream i , calculated from the data contained in the PATH message using PGPS theory, in order to obtain a desired maximum delay. Eventually, it can also specify an error margin on that target delay, if it needs bandwidth but not low delays (e.g., a video-streaming application). The session for which the capacity is reserved is characterized by a filter; so, a single reservation can apply to several streams (e.g., several sources in a conference). This is called a shared reservation.

Figure 4.16 shows how RSVP works even through non-RSVP routers: receiver A used the last hop address which it found in the PATH message as the destination address for the RESV message. This is in fact the address of the last RSVP router along the path: for RSVP, non-RSVP clouds appear between A and B as a direct link between A and B. If there is no congestion or significant delay in the non-RSVP cloud, the end-to-end reservations made by RSVP will still be valid.

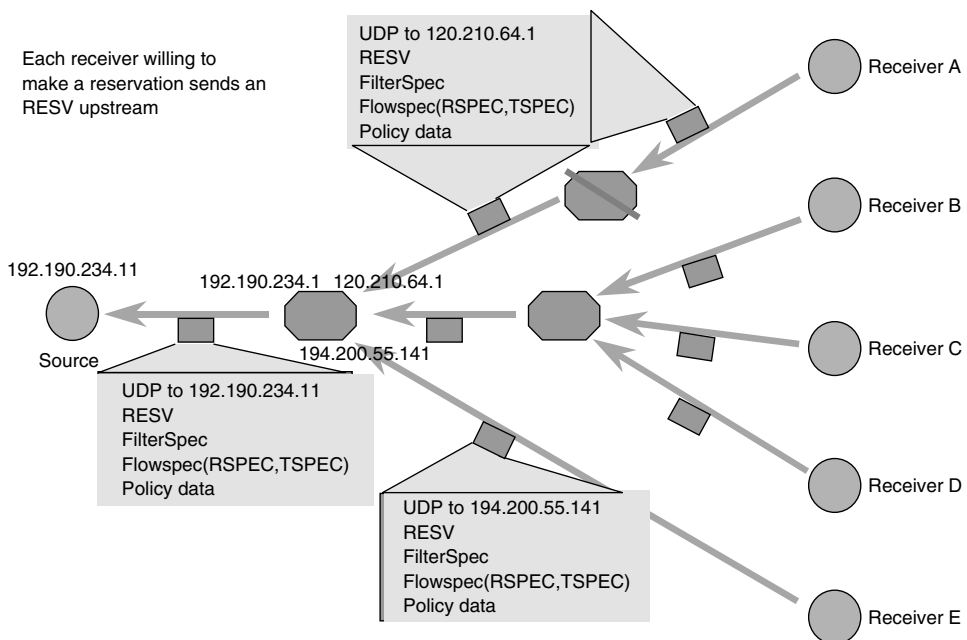


Figure 4.16 RSVP message propagation (multicast case).

Figure 4.16 also shows how multiple reservations for a multicast stream can be merged. If receiver B requires a low delay (large reserved rate) and receiver C is prepared to cope with more delay for the same multicast stream, then only the largest reservation will be forwarded upstream. In the case of multicast streams, reservation requests are initiated by *any* receiver and merged in the network. This is a very powerful feature of RSVP, which so far has no equivalent on ATM networks (although it could be included in ATM UNI 4.0).

During the reservation set-up phase, it is very important to avoid losing either the first PATH message or the RESV message, otherwise the reservation could be delayed by up to 30 s. For instance, over a DiffServ-enabled backbone, PATH and RESV messages could be transmitted over the netctrl (precedence level 7) class of service.

4.4.2.4 Using RSVP to set up a controlled load reservation

The RSVP controlled load service is very simple and can be implemented over custom-queuing routers. If a receiver requests a reservation of 200 kbits/s for a stream with bursts up to 10 kbits, each router can configure its scheduler to allocate an average of 200 kbits/s to the stream: for instance, if the outgoing link is an E1 line (2 Mbits/s), then the scheduler must service the queue allocated to the stream at least 10% of the time.

This is not enough to guarantee a low packet loss if the traffic is bursty: each RSVP router must also make sure that the queue buffer is large enough to accommodate bursts. For instance, in our example, if the scheduler services the stream for 1 ms in every 10 ms, then the worst case is if the burst occurs just after the scheduler has finished to service the stream: the stream traffic will accumulate for 9 ms (i.e., 10 kbits for the burst and $0.009 * 200$ kbits for the regular flow after the burst). In this case the queue buffer needs to be large enough to accommodate 11.8 kbits of data. The calculation can be refined if we also know the peak rate of the traffic, in which case the initial burst will not be considered instantaneous.

This step is repeated for every RSVP-enabled router on the path. Each router can change the characteristics of the stream, and in general the traffic will become increasingly bursty; so, routers downstream will have to allocate even larger buffers and may choose to reshape the stream. Some routers have low-capacity CPUs and may also become congested because of a lack of CPU power (this is especially true for flows generating small packets, such as IP telephony). The reservation algorithm should also make sure that enough CPU cycles will be saved for processing of the flow.

4.4.2.5 Using RSVP to set up a guaranteed service reservation

4.4.2.5.1 Example

In Figure 4.17, source A sends a stream to B and declares the following stream characteristics in the sender TSPEC and ADSPEC parts of the PATH message:

- *TSPEC*: ($p = 10$ Mbits/s, $L = 2$ kbits, $\rho = 1,024$ kbit/s, $\sigma = 32$ kbits).

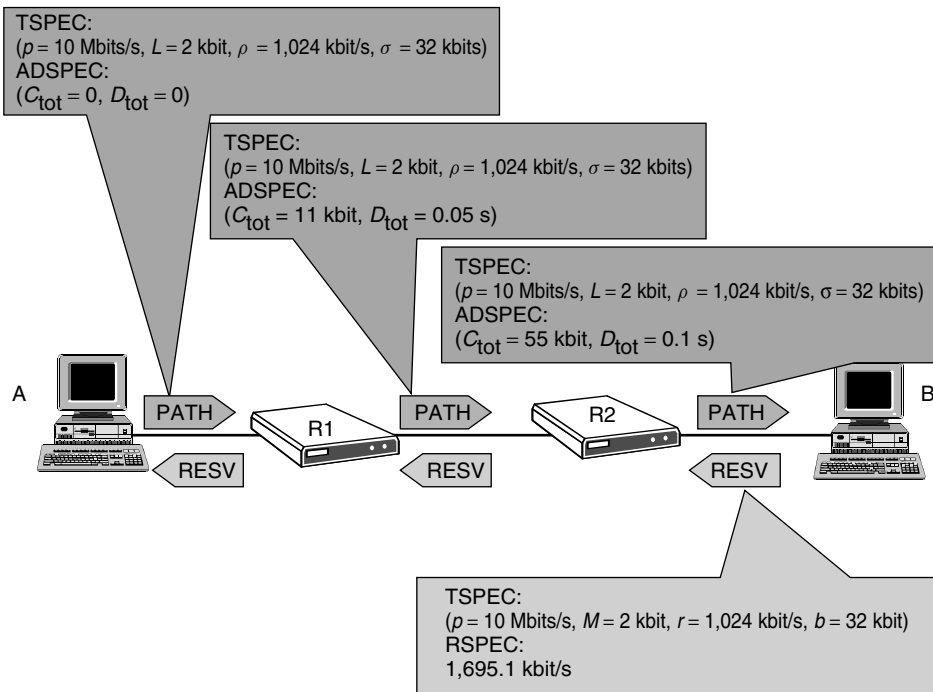


Figure 4.17 Updates to ADSPEC through the network.

- *ADSPEC*: ($C_{\text{tot}} = 0$, $D_{\text{tot}} = 0$).

The first RSVP router R1 keeps the TSPEC part of the PATH message unchanged but modifies the ADSPEC part (i.e., $C_{\text{tot}} = 11$ Kbit, $D_{\text{tot}} = 0.05$ s). The second RSVP router R2 relays TSPEC as it is and modifies ADSPEC (i.e., $C_{\text{tot}} = 55$ Kbit, $D_{\text{tot}} = 0.1$ s). The receiver B chooses the guaranteed QoS service to obtain a specific end-to-end delay. To find which reservation he needs as a function of the desired delay (which will always be greater than $D_{\text{tot}} = 0.1$ s), he solves the equation derived from the above results:

$$r = \frac{(p - \rho)(L + C_{\text{tot}}) + (\sigma - L)p}{(D_{\text{desired}} - D_{\text{tot}})(p - \rho) + \sigma - L} \quad \text{with the constraint } r > \rho$$

For $r = \rho$ the delay is simply $D = (\sigma + C_{\text{tot}})/\rho + D_{\text{tot}}$ or 0.185 s; so B can choose any delay between 0.1 and 0.184 s. Here are the results obtained with some reservation values:

Desired delay (s)	r to ask (kbits/s)
0.15	1,695.1
0.11	6,777.1
0.101	20,823.9

It is obvious that user B has to be reasonable and pay, because the reserved bandwidth R ‘explodes’ when the desired delay approaches D_{tot} !

B chooses a delay of 0.15 s and decides to request 1,695.1 kbit/s. It sends a RESV message with TSPEC = ($p = 10$ Mbits/s, $M = 2$ kbit, $r = 1,024$ kbit/s, $b = 32$ kbit) and RSPEC = 1,695.1 kbit/s toward A along the path followed by the PATH messages.

The receiver can also specify the ‘slack’ he can accept on top of D_{tot} . This is useful if B, for instance, ideally would like to have a low delay of 0.15 s but is prepared to accept a delay up to 0.185 s. This slack value is then transmitted in the RSPEC element. It can be used in a node if that node cannot allocate the requested bandwidth: in that case the node will ‘eat’ a part of the slack and pass on a decremented value, instead of rejecting the reservation.

RSVP can ‘work’ over non-RSVP clouds, since these clouds will forward PATH and RESV messages. But, these clouds are seen as direct links by RSVP, and the delay they introduce as well as the congestion state will not be reflected in PATH parameters: therefore, the RESV message will be wrong. This works only when the non-RSVP cloud is non-congested and is a low-delay area compared with the delay in the RSVP region.

4.4.2.5.2 Soft states

It would be a very bad idea to agree to make a reservation in a node if there is the slightest chance that this reservation is not properly terminated. In ATM or PSTN networks, this leads to rather complex, but safe, signaling. The reservations within an IP network are made even more complex because routes can change at any time such as when the network topology changes (e.g., after a link failure). This is generally considered a feature that gives IP a lot of robustness facing network failures. But when you consider a reservation along a given path, this becomes a serious issue. RSVP works around this problem by only making temporary reservations, which must be refreshed from time to time by the receiver of the stream: a reservation is a ‘soft state’ with a timeout.

Since PATH messages follow the path of the stream, they will follow the new routes. Therefore, new RESV messages, which follow the inverse path, will attempt to make reservations along the new route. The old reservations will not be properly terminated through signaling, but they will time out. In some cases it will not be possible to set up the reservation along the new path: for instance, if the network is too congested or if the policy along this path is different, but this should not happen very often in a well-designed network (the same situation could also occur in the PSTN if an important link broke down). However, there could be an adverse interaction between RSVP and dynamic routing algorithms assigning packets to the less loaded link: these algorithms should make sure that existing RSVP sessions remain intact (i.e., not change the route of PATH messages ‘on the fly’).

4.4.3 Scaling issues with RSVP

4.4.3.1 CPU limitations

RSVP itself is nothing more than a way to calculate and transmit the parameters that a node needs in order to perform a bandwidth reservation. RSVP is not responsible for

actually reserving bandwidth. A router must implement a scheduling or resource-sharing mechanism, such as PGPS. The class of algorithms that can support RSVP is generically called ‘weighted fair queuing’, but this name can apply to PGPS or other simplified schemes, such as SCFQ (self-clocked fair queuing). The resulting performance and actual delay limits obtained vary widely according to what is actually used.

The first difficult task is to be able to sort in real time all incoming flows based on the source address and port, and destination address and port. This function is commonly called a **multi-field classifier** (**MF classifier**). Many implementations are very sensitive to the number of flow filters that need to be recognized and have serious scalability limits when the number of filters exceeds a few dozens. Other implementations use optimized classification trees based on bit-per-bit analysis with a fixed convergence time that does not depend on the number of filters. There are even hardware-based real-time implementations.

The second difficult task is to schedule the packets to be served in an optimal order. A quick glance at the equations of PGPS gives an idea of the complexity of this task, which must be performed for every packet: PGPS, while it leads in theory to the tightest delay limits, requires a lot of processing power.

In reality, it is not the ‘complexity’ of RSVP that is the obstacle (actually, RSVP is simpler than many other signaling protocols), but the complexity of PGPS and other WFQ mechanisms. For instance, the most highly tuned kernel-mode Unix implementations of PGPS on a Pentium 166 (Ian Marsch, SICS) achieve a throughput at 90 Mbits/s over ten flows.

It is quite obvious that PGPS cannot be applied stream by stream in a backbone network. Most router vendors are using heuristics that approach the behavior of PGPS but require less CPU power. Still, it is impossible to scale per-stream queuing techniques to the throughput required in modern backbones. Backbones require approached techniques where all streams with similar properties and requiring similar priority handling are grouped and handled by the same queue.

4.4.3.2 Over-provisioning

A more fundamental problem is the fact that the hard-delay limits derived above have nothing to do with what is observed statistically. For a given reservation r , the delay observed would be *much* lower than the delay guaranteed by PGPS for almost all packets. The theory behind the guaranteed mode of RSVP leads to systematic over-reservations for all applications that can tolerate losing a delayed packet from time to time. It is expected that most applications, knowing this problem, will select the controlled load mode of RSVP and will simply use the D parameter of RSVP PATH as an indication of what delay they are going to experience (e.g., to set jitter buffer parameters): in this mode, the exact average bandwidth advertised by the source is reserved to avoid packet loss.

However, in the case of a link that carries much more best effort data than real-time data, the over-provisioning required by the guaranteed service is not as bad as it seems. This is a very common situation when we consider that peer-to-peer software agents and other fancy applets on our desktops can eat as much bandwidth as the developers think they need, while we cannot speak more than 24 hours a day. Most scheduling algorithms, including of course PGPS, are able to reallocate the bandwidth not actually used by a

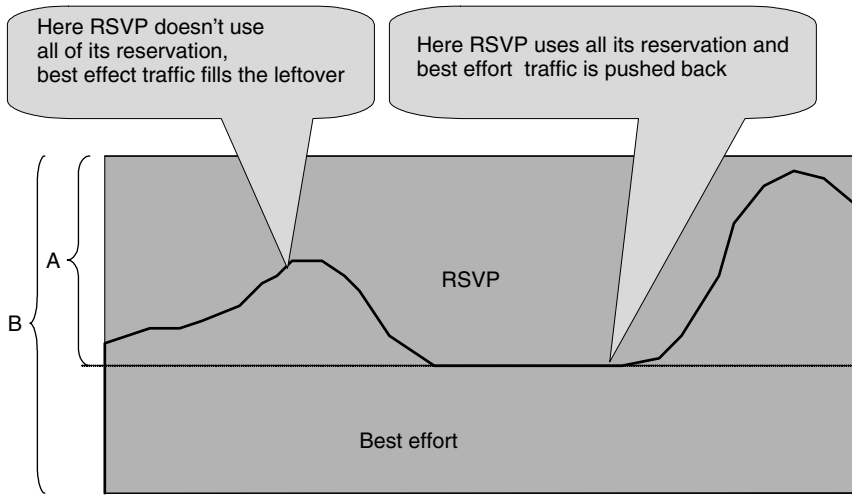


Figure 4.18 Best effort traffic uses unused reserved bandwidth. A: reserved bandwidth for RSVP streams; B: available bandwidth.

stream to the rest of the traffic: the extra bandwidth reserved but not actually used by RSVP will be used by best effort traffic and there is no waste (as shown in Figure 4.18). On links where real-time traffic is predominant, the network will refuse, based on PGPS theory, reservations that in practice it could have handled; this is indeed a problem. This problem is not specific to RSVP, it also occurs on ATM-based networks much more often, as these networks have mostly been used by professional organizations for applications very sensitive to QoS that have no best effort traffic.

4.4.3.3 State

RSVP is a connection-oriented technology. As such it requires network nodes to maintain state information about each connection in the network. This is a fundamental move from the connectionless paradigm of IP and probably the source of much of the debate around RSVP. Any other connection-oriented technology has the same problem: basically, these techniques do not scale as the number of connections increase through the network. In the optimal case of multicast flows, the amount of state information scales as the number of peripheral end points. In the case of unicast traffic, the amount of state information scales as the square of the number of peripheral end points!

Compared, for instance, with ATM, RSVP has its advantages and drawbacks. The one major advantage is that an IP network implementing RSVP requires state information only about QoS connections; so, if most of the traffic only needs best effort transport, it is only a small subset of the overall traffic. By comparison ATM will use a connection per stream, regardless of whether the stream has requested QoS or not (UBR connections). But ATM is based on persistent states: there needs to be signaling activity only at connection set-up and tear-down. Since RSVP is based on soft states, each connection needs periodic

signaling activity; so, the work required for the same amount of streams is much higher for RSVP than it is for ATM.

In the current state of the specification, it is hard to decide whether RSVP is really better or worse than other connection-oriented QoS techniques. However, there is an interesting perspective on the future of RSVP which could improve its scalability: using it mostly as an edge reservation request mechanism and thus simplify QoS management in the core network. This new layered model is demonstrated in Section 4.4.4, and a practical use of this model is discussed in Section 4.5.4.3.2.

4.4.4 Scaling RSVP with a layered architecture

The solution that emerges for the use of RSVP on commercial networks is to employ it as a layered architecture that utilizes the fact that RSVP messaging can cross a non-RSVP cloud. This solution focuses on the business requirements for RSVP: in short, use RSVP where it is very useful, and avoid it as much as possible when it is not strictly necessary. The business requirements are:

- (1) To give applications a uniform way to ask for a level of QoS and describe data streams, in order to minimize manual configuration and management tasks.
- (2) To find a way to guarantee a level of QoS, end to end, for each application that has requested it.
- (3) To provide authentication and facilitate billing.
- (4) To provide the necessary statistics in order to help the backbone provider properly dimension its network.

Requirements (1), (3), and (4) are for access nodes only. The only requirement that seems to require RSVP in the backbone is (2). If we can find another way to guarantee QoS in the backbone, then RSVP could be used only at the access network.

With these remarks, it is logical to divide the network into two separate concentric layers:

- In the outer layer, RSVP is used with flow-by-flow WFQ, which is possible because the bandwidth and the number of streams are still low. In addition, access routers deal with security and generation of accounting information for billing.
- In the core network, streams with ‘similar properties’ (see Section 4.4.4.1) and facing similar constraints in terms of delay and required bandwidth are grouped using classes of service. Core routers do not perform any per-flow accounting or policing, and are tuned to achieve a maximal throughput with minimal delay.

4.4.4.1 Flow grouping in the backbone

Section 4.3.3.2 summarizes results that relate to the delay limits achievable using a separate virtual queue for each stream scheduled using PGPS weighted fair queuing.

However, this scheme is not scalable due to the amount of processing power required. In this section we try to evaluate the impact of grouping several streams together in a single queue in the backbone.

In this section we call ‘similar’ those streams having similar characteristics in terms of burstiness/average rate ratio and maximum packet size. A ‘class of streams’ is composed of all streams i whose characteristics fall within the following limits:

- Average rate $k_i(\rho \pm \Delta\rho)$.
- Maximal burstiness $k_i(\sigma + \Delta\sigma)$.
- Maximal packet size L_{\max} is supposed to be common to all streams in this class.

We now suppose that N streams fit in this definition (e.g., the rate and burstiness chosen are typical of the G.723.1 sound channels of IP phones). The resulting combined stream will have an average rate of $(\rho \pm \Delta\rho) \sum_i k_i$ and a burstiness lower than $(\sigma \pm \Delta\sigma) \sum_i k_i$. With these results we can calculate a delay bound at each backbone hop, where c means ‘class’:

$$D_c^* \leq \frac{(\sigma \pm \Delta\sigma) \sum_i k_i}{(\rho \pm \Delta\rho) \sum_i k_i} + \frac{L_{\max}}{C} + Tr = \frac{(\sigma \pm \Delta\sigma)}{(\rho \pm \Delta\rho)} + \frac{L_{\max}}{C} + Tr \quad (A)$$

assuming that the aggregate stream is assigned a portion of total bandwidth C equivalent to $(\rho \pm \Delta\rho) \sum_i k_i$ (stability condition).

If each individual flow had been assigned a separate queue under PGPS and a capacity equal to the average rate, the result would have been:

$$D_i^* \leq \frac{\sigma_i}{\rho_i} + \frac{L_{\max}}{C} + Tr \quad (B)$$

Equations (A) and (B) give very similar results, which shows that grouping similar streams in the backbone does not cause any significant loss in guaranteed end-to-end delay. Moreover, (A) is really a worst case bound since it assumes burstiness is an additive parameter: in reality, when each individual stream is independent the resulting burstiness is much lower. How much lower depends on the exact nature of the data streams, but for a sum of periodical streams with random phase it can be calculated exactly. In Figure 4.19 the sporadic nature of source s is defined as $\text{MaxThroughput}(s)/\text{AverageThroughput}(s)$, where $\text{MaxThroughput}(s)$ is defined as the level that will be exceeded by $\text{Throughput}(s, t)$ with a probability lower than 10^{-9} .

If we now consider end-to-end delay, grouping several ‘similar’ flows is very beneficial. One reason is that the maximum burst size of the aggregate stream is likely to be much lower in proportion to the aggregate bitrate. In addition, in the end-to-end formula given for PGPS through H hops, one of the components of the delay was $(H - 1)L_{\max}/r$, with L_{\max} the largest datagram size: when grouping several flows, this now becomes $(H - 1)L_{\max}/Nr$, which is much less!

If the grouped flows are similar in sporadic nature, less bandwidth is needed to achieve the same delay limit with a very high probability. This suggests that streams in the backbone with a similar sporadic nature at a similar priority level should be grouped.

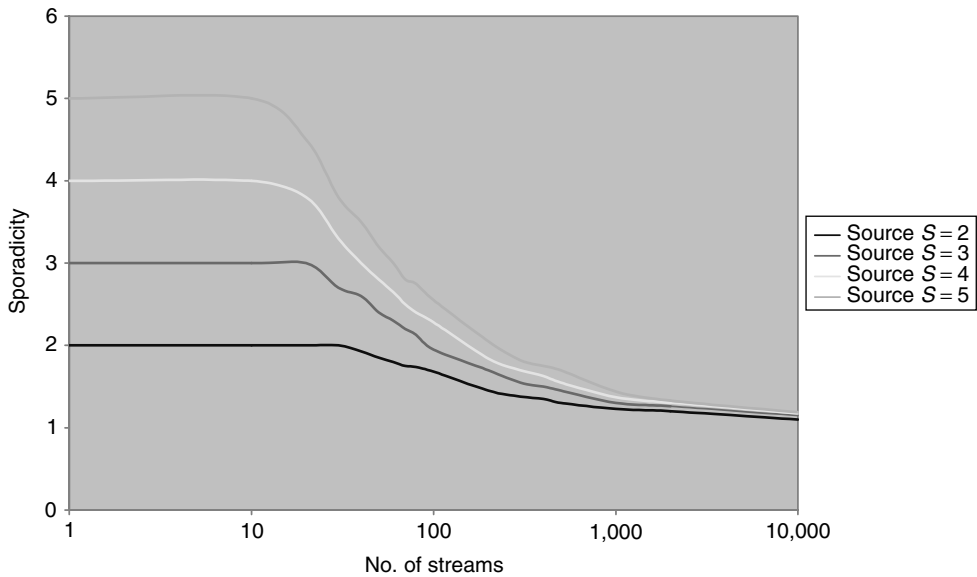


Figure 4.19 Sporadicity of aggregate streams.

4.4.4.1.1 Bandwidth management

In the previous section it was assumed that N streams were grouped. In reality, this number would vary with time, and it would be difficult to adjust dynamically the bandwidth reserved for this traffic class each time N changes. A possible heuristic for bandwidth reservation could be some over-provisioning of bandwidth, which would be incremented if the class uses more than 90% of the bandwidth or decremented if it uses less than 70% (the threshold given here is arbitrary and should be derived from effective dispersion of used bandwidth within the class). This hysteresis would reduce the number of bandwidth reservation changes in the backbone for that class.

Using class-by-class WFQ with some bandwidth, over-provisioning is an acceptable waste of bandwidth in most cases, since the unused reserved bandwidth is always available for best effort traffic at any time. On most IP backbones, best effort packets represent the vast majority of the traffic.

4.4.4.2 Using DiffServ with RSVP tunneling

The simplest way to simplify the provision of QoS in the backbone and avoid per-flow state is to ignore RSVP completely, relying on the simpler DiffServ architecture. DiffServ is an approach where all flows carried by the backbone are grouped in several classes of service, eliminating most of the scalability issues encountered by per-flow QoS provisioning. There are many ways to implement classes of service in the core:

- Using IP TOS/DS: internal routers can use class-by-class WFQ, where each class is determined according to the precedence bits of IP packets.

- Using layer 2 capabilities:
 - Providers with an ATM backbone can open several virtual channels (VCs) between concentration routers with several levels of QoS or simply use the ATM CLP (cell loss priority) bit to define two rough classes of service.
 - Providers with a frame relay backbone can open several VCs between border routers with various levels of QoS parameters or use the DE (discard eligible) bit.
 - Providers with MPLS (multiprotocol label switching) backbones can also define one level of QoS for each label.

These techniques can be combined with congestion control mechanisms, such as RED or WRED, to smooth TCP traffic and improve fairness between the flows within each class of service.

We have seen that RSVP is an end-to-end protocol; so, RSVP messages need to be passed between hosts across the backbone. This is not as simple as it seems. First, RSVP messages must be ignored in the backbone, otherwise we would run into scalability issues. Second, each RSVP packet is marked with a 'router alert' option in order to help routers identify this packet as one needing special treatment (the router alert option should be turned off, or ignored, when passing along the backbone). Some implementations (e.g., ISI) do not rely on the router alert option, but rather on interception of packets with protocol 46. In this case it is also possible to use a new IP protocol number that only access routers would recognize, or tunnel RSVP packets in an IP tunnel through the backbone.

Figure 4.20 shows a layered network architecture in which RSVP is used at the edge for admission control and as a way for applications to declare data flow properties. Data flows are then grouped into traffic classes. Core layer QoS mechanisms only consider these traffic classes.

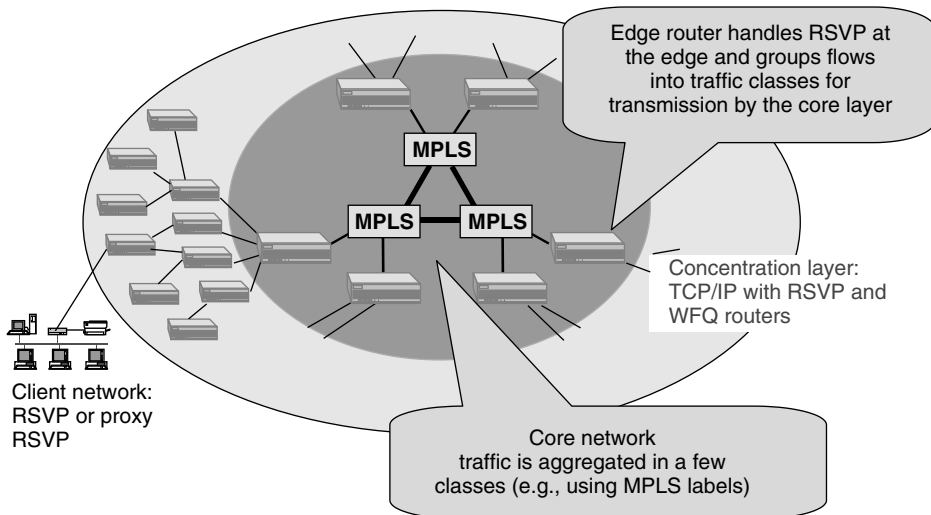


Figure 4.20 Scalable QoS using a layered architecture.

4.4.4.3 Handling of RSVP messages through the backbone

The RSVP and the DiffServ paradigms are based on opposite models: RSVP gives precedence to a stream based on the receiver's wishes, while DiffServ prioritization is controlled by the sender.

In the layered model, an edge router sits between the DiffServ domain and the IntServ (RSVP) domain. This edge router can modify the TOS/DS value of all packets injected in the backbone and, therefore, has complete control over the priority of these packets. In order to emulate receiver-based RSVP behavior, this router must decide which class of service to use for a flow based on the QoS requirements contained in the RESV messages (as shown in Figure 4.21). In this case, after analyzing the first RESV message, the edge router decides that this session must be aggregated into the 'medium' class of service and all the packet parts of this stream are marked accordingly.

4.4.4.3.1 PATH messages

For the end-user application, the network must behave as if it was RSVP-enabled end to end. Therefore, the PATH messages generated by the sender must cross the backbone and reach the receiver(s). Having received a PATH message describing a flow, the receiver may choose to send back a RESV message in order to reserve resources for this flow in the backbone.

However, if an implementation transmits PATH parameters transparently the receiver will have a false view of the backbone, because the parameters within the PATH messages will not have been updated to reflect latency and other characteristics of the backbone.

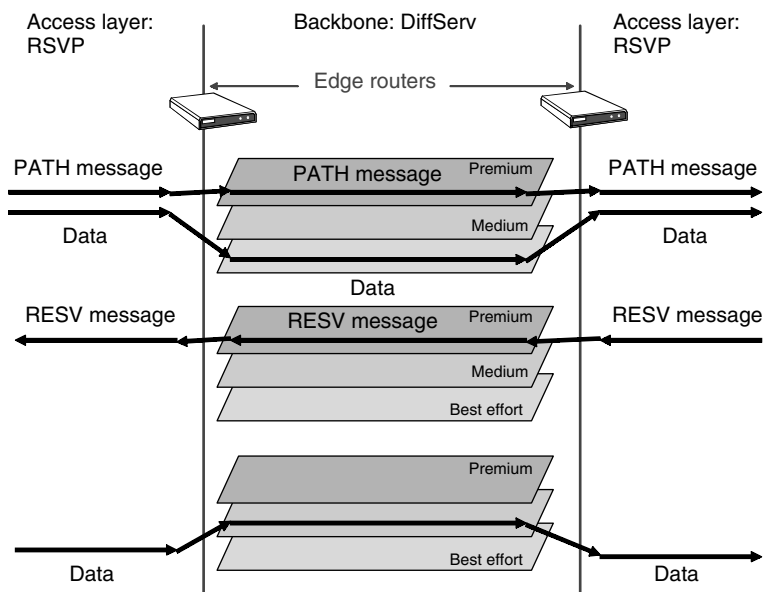


Figure 4.21 Handling an RSVP reservation through a DiffServ core.

Therefore, the receiver might be misled, thinking the backbone adds less delay than it actually does, and make a wrong reservation (in ‘guaranteed service’ mode).

If the backbone is built using powerful ‘gigarrouters’, then each PATH message can be updated at each hop. On the other hand, if the backbone is built using ATM or MPLS technology, the PATH message needs to be updated only once by an edge router that evaluates the overall transmission delay along the virtual circuit to the destination of that stream, making the backbone appear as a single node for RSVP. A practical approach is to propagate the value of the C parameter ‘as is’ and update only the D parameter. In other words, we consider the queuing delay added by the backbone not to be very sensitive to the characteristics of individual streams, but able to be approximated by an absolute delay value that does not depend on the capacity reserved for the individual stream. This approximation is generally valid if the number of aggregated streams is large. A single stream has a negligible influence on backbone transit delay. Therefore, we update D with a value representing the propagation delays and the average queuing delay through the backbone.

4.4.4.3.2 *RESV messages*

If the backbone uses routers that have enough processing power to handle RESV messages hop by hop, then each RESV message can be propagated normally through the backbone. But, instead of creating a separate queue for the stream as in regular RSVP operation, the router will direct that stream to the queue that is most appropriate for this type of session and the required class of service (as discussed in Section 4.4.4.1). These routers will maintain the bandwidth usage information for each class of service and eventually decide to add more bandwidth for a particular class.

For most backbones, the edge router receiving a RESV message will simply tunnel this RESV message to the edge router used by that stream to enter the backbone. This router will then be responsible for properly choosing the DiffServ DSCP field for packets of this stream. The Diffserv codepoint will be used to choose a specific traffic class for the backbone and may be mapped to layer two QoS mechanisms (e.g., in MPLS tags).

4.4.4.4 *Caveats*

With this layered approach, the ingress edge router will aggregate several flows over each class of service. This will not have a significant impact on the QOS level experienced by each individual flow except in some situations:

- When one of the flows does not conform to its TSPEC: if the provider has dimensioned each virtual link between the ingress and egress routers for an aggregate of flows with well-defined characteristics, one non-conformant flow may be enough to ruin the quality of service experienced by all other flows on the same class.
- When very sporadic flows are merged with smooth flows (e.g., video with voice).

Therefore, the access router must have the ability to check the TSPEC of each flow (and destroy or mark out-of-profile packets), and the service provider must avoid, when defining the classes of service of the backbone, merging sporadic and smooth flows.

4.5 The CableLabs® PacketCable™ quality-of-service specification: DQoS

Although simpler than the framework described in Section 4.4.4, the DQoS framework is one of the most advanced examples of a layered network architecture providing tight per-flow QoS control at the edge, while requiring only broad classes of service in the core backbone. Compared with the RSVP tunneling method described above, DQoS differs slightly by offering the option to terminate RSVP locally, in which case RSVP is used only as a local admission control and service-level request protocol, and not for end-to-end QoS provisioning.

4.5.1 What is DQoS?

Dynamic quality of service (DQoS) defines an architecture and a set of protocols to provide assured quality of service in the access portion of a cable network. DQoS has been specified within the PacketCable™ project of the CableLabs® consortium. The specification is publicly available on the PacketCable™ website (www.packetcable.com).

The access portion of a cable network is defined as the portion between the **MTA** and the **CMTS** (Figure 4.22):

- The MTA (multimedia terminal adapter) is the component that generates multimedia streams. It can be a PC, a stand-alone VoIP gateway for analog phones, an IP phone,

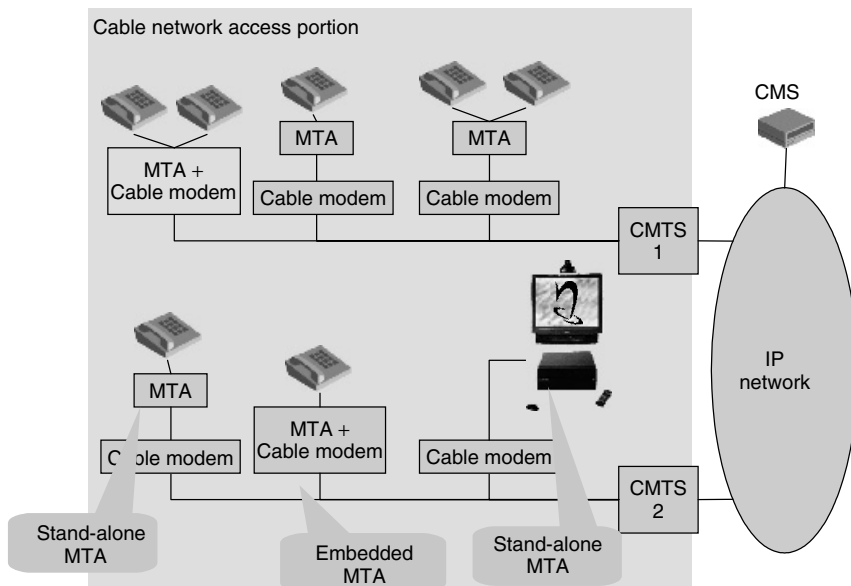


Figure 4.22 The PacketCable® ecosystem.

etc. The MTA is connected to a cable modem (CM). If the MTA and the cable modem are in the same device, it is called an ‘embedded MTA’, otherwise it is a ‘stand-alone MTA’, which can be connected to the cable modem (e.g., using an Ethernet cable or a USB cable).

- The CMTS (cable modem termination system) is responsible for collecting all data streams from cable modems and routing the data either to an external IP network or back to a cable modem. The CMTS is the only trusted entity of the access portion.

The MTAs are controlled by a **call management server (CMS)**, using one of the two call-signaling protocols defined by PacketCable™: **NCS (network-based call signaling)**, a variant of MGCP or **DCS (distributed call signaling)**, a variant of SIP). Today the market is almost completely NCS. The CMS also sets the proper QoS authorizations (**gates**) on the CMTS; this is the **gate controller** function in the DQoS framework.

Between the MTAs and the CMTS, two mechanisms may be used for QoS control:

- A layer 2 mechanism (DOCSIS 1.1 MAC), which is the only mandatory mechanism in DOCSIS 1.1 and is only available for embedded MTAs.
- A layer 4 mechanism based on an extended version of RSVP, called RSVP+, which is available both for embedded or stand-alone MTAs. In PacketCable™ versions after 1.0, RSVP+ support is required for non-embedded MTAs and the CMTS. For embedded MTAs, the RSVP+ or the MAC mechanism must be supported.

DQoS provides quality of service on a segment-per-segment basis; that is, on a call from CMTS1 to CMTS2, quality of service will be performed independently on the CMTS1 DOCSIS segment and on the CMTS2 DOCSIS segment, possibly using different mechanisms (MAC- or RSVP-based). Each service provider and each segment can select its own preferred DQoS mechanisms. Optionally, the backbone segment between the two CMTSs can also implement a QoS mechanism (IntServ or DiffServ), but the protocols used at this level are not within the scope of DQoS.

4.5.2 Session-per-session QoS reservation

DQoS allocates resources to the flow of data between two applications running on separate endpoints. Bidirectional² data communication is called a *session* (Figure 4.23). The QoS is provided individually to both the unidirectional upstream data flow and the unidirectional downstream data flow within a session. The QoS is provided only to authorized sessions, and for each session usage can be monitored: these accounting data enable usage-based charging.

The term ‘dynamic’ is used because the QoS policy category of a user can change (e.g., ‘gold’ to ‘bronze’) without resetting the cable modem. The QoS settings of a session can

²This terminology is specific to DQoS. In the rest of the chapter the word ‘session’ refers to a monodirectional data stream

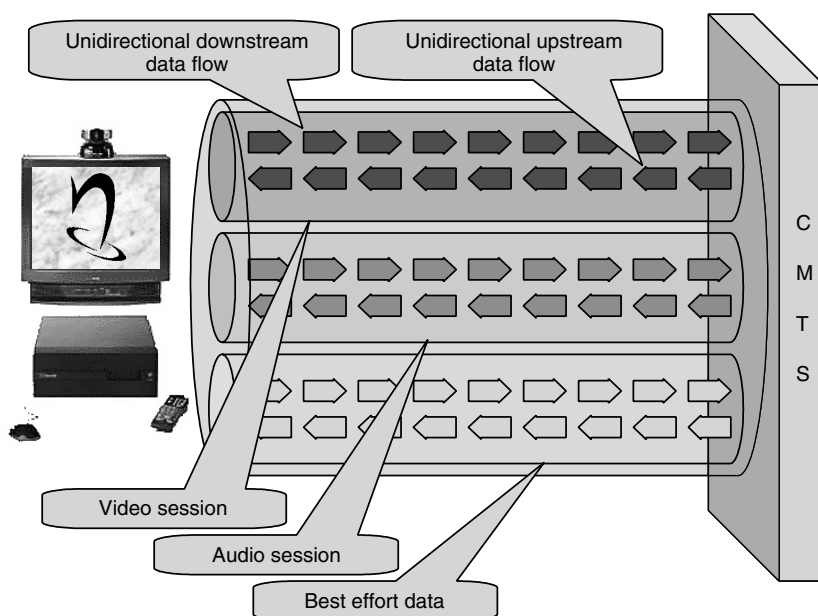


Figure 4.23 Flows and sessions.

change in the middle of a session. For instance, if there is a codec change from G729 to G711 for a fax, the MTA can dynamically commit more bandwidth if it had already reserved enough capacity or can attempt to reserve more bandwidth if the reserved capacity is not enough.

The CMTS allocates and schedules the bandwidth corresponding to each session reservation, performing the role of the **policy enforcement point (PEP)**, using the terminology of the IETF **Resource Allocation Protocol** framework defined in **RFC 2753**). The CMTS performs this function by implementing a **DQoS Gate** (a set of packet classification and filtering functions, as described in Figure 4.24) for each session between the cable network and the IP backbone or between MTAs on the cable network. Each CMS implements a gate controller function that installs and controls gates on the CMTS: this is the policy decision point (PDP) in the IETF Resource Allocation Protocol framework. If a gate is closed, then the traffic will either be dropped or forwarded in the best effort class, depending on service provider policy.

The CMTS is responsible for providing the QoS requested by cable modems if allowed to do so by the current policy, allocating the proper upstream capacity on the shared medium. At layer 2 of the cable network, the CMTS uses the mechanisms defined in DOCSIS 1.1 at the MAC level (not covered here) to implement the QoS policy. At the IP level, the CMTS also verifies that IP streams sent by cable modems are properly shaped, and verifies and eventually resets the DSCP of the IP packets it sends to the backbone. In the other direction, the CMTS classifies the IP packets coming from the backbone interface, applies traffic shaping, and DSCP marking before forwarding these packets on the cable network.

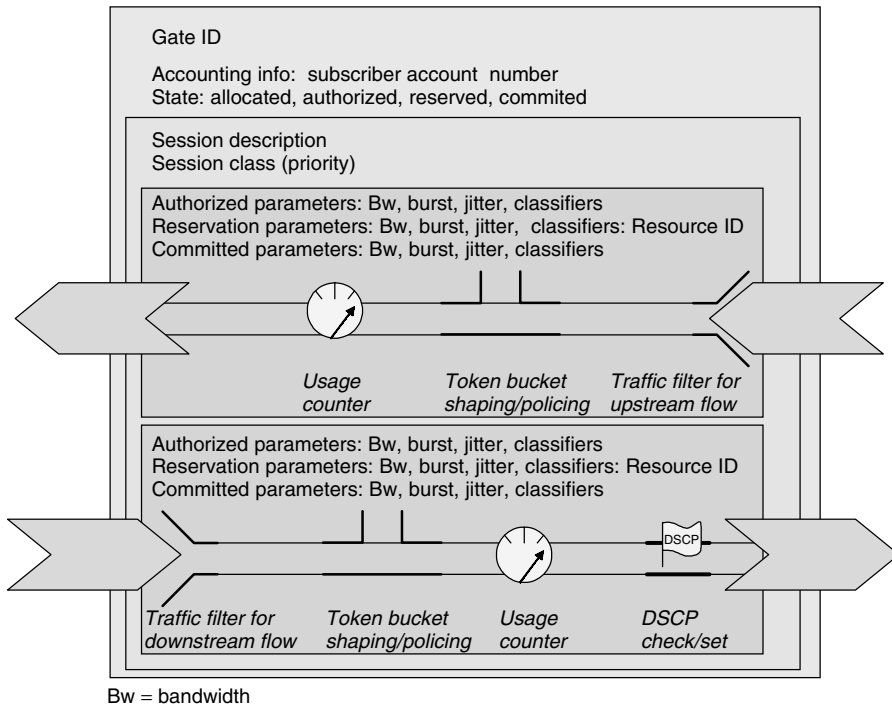


Figure 4.24 The DQoS 'gate'.

The cable modem is responsible for implementing DOCSIS 1.1 QoS mechanisms to request and obtain QoS on the cable network. It must also classify upstream IP packets according to the filters declared to the CMTS and shape them properly according to the declared token bucket parameters. Since the cable modem is not trusted, the CMTS will double-check that the flow conforms to the profile declared in the gate for this session.

4.5.3 Two-phase reservation mechanism

Figure 4.24 shows that the gate can have several states:

- *Allocated.* The gate has been created at the initiative of the gate controller, but not yet initialized with resource authorization parameters.
- *Authorized.* The gate controller has set the maximum level of resources (envelope) that can be reserved within this gate. This restriction applies to any subsequent reservation request. A gate controller can change an authorization for a given gate, but this only applies to future reservation requests. Since it establishes gates and gate authorizations in advance of a resource request, the gate controller can remain unaware of the state of sessions in progress (stateless model). For voice traffic, a permanent gate is allocated to signaling flows and a per-call gate is established for each call.

- *Reserved.* This state corresponds to admission, the first phase of the QoS reservation mechanism. Reserved resources are available to best effort traffic until they committed, but they are removed from the pool of resources available for admission control. Usually, the MTA is responsible for reserving resources. Reservation is soft state and will expire unless it is refreshed. At the end of reservation, DOCSIS link service flows are in 'admitted state'.
- *Committed.* The gate is in committed state once an MTA has sent a confirmation that the resource is being used. The gate is only open after reservation has been committed, preventing fraud and theft of service. Excess reserved resources beyond committed resources are released unless the MTA periodically refreshes the reservation (e.g., necessary for calls on hold). At the end of the commit phase, DOCSIS link service flows are in 'active' state and usage recording starts. In practice, the IP flows of a session in 'committed' state use a WFQ-prioritized waiting queue, which ensures they will receive a guaranteed level of service. If, however, the effective resource usage of these streams remains below the committed capacity, then the scheduling algorithm of the CMTS, if appropriately designed, will enable best effort traffic to use leftover capacity (this is the case with PGPS scheduling, compare Section 4.3.3.2).

The binding between a reservation and a session is dynamic. For instance, for calls on hold, a reservation will be used for the active call and the associated session switches to the held call when it becomes active.

When reserving resources for a session, a session class is specified, allowing the CMTS to keep resources for high-priority traffic (e.g., emergency calls). Session classes may be overlapping (e.g., 50% is the maximum for normal calls and 70% for emergency calls) and may be pre-emptive. The session class is communicated in the Gate-Set request of the CMS to the CMTS together with the authorized session envelope.

The MTA is not forced to commit resources if there is a reservation or to reserve resources that have been authorized. For instance, if port-to-port calls remain on the same MTA (same IP address), IP packets are not forwarded on the cable network. This situation could be recognized by the CMS, in which case the CMS sends a connection request without a Gate-ID, indicating that the MTA must not reserve or commit resources. But, in the most common case such a situation is not necessarily known at the beginning of the call. If a Gate-ID was communicated to the MTA and the far end was still not known, the MTA must tear down the service flow (reservation and/or commit) once the port-to-port call is recognized.

4.5.3.1 Separate MTA and CM: the MTA to CMTS QoS protocol

The admission part of the QoS reservation two-phase process uses a superset of RSVP, named RSVP+. The main difference is that, unlike RSVP, a PATH message from the cable modem in RSVP+ requests resources in both directions (upstream and downstream), and a RESV message from the CMTS confirms the reservation request has passed admission control for both directions (Figure 4.25). In RSVP, the PATH message applies only to the stream sent by the PATH sender.

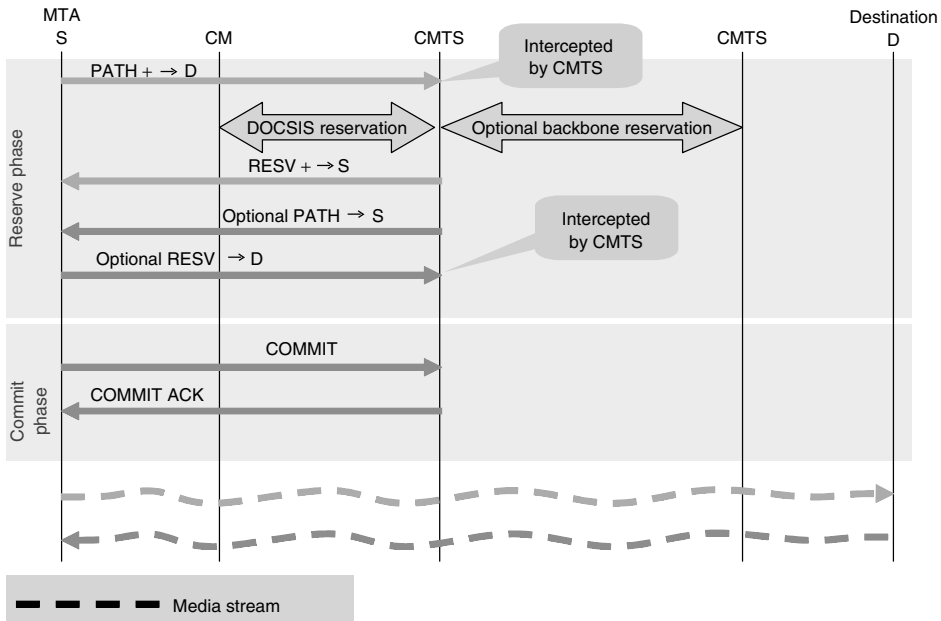


Figure 4.25 Use of RSVP+.

The PATH message of RSVP+ is sent to the same destination as the data flow, as in RSVP, but it is intercepted by the CMTS and therefore never reaches the destination. In order to preserve compatibility with RSVP routers that may be present between the CM and the MTA, the CMTS may also send a PATH message toward the MTA.

SDP information sent in the call control protocol from the CMS to the MTA contains enough details to allow calculation of an RSVP flowspec for well-known codecs. For other codecs, SDP can include explicit bandwidth information:

```

b:<modifier CT for Conference Total or AS for Application
  Specific>:<bandwidth value including IP/UDP/RTP overhead>
    
```

RSVP+ exchanges also cause the CMTS to initiate DOCSIS MAC-level QoS reservation with the cable modem.

The commit phase is performed by the MTA using a separate COMMIT UDP message (not an RSVP message) sent to an address specified in the RSVP + RESV message. It is acknowledged by a COMMIT ACK.

4.5.3.2 *Embedded MTA: the optional MTA to CMTS protocol – the one used in practice*

An embedded MTA may use the MAC control service interface described in the DOCSIS 1.1 RFI specification, using DSx messages to reserve resources on the cable plant instead of the RSVP+ interface. This mechanism cannot be used if the MTA and the cable modem

are separate entities. In practice, in all cable networks today the MTA and the CM are integrated, in order to simplify the provisioning of telephony lines to customer premises and to ensure that the service will be maintained for emergency calls using the batteries of cable modem. RSVP+ then becomes useless, like the rest of the DQoS framework, because the CM is considered, in practice, a trusted element of the network. Therefore, contrary to what is sometimes heard, DQoS is not a prerequisite for the deployment of telephony over cable networks. It is only required if the cable network operator wishes to provide service to non-integrated MTAs over the end customer LAN. Unfortunately, there are no—to our knowledge—non-integrated MTAs with support for RSVP+ on the market. It seems the market for non-integrated MTAs is too small to justify developing specific versions of analog VoIP gateways for cable networks. This is the reason the DQoS framework remains mainly an interesting theoretical exercise, one that certainly prefigures future developments for next generation broadband networks. But, in reality it is of little use on current cable networks (except as an argument to sell next generation CMTS systems).

4.5.4 CMS to CMTS communications

The CMS and CMTS communicate using the IETF Common Open Policy Service (COPS) protocol.

4.5.4.1 The COPS protocol

COPS is defined in RFC 2778. It is designed to convey control communications between one or more policy enforcement points (PEP) and a policy decision point (PDP). These terms are defined by the IETF framework for policy-based admission control (RFC 2753): a PEP is responsible for executing a security policy set by the PDP. PEPs filter and classify IP packets, and may need to mark them with appropriate DSCP codes or to perform shaping on the data streams. They can also accept or reject RSVP QoS requests. The COPS protocol is not completely specified and needs to be profiled for any given application. In the following sections we focus on the DQoS profile.

4.5.4.1.1 Generic COPS message format

The COPS protocol is based on messages that begin with a specific header and contain multiple encapsulated objects (Figure 4.26). The C-Num octet identifies the type of object from among the 16 types of objects used by COPS, while the C-Type octet identifies the subtypes of a given object. The objects used by the PacketCable™ DQoS are indicated in bold in Table 4.2.

Most of the objects in Table 4.2 are placeholders for information which must be specified separately in a COPS profile. In the case of DQoS, a summary of the profile can be found in Section 4.5.4.2.

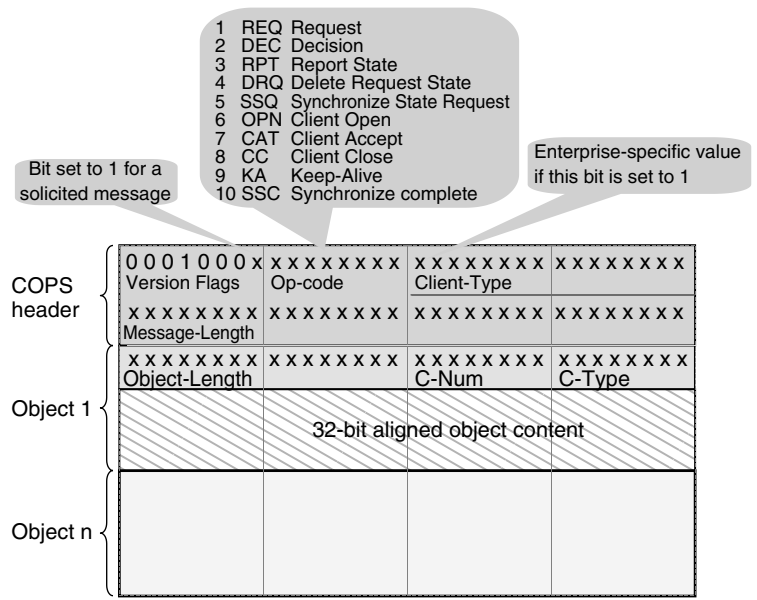


Figure 4.26 COPS message format.

4.5.4.1.2 Generic COPS operations

PEPs initially connect with a PDP using a TCP socket to port number 3288 and then begin the COPS session by sending a Client-Open (OPN) message. The PDP can either redirect the PEP to another PDP (Client-Close message with a <PDPredirAddr> object) or accept the connection by sending a Client-Accept (CAT) message. The integrity of the connection is then checked periodically using Keep-Alive (KA) messages, until the connection is closed with a Client-Close message.

The PEP then sends Request (REQ) messages to the PDP, which installs a QoS-related state (specified in the COPS profile) in the PDP. The PDP replies to each request with a Decision (DEC) message. Each Request is identified by a client handle object, generated by the PEP, which is copied in the PDP DEC message. This handle is associated with the state installed by the request, and the PDP can send new DEC messages regarding that state until the client handle is explicitly closed by the client (using a Delete Request State, or DRQ, message). COPS messages are illustrated in Figure 4.27.

4.5.4.2 COPS profile for DQoS

The DQoS COPS profile is specified in sect. 5.2 of the DQoS specification. In DQoS, the gate controller (PDP) initiates the COPS connection by establishing a TCP connection to the IP address of the CMTS (port 2126). DQoS PEPs do not support PDP redirections in Client-Close messages. The COPS Client-Type for a DQoS PEP is 0x8008.

Table 4.2 COPS specific objects

C-Num	Object type	C-Type and subtypes
1	Handle: unique value that identifies an installed state	1: Client handle
2	Context: the type of event that triggered a query	1: Request-Type/Message-Type
3	In interface: incoming interface on which a request applies	1: IPv4 address + Interface 2: IPv6 address + Interface
4	Out interface: outgoing interface on which a request applies	1: IPv4 address + Interface 2: IPv6 address + Interface
5	Reason code: reason for a delete request message	1
6	Decision: decision made by a PDP	1: Decision flags 2: Stateless data 3: Replacement data 4: Client-specific decision data 5: Named decision data
7	LDPD decision: decision made by a PEP-local PDP	Same as Decision
8	Error: identifies a COPS protocol error	1
9	Client-specific info (SI)	1: Signaled client SI 2: Named client SI
10	Keep-Alive timer	1: KA timer value
11	PEP identification	1: ASCII string
12	Report type	1
13	PDP redirect address	1: IPv4 address + TCP port 2: IPv6 address + TCP port
14	Last PDP address	1: IPv4 address + TCP port 2: IPv6 address + TCP port
15	Accounting timer	1: value
16	Message integrity	1: HMAC digest

As shown in Figure 4.28, DQoS really uses COPS as a transport or tunneling protocol, without really using its semantics: it uses a single COPS request and handle, and then uses COPS DEC messages and RPT messages to exchange DQoS-level messages:

- From the GC to the CMTS: DQoS primitives are placed inside the Decision Object (object C-Num 6, C-Type 4) of a COPS DEC message.
- From the CMTS to the GC: DQoS primitives are placed inside a Signaled Client SI object (C-Num 9, C-Type 1) of a COPS RPT message.

The Context object in the COPS Decision message has the R-Type (Request Type Flag) value set to 0x08 (Configuration Request) and the M-Type set to zero. The Command-Code field in the mandatory Decision-Flags object (C-Num 6, C-Type 1) is set to 1 (Install Configuration).

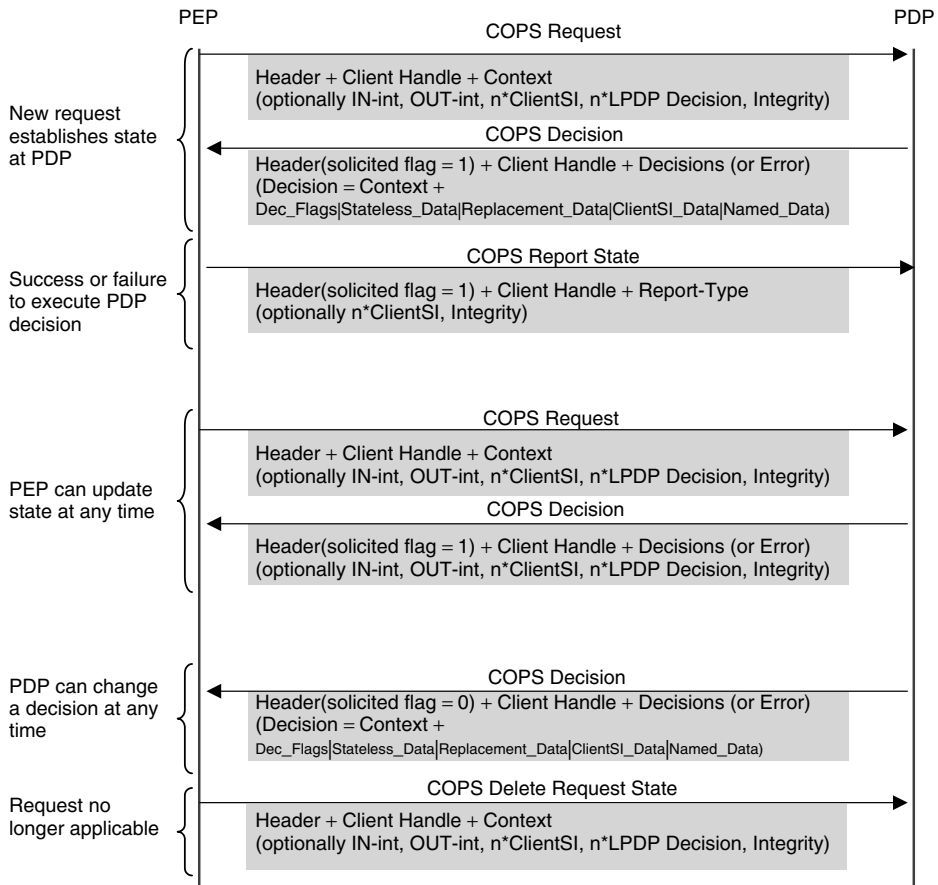


Figure 4.27 PEP to PDP dialog.

4.5.4.2.1 DQoS commands

The following commands are defined by DQoS (encapsulated DQoS-specific subobjects are shown in *italic*):

- GC-initiated messages to CMTS:
 - **<Gate-Alloc-Command>**=<COPS Header> <Handle> <Context> <Decision-Flags> <Decision-Header> <TransactionID> <Subscriber-ID> [<Activity-Count>]. Gate-Alloc validates the number of simultaneous sessions allowed to be set up from the originating MTA and allocates a Gate-ID to be used for all future messages regarding this gate.
 - **<Gate-Set-Command>**=<COPS Header> <Handle> <Context> <Decision-Flags> <Decision-Header> <Transaction-ID> <Subscriber-ID> [<Activity-Count>] [<Gate-ID>] [<Event-Generation-Info>] [<Electronic-Surveillance-Parameters>]

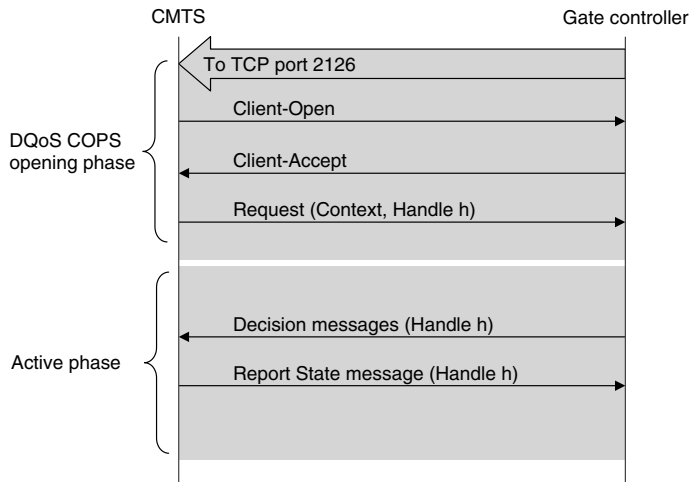


Figure 4.28 Opening a DQoS connection.

[<Session-Description-Parameters>] <Gate-Spec> [<Gate-Spec>]. Gate-Set initializes and modifies all the policy and traffic parameters for the gate or set of gates, and sets the billing and gate co-ordination information.

- <**Gate-Info-Command**>=<COPS Header><Handle><Context><Decision-Flags><Decision-Header><Transaction-ID><Gate-ID>. GATE-INFO is a mechanism by which the gate controller can discover all the current state and parameter settings of an existing gate or set of gates.
- <**Gate-Delete-Command**>=<COPS Header><Handle><Context><Decision-Flags><Decision-Header><Transaction-ID><Gate-ID><PacketCable-Reason>. Gate-Delete allows a gate controller to delete a recently allocated gate under certain circumstances.
- CMTS to GC responses:
 - <**Gate-Alloc-Ack-Response**>=<COPS-Common-Header><Handle><Report-Type><ClientSI-Header><Transaction-ID><Subscriber-ID><Gate-ID><Activity-Count>.
 - <**Gate-Alloc-Err-Response**>=<COPS-Common-Header><Handle><Report-Type><ClientSI-Header><Transaction-ID><Subscriber-ID><PacketCable-Error>.
 - <**Gate-Set-Ack-Response**>=<COPS-Common-Header><Handle><Report-Type><ClientSI-Header><Transaction-ID><Subscriber-ID><Gate-ID><Activity-Count>.
 - <**Gate-Set-Err-Response**>=<COPS-Common-Header><Handle><Report-Type><ClientSI-Object>.
 - <**Gate-Info-Ack-Response**>=<COPS-Common-Header><Handle><Report-Type><ClientSI-Header><Transaction-ID><Subscriber-ID><Gate-ID>[<

Event-Generation-Info>][<Electronic-Surveillance-Parameters>] [<Session-Description-Parameters>][<Gate-Spec>] [<Gate-Spec>].

- **<Gate-Info-Err-Response>**=<COPS-Common-Header><Handle><Report-Type><ClientSI-Header><Transaction-ID><Gate-ID><PacketCable-Err>.
- **<Gate-Delete-Ack-Response>**=<COPS-Common-Header><Handle><Report-Type><ClientSI-Header><Transaction-ID><Gate-ID>.
- **<Gate-Delete-Err-Response>**=<COPS-Common-Header><Handle><Report-Type><ClientSI-Header><Transaction-ID><Gate-ID><PacketCable-Err>
- CMTS-initiated messages (use COPS to GC but Radius to remote CMTS) which can also be GC-initiated (using COPS):
 - **<Gate-Open>**=<COPS-Common-Header><Handle><Report-Type><ClientSI-Header><Transaction-ID><Gate-ID>.
 - **<Gate-Close>**=<COPS-Common-Header><Handle><Report-Type><ClientSI-Header><Transaction-ID><Gate-ID><PacketCable-Reason>. Gate-Open allows the CMTS to inform the gate controller that gate resources have been committed. Gate-Close allows the CMTS to inform the GC that the gate has been deleted due to MTA interaction or inactivity. These messages provide a feedback path from CMTS to CMS in order to allow for accurate call-state management at the CMS element.

4.5.4.2.2 DQoS subobjects

4.5.4.2.2.1 *Transaction-ID* The Transaction-ID is used by the gate controller to match CMTS responses to previous GC requests.

Length = 8	S-Num = 1	S-Type = 1
2-octet Transaction-ID	Gate Command Type	

The following Gate Command Types are defined:

GATE-ALLOC	1
GATE-ALLOC-ACK	2
GATE-ALLOC-ERR	3
GATE-SET	4
GATE-SET-ACK	5
GATE-SET-ERR	6
GATE-INFO	7

GATE-INFO-ACK	8
GATE-INFO-ERR	9
GATE-DELETE	10
GATE-DELETE-ACK	11
GATE-DELETE-ERR	12
GATE-OPEN	13
GATE-CLOSE	14

4.5.4.2.2.2 *Subscriber-ID* This identifies the subscriber for this service request.

Length = 8	S-Num = 2	S-Type = 1
IPv4 address (for IP-v6, use S-Type = 2)		

4.5.4.2.2.3 *Gate-ID* This specifies the gate referenced in a command message or assigned by the CMTS in a response message.

Length = 8	S-Num = 3	S-Type = 1
Gate-ID		

4.5.4.2.2.4 *Activity count* In a Gate-Alloc message, this represents the maximum number of gates that can be allocated to a Subscriber-ID. In a Gate-Set-Ack or Gate-Alloc-Ack, it indicates the number of gates assigned to a single subscriber.

Length = 8	S-Num = 3	S-Type = 1
32-bit counter		

4.5.4.2.2.5 *Gate-Spec* The Gate-Spec object defines the filters that identify a flow, the direction of the flow, and its token bucket parameters for authorization (the authorization envelope). Filter values of zero (e.g., IP ID = 0) are wildcards (in this case the IP

indicated in the header of the IP packet parts of the flow can be anything). Two Gate-Spec objects must be used for a bidirectional session.

Length = 60		S-Num = 5	S-Type = 1
0:Downstream/1:Upstream	IP ID	Flags	Session-Class
Source IP address (32 bits)—0 for wildcard			
Destination IP address (32 bits)—0 for wildcard			
Source port—0 for wildcard		Destination port—0 for wildcard	
DiffServ DSCP			
Timer T1 value—maximum authorization to commit time (in ms)			
Timer T7 value—maximum duration of a single gate open state when a flow crosses two CMTS (in ms)			
Token Bucket Rate (r bytes/s)			
Token Bucket Size (b bytes/s)			
Peak Data Rate (p bytes/s)			
Minimum Policed Unit, (i.e., smallest IP packet in the stream (m octets))			
Maximum Packet Size (M bytes)			
Rate (R bytes/s)			
Slack Term, (i.e., acceptable delay on top of theoretical delay obtained for $R = r$ (s))			

The flag value 0x01 can be used to tell the CMTS to automatically commit the reserved resources without waiting for an MTA command.

4.5.4.2.2.6 Event-Generation-Info This element is included if the CMS wants the CMTS to generate accounting records.

Length = 44	S-Num = 7	S-Type = 4
Primary Record Keeping Server IP		
Primary Record Keeping Server Port	Flags	

Secondary Record Keeping Server IP		
Secondary Record Keeping Server Port	Flags	
Billing Correlation ID (24 bytes)		

4.5.4.2.2.7 Electronic surveillance parameters This enables the CMS to ask the CMTS to duplicate call-related events or even call media streams, and to send them to an interception device.

Length = 20	S-Num = 10	S-Type = 1
DF-IP-Address-For-CDC (IP address where call events should be sent)		
DF-Port-For-CDC	Flags: 1 = Send Events to CDC, 2 = Send Content to CCC	
DF-IP-Address-For-CCC (IP address where call content should be sent)		
DF-Port-For-CCC		

4.5.4.3 Sample call flow

4.5.4.3.1 Embedded MTA, no use of Gate-Open messages

In this first sample call flow (Figure 4.29), the embedded MTAs use layer 2 (DOCSIS) signaling for QoS reservations over the cable plant. The two CMSs use the DCS version of SIP to communicate between them and the NCS version of MGCP to control the MTAs.

CMS_O (originating) uses the Gate-Alloc message to retrieve a Gate-ID handle from the CMTS. Subsequently, it passes this Gate-ID in all GC commands. The gate is initially in ALLOCATED state.

Note the special use of SIP: the first INVITE does not cause the called phone to ring; instead, CMS_T (terminating) waits to receive an UPDATE message from CMS_O. This call flow makes sure that the destination phone does not ring if there are not enough resources end to end:

- The 183 Progress provisional response is sent from CMS_T to CMS_O to indicate that sufficient resources have been reserved on the terminating site (and that optionally one-way backbone provisioning from the terminating side to the originating side if RSVP is used in the backbone).
- The UPDATE message from CMS_O indicates to CMS_T that sufficient resources have been reserved on the originating side (and that optionally one-way backbone provisioning from the originating side to the terminating side if RSVP is used in the backbone).

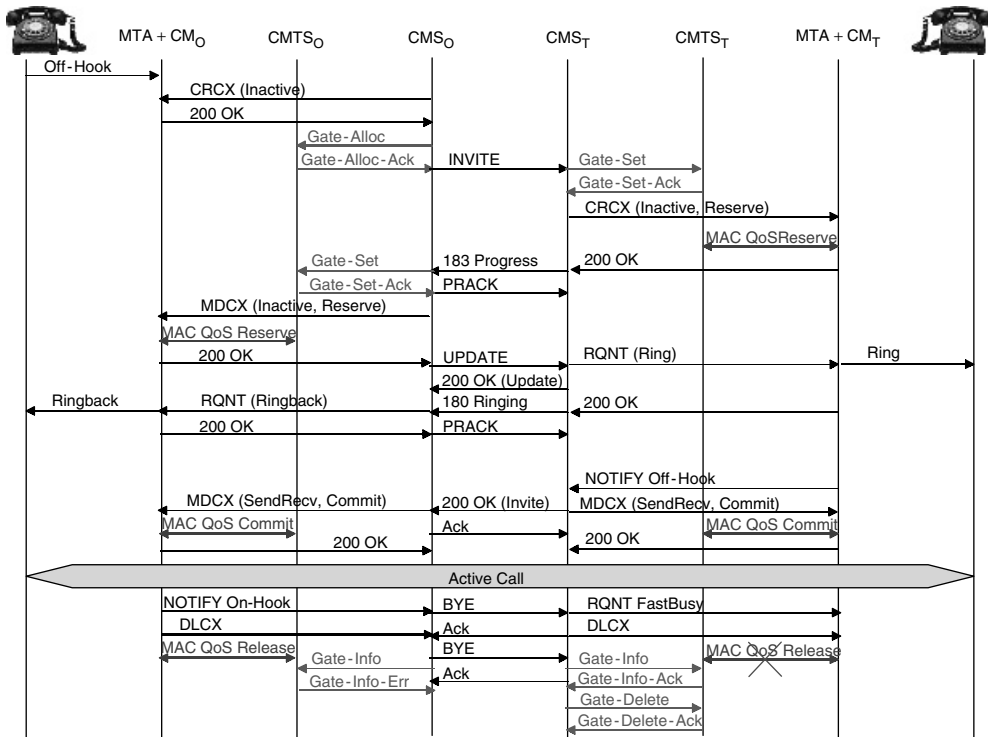


Figure 4.29 EMTA scenario.

In order to secure resources on the cable plant, the CMS first gives an Authorization to the gate indicating certain filters and traffic properties in the Gate-Set message. The gate transitions to AUTHORIZED state. The CMS then indicates to the MTA (via the call control protocol) that the MTA should reserve these resources. MGCP NCS is extended to support the indication of the target Gate-ID (L: dq-gi:<gate-ID>). Using the information provided by the call control, the MTA then reserves the required resources via the MAC-layer protocol and confirms successful reservation via the call control protocol (MGCP 200 OK). If MAC-level reservation takes time, the MTA can send a provisional '100 277 PENDING' response. If the reservation is successful, the gate transitions to RESERVED state.

Once the resources are reserved, the MTA can be sure that a command to commit these resources and begin to use them will not fail. The reserved gate state is soft state, so the reservation should be refreshed if not used quickly. After a commit command from the MTA, the gate is in COMMITTED state.

The CMS also gets involved in checking that the MTA closes the gate properly by releasing resources at the end of the call. In case the MTA fails to do so (as shown for MTA_T), then the CMS Gate-Info command will succeed because the gate is still in place. If this occurs, the CMS forces the CMTS to close the gate with a Gate-Delete command.

4.5.4.3.2 Stand-alone MTA with RSVP+

If the MTA and the cable modem are physically separate, the MTA cannot use the DOCSIS MAC layer to request resources: so a layer 4 protocol, based on RSVP, is used, which triggers a MAC layer QoS reservation initiated by the CMTS. A sample call flow is described in Figure 4.30.

On the cable plant, full duplex bandwidth is reserved after the first PATH/RESV exchange, because of the extensions of RSVP+. However, the CMTS still sends a PATH toward the MTA because the normal RSVP semantics apply between the cable modem and the MTA; therefore, each PATH/RESV exchange between the CMTS and the MTA only reserve one-way resources in the segment between the cable modem and the MTA.

4.6 Improving QoS in the best effort class

While it is easy to understand that a network provider will try to offer low loss rates and low latency to its premium customers, one might wonder what he can do for its best effort customers. This section describes how to improve fairness among best effort customers (each customer must have an equal opportunity to use the capacity of the best effort class) and how to prevent some issues that may occur with TCP.

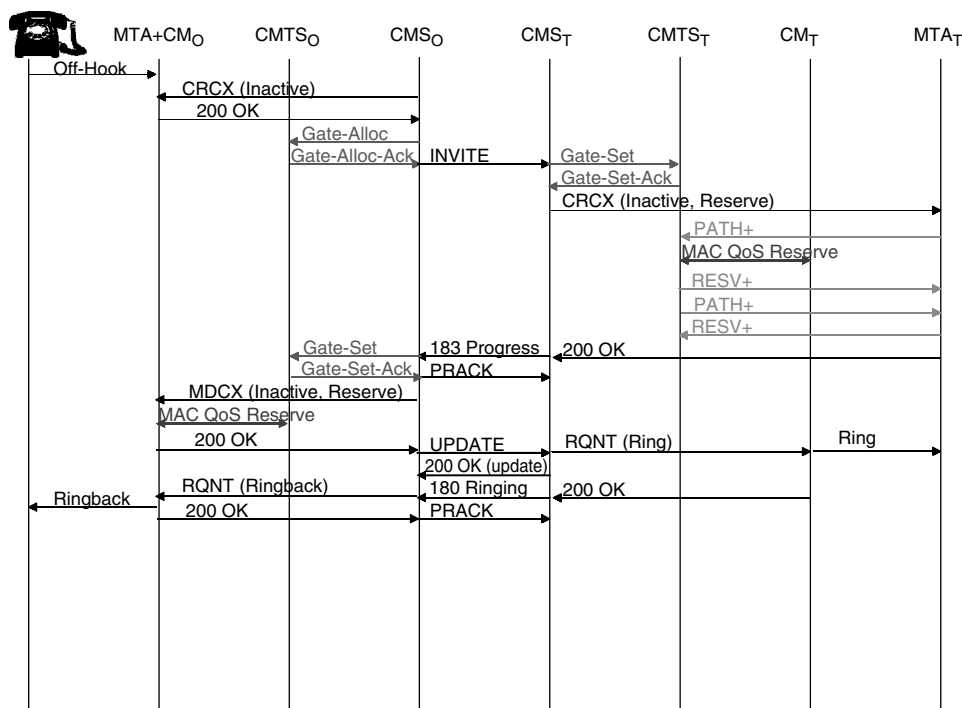


Figure 4.30 Stand-alone MTA scenario with RSVP+.

4.6.1 Issues with UDP traffic

UDP traffic has no standard rate-limiting feature in case of congestion. Properly written applications should be able to detect network congestion and react by backing off the rate of UDP traffic (e.g., all applications using RTP/RTCP can detect congestion when RTCP reports increasing packet loss and higher latency). But, on the best effort class where everyone pays a flat fee, it is very tempting to create a UDP application which has a high redundancy scheme and which reacts to congestion not by reducing its bitrate, but by *increasing* it with more redundant and forward error correction packets!

Unfortunately, it will work, because most best effort queues are handled in FIFO mode and packets are dropped when the queue overflows. Statistically, the packets that remain in the queue without being dropped represent a percentage of the offered traffic: so the more you offer traffic to the node, the more packets you get through the node, as shown in Figure 4.31. Such behavior will cause “honest” users to be gradually excluded from the network, while greedy users will enjoy it all.

4.6.2 Issues with TCP traffic

4.6.2.1 TCP congestion avoidance and spontaneous synchronization

All modern TCP stacks implement congestion control algorithms developed by Van Jacobson, as specified in RFC 1122 (and updated in RFC 2001). These algorithms interpret packet loss as an indication that the network is congested and react by slowing down the rate at which TCP injects traffic in the backbone.

The Van Jacobson and Karels congestion control algorithm is one of the most popular. It works by defining, in addition to the usual receiver window size (the maximum number

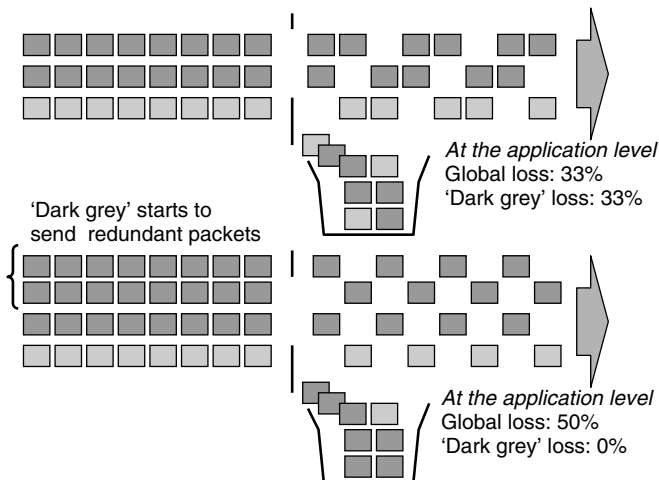


Figure 4.31 Greedy application pushing out other sessions.

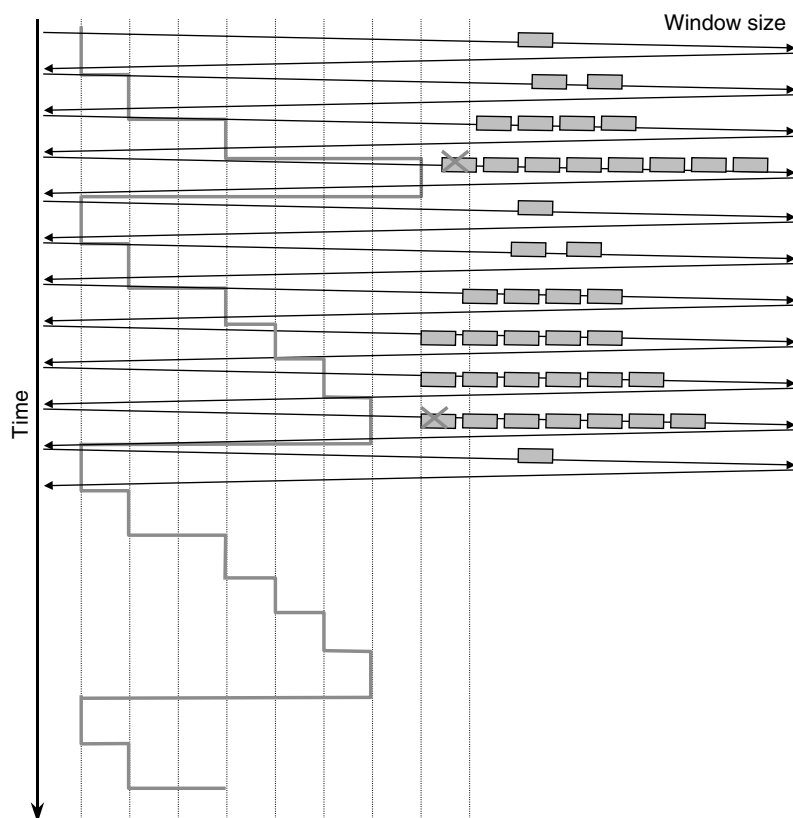


Figure 4.32 TCP window size backs off when packet loss is detected.

of bytes that the receiver is prepared to receive and not yet acknowledged), a congestion window. The actual window size used during transmission is the minimum of the receiver window size and the congestion window. Figure 4.32 shows the slow-start mechanism of this algorithm, as well as the TCP stack reaction to packet loss. Congestion window size begins with the size of one segment (512 or 536 bytes unless the other end requests a different size) and gets one segment larger for each acknowledgement received without loss. This exponential growth is characteristic of slow-start mode. If there is a timeout while waiting for an ACK or if there are duplicate ACKs sent by the receiver,³ the sender deduces that there is some congestion and sets a congestion avoidance threshold to one-half of current window size. Then the sender sets the window size to one segment and increases it in slow-start mode. Once the window size gets larger than the congestion avoidance threshold, the sender increases the window size by $(\text{segment_size}/\text{window_size})$ segments for each ACK, and the window size increases linearly with time.

³ Modern TCP stacks (like the 4.3 BSD ‘Reno’ stack) implement a specific fast retransmit/fast recovery procedure instead of going into slow start immediately when the receiver sends duplicate ACKs (duplicate ACKs are sent when the receiver receives out-of-sequence packets).

When a FIFO queue overflows, many active TCP sessions are very likely to lose an IP packet *Simultaneously*. These sessions will back off (Figure 4.32) and hopefully the congestion will disappear. Then all these TCP sessions will see that they no longer lose packets and will increase their window size little by little . . . and after a while congestion is back, packets are dropped, and all sessions back off again. This leads to an oscillation of TCP traffic caused by synchronization of VJ algorithms after queue overflow.

4.6.3 Using ‘intelligent’ packet discard

In addition to the synchronization problem described above, TCP implementations that incorporate congestion avoidance are network-friendly, while UDP programs or hacked TCP stacks will continue to flood congested links. TCP stacks with congestion avoidance will rapidly back off and progressively let the aggressive traffic use the complete capacity of congested links. Fortunately, it is possible to improve the behavior of the network by discarding packets ‘intelligently’.

Random early detection (RED) allows the backlog in routers to be kept as low as possible, while avoiding TCP synchronization. A RED router starts dropping packets at random before the FIFO buffer actually overflows, as shown in Figure 4.33. The session that experiences packet loss should start to reduce its bitrate. Little by little more sessions will experience some random packet loss and reduce the aggregate bitrate smoothly before real congestion occurs. Without this policy, all sessions will experience some synchronized packet loss and reduce their bitrate simultaneously when the buffer overflows, which may cause undesired synchronization and possible oscillations.

With weighted RED (WRED), it is possible to go further and decide to increase packet loss for a class of sessions. **WRED** can be used to do some basic prioritization (e.g., traffic in class ‘premium’ would experience packet loss only after class ‘medium’), but more interestingly it enforces fairness. In Figure 4.31, it is relatively easy to identify the greedy ‘blue’ stream as one getting more bandwidth than it should (e.g., by analyzing lost packets). The normal behavior of a router is to lose packets of a given session proportionally to the traffic offered by the session. We have seen that this actually encourages the use of aggressive redundancy schemes. WRED introduces some nonlinearity (e.g., in Figure 4.31 the router could decide to drop 80% of the ‘blue’ packets).

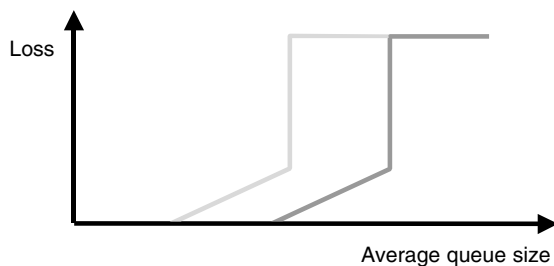


Figure 4.33 Probability of packet loss introduced by WRED according to transmission queue size.

4.7 Issues with slow links

4.7.1 Queuing

Slow links are especially challenging for delay-sensitive applications. For instance, a 1,500-octet packet (a common size since it is the MTU over Ethernet) needs over 400 ms to be transmitted over a 28.8-kbit/s PPP link. This would not be an issue if there was only the delay-sensitive application on the link, because this application could use smaller packets. Unfortunately, the line is often shared with other applications using larger packets, such as a web browser, and urgent packets will be delayed if they are queued behind large packets (see Figure 4.34).

4.7.2 Overhead

Another issue is that the overhead of the IP suite is quite high (e.g., in each packet of a telephony or video application, IP uses 20 bytes, UDP uses 8 bytes, and RTP uses 12 bytes). The link-layer overhead is also significant: PPP and HDLC take an additional 4 bytes per packet on a modem link, a frame relay link will add 9 bytes per packet. Over ATM the overhead depends on packet size: for an 80-byte packet (40 ms G.729) it adds 25 bytes (this overhead is called the ‘cell tax’).

To see how bad it is, consider what happens with IP phones using the G.723.1 coder. The codec has an audio frame length of 30 ms. In 6.4-kbit/s mode (MP-MLQ), each frame is 24 octets long. When the IP phone stacks only one frame per packet, each IP packet

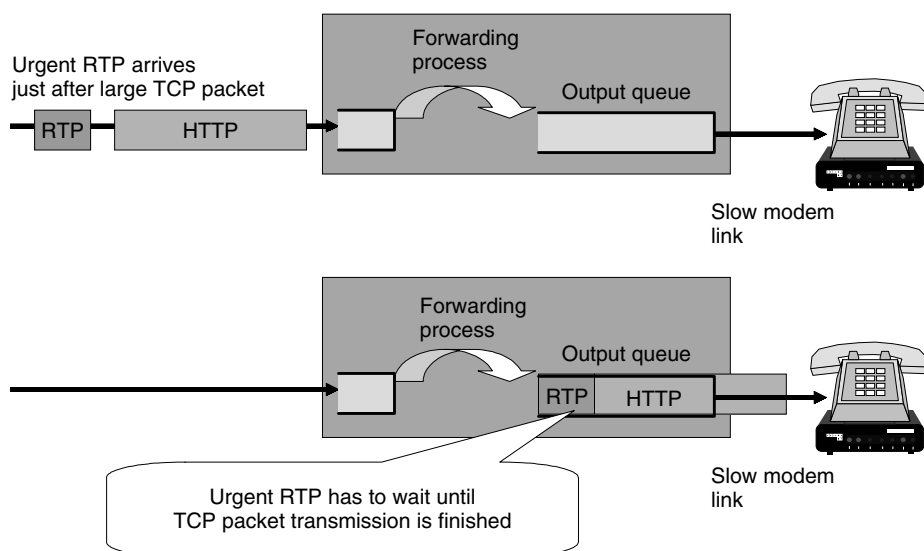


Figure 4.34 Latency caused by large unfragmented packets.

is $24 + 40$ bytes long, and PPP + HDLC will make it 68 bytes long on the PPP link. A packet is sent every 30 ms, and the resulting bitrate is over 18 kbits/s. So only one-third of the bitrate is actual audio information from the G.732.1 codec!

In order to leave some room for data and video over 28.8-bit/s modems, applications such as Microsoft NetMeeting® have to stack several frames per packet. A common choice is to group 4 G.723.1 frames in one packet. This reduces the bitrate to 7.7 kbit/s, but the packetization delay becomes $4 * 30 = 120$ ms (128 ms including look-ahead). If we assume the best case situation where this packet can be sent immediately, it will need 30 ms to get through the 28.8-bit/s modem link. This results in network delay and again 30 ms at the egress modem ... Unfortunately, this sets conversational delay to a level that is unacceptable for most non-hobby users.

4.7.3 Overhead compression

The overhead issue can be solved by using header compression. Header compression is based on a simple idea: since about half of the overhead of an IP/UDP/RTP packet is constant for a given stream (e.g., the source and destination IP addresses and ports are constant), it is therefore possible on a point-to-point link to negotiate a shorter ‘index’ for those constant values when the stream is set up. Moreover, some fields increment by constant values and can be reconstructed at the receiving end if a proper per-stream state is maintained.

Such an IP header compression algorithm has been standardized by M. Engan, S. Casner, and C. Bormann in RFC 2509 ‘IP Header Compression over PPP’ (February 1999), and extended by S. Casner and V. Jacobson for applications using RTP in RFC 2508 ‘Compressing IP/UDP/RTP Headers for Low-speed Serial Links’ (February 1999). This method is also called **compressed RTP (CRTP)** and can be negotiated as part of the serial link protocol negotiation (e.g., as PPP options). More recently there was some work to improve the tolerance of header compression to packet loss: an improved robust header compression (ROHC) has been defined in RFC 3095 mainly for 3G networks. Most of the time, the IP/UDP/RTP headers are compressed to just 2–4 bytes (2 bytes if the UDP checksum is not used) instead of 40 for a full IP/UDP/RTP header. While compression can be done end to end for RTP alone (preserving IP-addressing information), a compression protocol for all three protocols (IP/UDP/RTP) can only work on a link-per-link basis (IP-addressing information is not available in compressed packets).

With CRTP, the sending host replaces the large header with a small index to a session context and differential values for variable fields. The receiving host reverses the operation. In each IPv4 header of a packet stream belonging to a given context (Figure 4.35), only packet length, packet ID, and header checksum will change. The packet length indication is redundant with framing of the link layer and CRC is already provided by link layer: both can be reconstructed. Packet ID changes by one or a small increment for each packet, so only the increment value needs to be in the context. For UDP, the length field is redundant. For RTP the SSRC value is constant for a context and the sequence number is incremented by one unless packets are disordered. The RTP timestamp is incremented by multiples of the frame size for audio (e.g., 2 if there are 2 codec frames in each RTP

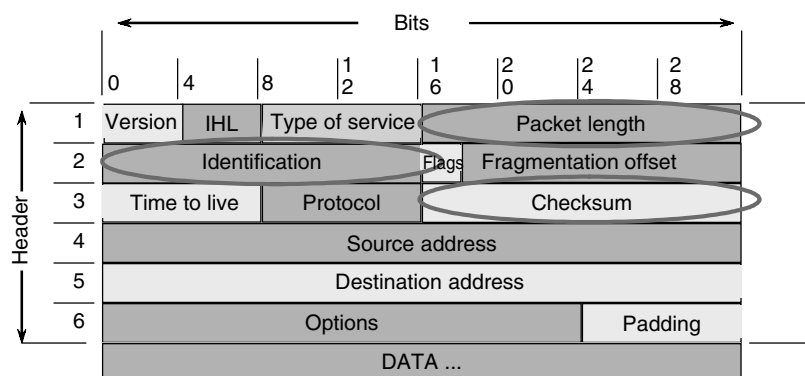


Figure 4.35 Very few fields change between IP packet parts of the same flow.

packet), and for video it will change for each first packet of a video frame description, then remain constant for RTP packets containing the rest of the video frame description. Only the increment size needs to be in the context. The RTP M bit will change each time speech commences but would take too much overhead if compressed (e.g., there would be a need to resynchronize context at each change).

In each CRTP packet, the context is identified by a session **context identifier (CID)** of 1 or 2 octets and a 4-bit serial number is added to detect packet loss on the link. Each host can generate multiple RTP sessions and, therefore, a session context must be associated with each group (source IP address, source port, destination IP address, destination port, and RTP SSRC field). The link layer (e.g., PPP) also needs to convey an indication of the format of the packet (no compression: `FULL_HEADER`; only IP + UDP header compressed in 6 bytes or just 2 if UDP checksum is disabled: `COMPRESSED_UDP`; IP + UDP + RTP headers compressed, generally just 2 bytes in total: `COMPRESSED_RTP`; and packet indicating that a context identifier is invalid: `CONTEXT_STATE`). A bit mask indicates which fields differ from their prediction and, therefore, conveyed in differential form in the compressed packet. Any packet that cannot be predicted using one of the compressed formats (unusual change) will be sent in `FULL_HEADER` format. The context contains:

- Full IP/RTP/UDP headers, initialized with values found in the `FULL_HEADER` packet.
- Increment values of IP packet ID, RTP timestamp, initialized by comparing the values of the first two transmitted stream packets (the increment value for the RTP sequence number is implicitly set to one).
- Current 4-bit sequence number.

The CRTP compression scheme is optimized for duplex links and uses a backward `CONTEXT_STATE` packet to signal a loss of synchronization between coder and decoder. All transmitted packets are lost as long as CRTP contexts are not resynchronized.

The scheme works best with RTP packets, but since the compression scheme is able to reconstruct the IP packet exactly (lossless), it will not cause any harm to a UDP packet

improperly recognized as an RTP packet. A pair of routers using CRTP can therefore use relatively simple pattern matching to recognize potential RTP packets, even if there is a small probability that some non-RTP UDP packets will match by error. The routers need to keep a cache of contexts (which get invalidated too frequently) in order to avoid using RTP compression for these data streams.

It is interesting to note that IP and TCP header compression has also been introduced in the UMTS standard for transport of IP packets over the UTRAN (Universal Terrestrial Radio Access Network) radio link. The PDCP (Packet Data Convergence Protocol) layer compresses IP headers using RFC 2507 ('IP Header Compression') and TCP/IP headers using RFC 1144 ('Compressing TCP/IP headers for Low-speed Serial Links'). Unfortunately, RTP compression has not yet been defined for PDCP, but PDCP is extensible; therefore, it is likely that RTP compression will also be supported one day, especially if mixed-mode VoIP/WiFi and UMTS 3G terminals become more common.

4.7.4 Packet fragmentation, prioritization over serial links

The queuing issue can be solved by allowing real-time packets to pre-empt non-real time packets. Two techniques can be used for pre-empting:

- Using the **multilink PPP (PPP-MP)** protocol (RFC 1990), also called **link fragmentation and interleaving (LFI)**. The original design of multilink PPP was to allow the bundling of several physical or logical links into a single virtual link. Compared with plain PPP, PPP-MP provides additional means to sequence and correlate fragments arriving from several links. Since multilink PPP is able to fragment packets, it can be used to stop transmission of a long packet, send an urgent packet, and resume transmission of the long packet. Figure 4.36 shows the multilink PPP bitstream format, when the multilink short-sequence number fragment format is used (only 12 bits instead of 24, and PPP compression of address, control, and most significant PID bytes assumed active). This simple solution has the advantage of being compatible with existing PPP-MP implementations, but cannot be used for more than one pre-emption level. This is because the PPP-MP specification requires the sequence numbers of fragments of a packet to be monotonically increasing and contiguous.⁴ Therefore the only way to send an urgent packet between fragments is to send it without the PPP-MP header,

⁴ Extract from RFC 1990: "On each member link in a bundle, the sender **MUST** transmit fragments with strictly increasing sequence numbers (modulo the size of the sequence space). This requirement supports a strategy for the receiver to detect lost fragments based on comparing sequence numbers. The sequence number is not reset upon each new PPP packet, and a sequence number is consumed even for those fragments which contain an entire PPP packet, i.e., one in which both the (B)eginning and (E)nding bits are set."

An implementation *must* set the sequence number of the first fragment transmitted on a newly constructed bundle to zero. (Joining a secondary link to an existing bundle is invisible to the protocol, and an implementation *must not* reset the sequence number space in this situation).

The receiver keeps track of the incoming sequence numbers on each link in a bundle and maintains the current minimum of the most recently received sequence number over all the member

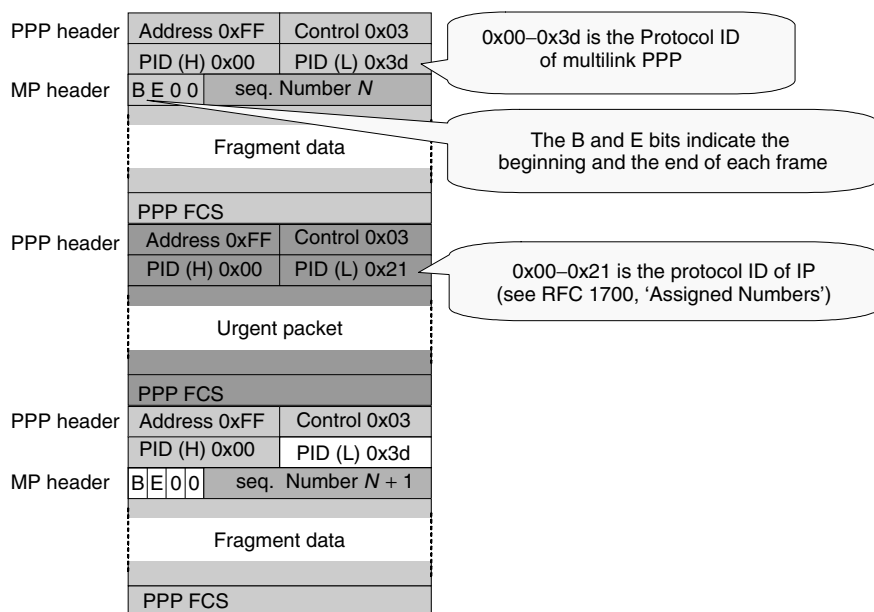


Figure 4.36 Insertion of an urgent packet in the PPP multiplex. Another limitation to using PPP for QoS purposes is that it cannot be used on bundles of multiple serial links, as there is no sequencing information for inserted priority packets, which therefore would not be kept in sequence.

which is allowed by RFC 1990. An urgent packet interleaved between two fragments of a long packet are shown in Figure 4.36 (HDLC 0×7E flags at the beginning and end of each frame are not represented).

- RFC 2686, **'The Multi-class Extension to Multi-link PPP' (MCML)**, addresses this limitation and makes it possible to manage a priority class over multiple serial links. Delay-sensitive traffic ('class 0') is also fragmented just like pre-empted non-delay-sensitive packets ('class 1'). All sequence numbers are managed per class, resolving the ambiguities that prevented doing the same on multilink PPP.

Both techniques should be used over links with relatively low packet loss and low round trip delays, which is usually the case with access links. Some enhancements of ML-PPP have been proposed to address these limitations, but they are not widely implemented. ML-PPP and MCML are intended to be used on links with typically less than 2 Mbps of bandwidth; for higher speed links the gain on delay is small and there may be excessive impact on router CPU usage, especially for a service provider aggregation router if it has to manage several customer serial links.

links in the bundle. The receiver detects the end of a packet when it receives a fragment bearing the (E)nding bit. Reassembly of the packet is complete if all sequence numbers up to that fragment have been received.

4.8 Conclusion

Quality of service over packet networks is not a trivial issue, regardless of the technology. There has been a lot of theoretical work lately, and a consensus on the deployment strategy needed to build a large, QoS-aware, IP backbone is slowly emerging.

The main factor driving the latency aspect of quality of service is the length of the data packet divided by the transmission link speed. Small packets are preferable because they tend to introduce less latency: this is the background behind the advantage of using native ATM for small to medium links, typically below 1 Mbit/s. However, small packets also come with a big disadvantage: they create a lot of overhead (sometimes called the 'cell tax' in ATM); therefore, if the bandwidth is large enough, large packets become preferable because they are more efficient.

From this point of view, IP is clearly the technology of the future. In most IP backbones, bandwidth is already more than adequate to reduce packet size-related latency to something insignificant. The same is true within corporate LANs, with 10/100/1,000 Mbit/s Ethernet links deployed massively. The last problematic bottleneck in many networks is the local loop. Non-broadband techniques (ISDN, modem) require fragmentation techniques to reduce link latency. Today, first-generation broadband techniques, such as cable or ADSL, offer barely the bandwidth necessary to transmit IP packets without fragmentation in the upstream direction (about 512 kbit/s to 1 Mbit/s maximum) and are often restricted further by service providers. Second generation broadband techniques will offer a more comfortable bandwidth: ADSL2 will offer 10 Mbit/s downstream and 1.2 Mbit/s upstream,⁵ thus offering better support for IP-level quality of service over multiple parallel transport channels.

ADSL2+ doubles the downstream bandwidth figure by doubling the occupied frequency spectrum. For non-residential use requiring symmetric bandwidth, SDSL is already more than adequate for transporting low-latency IP packets (Table 4.3).

The deployment of xDSL technologies is a very big success, but they will probably increasingly compete with Ethernet as a first-mile technology, over a fiber to the building infrastructure. These techniques are promoted by the Ethernet First Mile Association (EFMA) and the Metro Ethernet Forum (MEF). Some cities already have a significant coverage of fiber to the building infrastructure, e.g., Milan—Figure 4.37—with a 100,000 homes covered by a combination of fiber provision in the building and Ethernet to the home. Optical fiber in the local loop will remove the last bottleneck for end-to-end quality of service at 100 Mbps and over.

In this chapter, we focused on IPv4. There is sometimes the perception that IP will be able to provide QoS only once IPv6 is deployed. This is clearly wrong. All important QoS mechanisms present in IPv6 are also available in IPv4, and IPv4, given the current state of access loop technology, creates much less overhead. With DiffServ and the ever-improving layer 2 technologies, IPv4 is now ready for widespread deployment of applications requiring quality of service, like VoIP. The last area where significant

⁵ On average ADSL2 offers 50 kbit/s more bandwidth than ADSL, the 200-kbit increase of the upstream channel is obtained by taking the bandwidth previously reserved for baseband voice ... assuming voice over packet will be used!

Table 4.3 DSL performance table

Technique			Downstream (Mbps)	Upstream (Mbps)	Max bandwidth distance (km)
Assymetric	ADSL with splitter	ADSL/RADSL	<8.2	<1	2.5
		ADSL2 (G.992.3/G.992.4)	<12	<1.2	3
		ADSL2+ (G.992.5)	<26	<1.2	1.5
		VDSL	<26	<9	1
	Splitterless ADSL	UDSL/UADSL/RDSL/CDSL/ DSLlite/ADSLlite/Glite	<1.5	<0.512	≈2
Symmetric	SDSL	1 twisted pair: SDSL/SPDSL/HDSL(G.991.1)/ HDSL2 (ANSI T1418)	<2.3		3
		SHDSL (G.991.2) in single-pair mode	<2.3		4
		Ethernet			
	Ethernet	IEEE P.802. ah EFM copper (EFMC) Metro Ethernet Forum E-line, user network interface	10		0.75
		IEEE P.802.ah EFM single-mode fiber (EFMF)	100–1,000 ps		10
		IEEE P.802.ah EFMP passive optical network-shared fiber with optical splitters	1,000		20



Figure 4.37 Fiber deployment in Milan (FastWeb).

improvements are still desirable is access control, in order to secure IP backbones against DoS attacks, which become more threatening as the bandwidth of end-user access links increases. This seems to be where IntServ may have a much more pre-eminent role in the future.

4.9 References

- A.K. Parekh and R.G. Gallager. A generalized processor sharing approach to flow control in integrated services networks, Part I. *IEEE/ACM Transactions on Networking*, **1**(3), pp. 344–357 (June 1993).
- A.K. Parekh and R.G. Gallager. A generalized processor sharing approach to flow control in integrated services networks, the multiple node case. *IEEE/ACM Transactions on Networking*, **2**(2), pp. 137–150 (April 1994).
- S.J. Golestani. *A Self-clocked Fair Queuing Scheme for Broadband Applications*. Bellcore, ATT Research Labs.
- N.R. Figueira and J. Pasquale. An upper bound on delay for the virtual clock service discipline. *IEEE/ACM Transactions on Networking*, **3**(4) (August 1995).
- V. Jacobson. Congestion avoidance and control. *Proceedings of ACM SIGCOMM '88*, pp. 314–329 (August 1988).
- S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, **1**(4), pp. 397–413 (August 1993).
- [RFC 1812] Requirements for IPv4 Routers (F. Baker).
- [RFC 2205] Version 1 Functional Specification of RSVP.
- [RFC 2211] Specification of the Controlled-load Network Element Service.
- [RFC 2212] Specification of the Guaranteed Quality of Service.
- [RFC 2474] Definition of the Differentiated Services Field in the IPv4 and IPv6 Headers.
- [RFC 2475] An Architecture for Differentiated Services.
- [RFC 1812] Requirements for IPv4 Routers.
- [RFC 2508] Compressing IP/UDP/RTP Headers for Low Speed Serial Links (February 1999: S. Casner, V. Jacobson).
- [RFC 1990] The PPP Multilink Protocol (MP) (August 1996: K. Sklower, B. Lloyd, G. McGregor, D. Carr, T. Coradetti).
- [RFC 1661] The Point-to-Point Protocol (PPP) (July 1994: W. Simpson, editor).
- [RFC 1662] PPP in HDLC-like Framing (July 1994: W. Simpson, editor).

5

Network Dimensioning

5.1 Simple compressed voice flow model

5.1.1 Model of popular voice coders

Most recent vocoders implement a voice activity detection algorithm, which uses significantly less bandwidth in silence periods than in active speech periods.

For most coders the activity bitrate is constant, and we call it M . In silence periods very basic coders will have a null bitrate, but coders such as G.723.1 or G.729 actually send some information to describe the background noise level and other parameters. We will call the bitrate during silence periods m .

m and M values for popular coders are shown in Table 5.1 for IP packets containing about 30 ms of speech, including all IP overhead.

In Table 5.1, one may wonder why G.723.1 in its 6.4-kbit/s mode has an M value of 16 kbit/s. This is because 6.4 kbits per second is the bitrate at the output of the coder and does not include transport overheads, such as RTP/UDP/IP headers. We want to properly dimension IP links, and therefore we must take into account such overheads. Table 5.2

Table 5.1 M and m values for common voice coders, when using 30-ms IP packets

Codec	Frames/IP packet	M (kbit/s)	m (kbit/s)
G.723.1 (5.3 kbit/s)	1	16	11.73
G.723.1 (6.4 kbit/s)	1	17.07	11.73
G.729	3	18.67	13.87
Lucent SX7003P	2	20.27	13.87

Table 5.2 Detailed calculation of the average bitrate, with and without RTP compression

IPv4 overhead (octets)	20						
UDP overhead (octets)	8						
RTP overhead (octets)	12						
Activity rate (%)	35						
		G.723.1			SX7003P		G.729.A
		<i>G (5.3 kbit/s)</i>	<i>(6.4 kbit/s)</i>	<i>(silence)</i>	<i>4.8 kbit/s</i>	<i>(silence)</i>	<i>8 kbit/s</i>
Frame size (data octets)		20	24	4	18	6	10
Frame duration (ms)		30	30	30	15	15	10
Frames per IP packet		1	1	1	2	2	3
Bitrate without overhead (kbit/s)		5.33	6.40	1.07	9.60	3.20	8.00
Octets per IP packet		20	24	4	36	12	30
Overhead IPv4+UDP+RTP (octets)		40	40	40	40	40	40
Bitrate with overhead (kbit/s)		16	17.07	11.73	20.27	13.87	18.67
Average bitrate (with activity rate)		13.23	13.60		16.11		15.55
Overhead cRTP (octets)		2	2	2	2	2	2
Bitrate with overhead (kbit/s)		5.87	6.93	1.60	10.13	3.73	8.53
Average bitrate (with activity rate)		9.68	10.05		12.56		12.00

Table 5.3 IP bitrate calculation formulas

	Coder activity	(silence)
Frame size (data octets)	FS_a	FS_s
Frame duration (ms)	FD_a	FD_s
Frames per IP packet	n_a	n_s
Bitrate without overhead (kbit/s)	$=FS_a * n_a / FD_a$	$=FS_s * n_s / FD_s$
Octets per IP packet	$=FS_a * n_a = \text{Payload}_a$	$=FS_s * n_s = \text{Payload}_s$
Overhead IPv4+UDP+RTP (octets)	$20 + 8 + 12 = Ov$	$20 + 8 + 12 = Ov$
Bitrate with overhead (kbit/s)	$=(\text{Payload}_a + Ov) * 8 / (n_a * FD_a) = M$	$=(\text{Payload}_s + Ov) * 8 / (n_s * FD_s) = m$
Average bitrate (with activity rate)	$=M * \text{act_rate} + m * (1 - \text{act_rate})$	

shows the spreadsheet that was used to calculate the results listed in Table 5.1. It is also interesting to read the bitrate results when the IP/UDP/RTP headers are compressed using cRTP to just 2 octets. The template for calculating such values is straightforward (Table 5.3).

Additional input is needed to calculate the actual bitrate at the physical layer, because IP packets themselves are encapsulated in Ethernet (+26 octets, as shown in Figure 5.1), HDLC (+7 octets), frame relay (+9 octets), PPP frames (+7 octets), or ATM cells using AAL5 (5 header octets for 48 payload octets)!

Figure 5.2 shows network capture of an RTP packet over Ethernet (the Ethernet preamble length and FCS fields do not appear), containing a single frame of G.723.1-encoded voice.

In order to reduce this enormous overhead, multiple compressed voice frames are frequently concatenated in each packet. Of course, packetization delay increases accordingly (number of frames in the packet times the frame delay), so there is a limit to the number of frames that can be accumulated per packet. Mouth-to-ear delay¹ must preserve good interactivity and must be acceptable, given the level of echo cancelation in terminal devices. For low-quality PC-to-phone applications, some vendors accumulate up to 90 ms of speech in each packet (3 G.723.1 frames), but professional use of VoIP should stay in the 20 to 50-ms range.

Given the behavior of most voice coders, we model the one-way network bitrate during a voice conversation by a two-level function characterized by the suite of active speech intervals $T_{\text{active}}(i)$ and the suite of silence intervals $T_{\text{idle}}(i)$, as shown in Figure 5.3. Activity rate a is defined as the limit when i gets infinite of $\sum(T_{\text{active}}) / \sum(T_{\text{idle}} + T_{\text{active}})$. A good average value is usually 0.35, but in order to be on the safe side we take $a = 0.5$ in most of our calculations (many VAD detectors remain in active state some time after the actual active period is over, augmenting the T_{active} period).

¹ For more details on acceptable mouth-to-ear delays see Chapter 3 on voice quality.

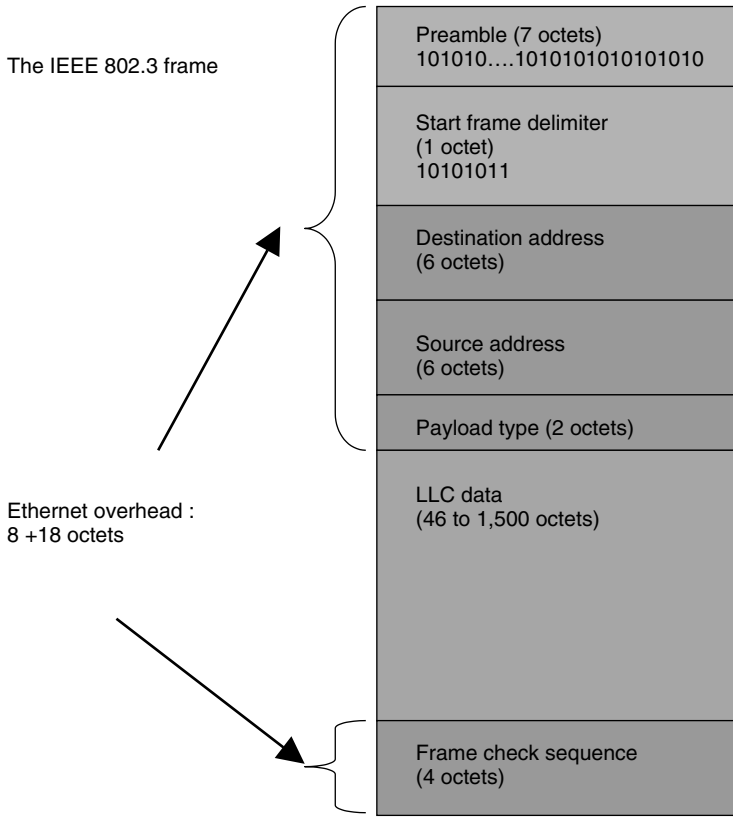


Figure 5.1 Overhead introduced by Ethernet.

5.1.2 Model for N simultaneous conversations using the same coder

When there are N simultaneous uncorrelated conversations on the same link, they will rarely be active simultaneously and, therefore, the required bandwidth will be less than $N \cdot M$.

When N gets very large it is intuitively obvious that the required bandwidth will be N times the average bitrate of the coder: $N \cdot [aM + (1 - a)m]$. But for small values of N some calculations are needed.

The probability of I conversations being simultaneously active among N can be expressed as:

$$P(I) = C_N^I \cdot a^I \cdot (1 - a)^{N-I} = \frac{N!}{I!(N - I)!} \cdot a^I \cdot (1 - a)^{N-I}$$

(to check that the sum is 1, note the natural development of $[a + (1 - a)]^N$). The results for 30 sessions are shown in Figure 5.4.

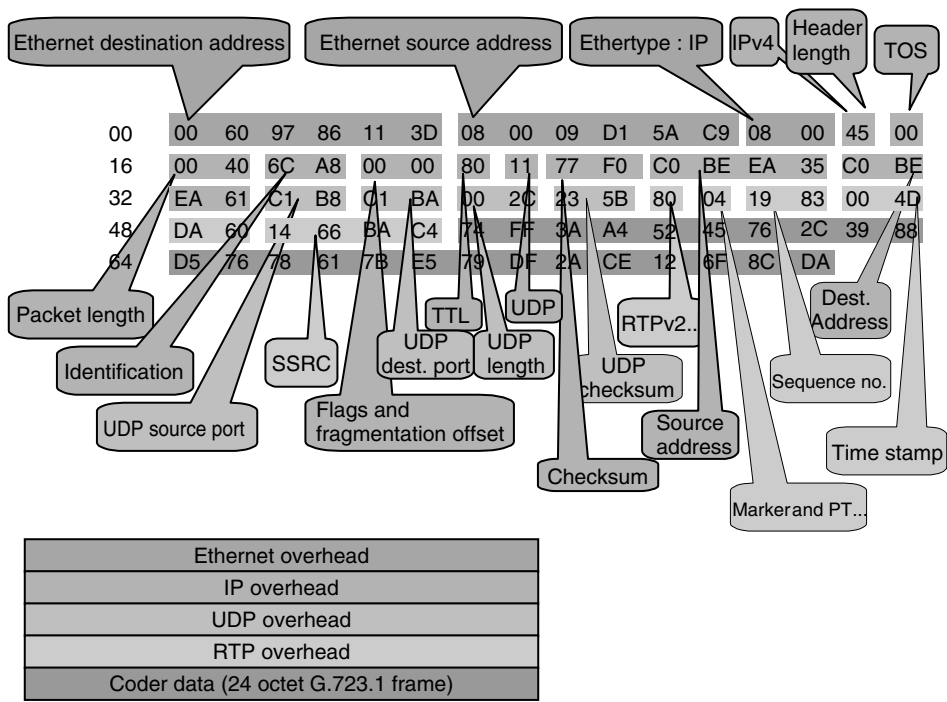


Figure 5.2 Try to find voice in this sea of overheads!.

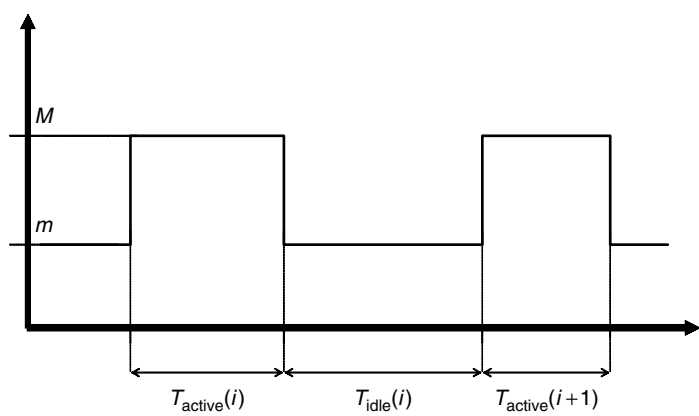


Figure 5.3 Simple on-off voice model.

The average one-way bitrate on a link with N simultaneous conversations is therefore:

$$Average_bitrate = \sum_{I=0}^N P(I) * [IM + (N - I)m]$$

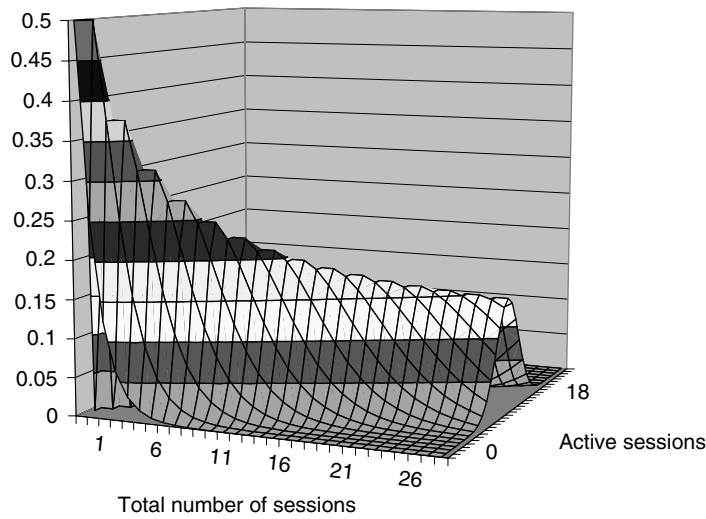


Figure 5.4 Probability of having I active sessions among N (for this illustration we took $a = 0.5$).

which can be simplified noticing that:

$$\begin{aligned} \sum_{I=0}^N \frac{N!}{I!(N-I)!} * I x^I * b^{N-I} \\ = x \sum_{I=1}^N \frac{N!}{I!(N-I)!} * I x^{I-1} * b^{N-I} = x \left(\frac{\partial (x+b)^N}{\partial x} \right) = x N (x+b)^{N-1} \end{aligned}$$

and therefore by substituting $x = a$ and $b = (1 - a)$:

$$\sum_{I=0}^N I * P(I) = aN \quad (A)$$

which simplifies our average bitrate expression into:

$$\text{Average_bitrate} = MaN + Nm - maN = N(Ma + m(1 - a))$$

which is the sum of average rates and was intuitively obvious. It can be shown, for such a probability law, that the standard deviation of the number of active session data streams is $\sigma = \sqrt{Na(1-a)}$, this value measures the amount of dispersion around the average.

For large values of N , the probability $P(I)$ can be approximated as a continuous function, and the Moivre–Laplace theorem shows that:

$$P(I) \rightarrow \frac{1}{\sqrt{2\pi}\sigma} * e^{\frac{-(I-Na)^2}{2\sigma^2}}$$

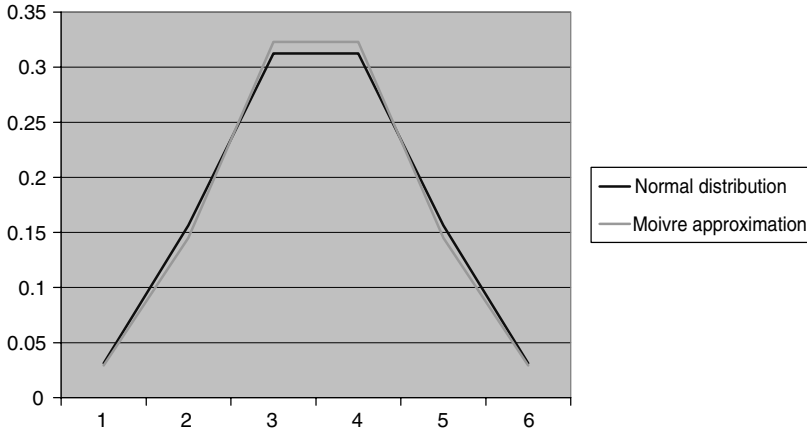


Figure 5.5 Exact probability versus Moivre approximation for $N = 5$.

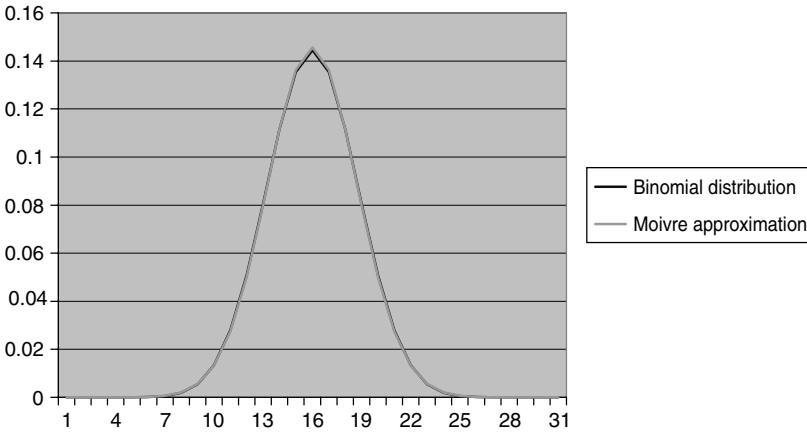


Figure 5.6 Exact probability versus Moivre approximation for $N = 30$.

$$P(\text{Bitrate}) = \frac{1}{\sqrt{2\pi}\sigma_b} * e^{\frac{-(\text{Bitrate} - \text{Average_rate})^2}{2\sigma_b^2}}$$

where $\sigma_b = (M - m)\sqrt{Na(1 - a)}$. Compared with the exact calculation, it is much easier to use this function when N is large (see Figure 5.5 for $N = 5$ and Figure 5.6 for $N = 30$). This can be found in most worksheets (in Excel this is the NORMDIST function).

5.1.3 Loss rate and dimensioning

From the previous results, we will try to evaluate which packet loss rate occurs when a bandwidth B is reserved on a link and there are N simultaneous conversations on that

link. Once we know this, it will be possible to calculate the optimal value of B for a given acceptable loss rate.

To facilitate the reading of the results, we will express B as a multiple of the elementary average bitrate of a single one-way voice channel during a conversation: $aM + (1 - a)m$.

5.1.3.1 Loss rate (without queuing)

Each time I sessions are simultaneously active, the offered bitrate is:

$$Incoming_bitrate(I) = IM + (N - I)m$$

If we drop packets to reduce the bitrate to B , the loss rate when I sessions are simultaneously active is (with fluid approximation):

$$Loss_rate(I, B) = \max(Incoming_bitrate - B, 0)$$

and the number of bits lost during a time interval T_i is:

$$Loss(I, B, T_i) = T_i * Loss_rate(I, B)$$

Over a time interval T , we group the packets lost for each subinterval during which I conversations are simultaneously active (the cumulated duration of each group is $T_i = P(I)$ if $T = 1$). We can calculate the average packet loss rate by dividing the total traffic lost by the offered traffic during T :

$$\begin{aligned} Average_loss_rate(N, B) &= \left[\sum_{i=0}^N Loss(I, B, T_i) \right] / \left[\sum_{i=0}^N T_i(Nm + I(M - m)) \right] \\ &= \frac{\sum_{i=0}^N T_i * \max(Nm + I(M - m) - B, 0)}{[Ma + (1 - a)m] * N * T} \end{aligned}$$

where we use relations (A) to simplify the denominator.

In order to find the minimal bandwidth of a link that has to carry N calls, one must increase B until the calculated percentage loss equals the maximum tolerable average loss rate. As an example, Table 5.4 is what we get for the G.729 coder with 3 frames per packet (including IP/UDP/RTP headers, with $M = 18.67$ and $m = 13.87$) and an activity rate of 0.5, where we considered that average packet loss rate had to be kept below 0.5% from 1 to 15 streams and below 1% for more than 15 streams. On average the bit loss rate and the packet loss rate are equal.

In fact, these results (based on average loss rates) should be considered with care when there are only very few conversations, because the active period at maximum bitrate (maximum loss rate) of the resulting stream can be very long (10 s or more for a single stream), and during this time users will experience voice quality with a loss rate equal to the loss rate when all streams are active, which can be much larger than the average loss rate; this is mitigated in reality by the small buffers of routers and results in jitter,

not packet loss, if not too many data are accumulated in the router buffer. We calculate in Table 5.5 an indication of the data that would accumulate during the peak period (considering that peak period duration is inversely proportional to the number of streams) if the router has sufficient buffers.

Note that not too many data can be accumulated in a buffer, otherwise the corresponding delay creates unacceptable jitter and packets are lost when they arrive at the destination. We need to dimension the link size and the size of router buffers in order to keep the packet loss low enough while not creating excessive jitter. It is acceptable to consider that some packets will be queued in small router buffers (e.g., 2 RTP packets or 60 ms of speech in our sample case of G.729 with 3 frames per packet), creating jitter, and that the buffer will overflow during peak periods, resulting in packet loss. This consideration leads us to suggest a target of 0.5% average loss for 1 to 15 conversations, not 1%, in order to take into consideration user perception of packet loss during peaks.

5.1.3.1.1 Overhead bandwidth, loss rate distribution

Here we use the term ‘overhead’ for the bandwidth that must be provided in excess of the average rate to accommodate peak traffic. This is unrelated to the IP overhead that was discussed in other sections.

When a link is used to carry only a few conversations, we note that it must be dimensioned almost as if all conversations were always active, which leads to a significant portion of the link bandwidth being unused most of the time. In the example above the maximum overhead (for a single conversation) is only 13.8%, because the M and m bitrates are not very different, due to the impact of IP headers on bandwidth. But on a link using compressed RTP ($M = 8.53$, $m = 3.73$), such as that illustrated in Figure 5.7, the reduction of bandwidth overhead with the number of conversations is much clearer, from 40% to less than 5%. As the number of conversations increases, the link size gets closer to the number of channels times the average bitrate of one channel and the link utilization rate increases. This is because the variance of a sum of n data streams (which approximates dispersion around the average), varies as \sqrt{n} and therefore the bandwidth overhead required for an acceptable loss rate, compared with the average total voice bitrate, varies as $\sqrt{n}/n = 1/\sqrt{n}$.

Table 5.5 Assessment of data accumulated in the router buffers during the peak period

Number of conversations	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Average bitrate	16.3	32.5	48.8	65.1	81.4	97.6	113.9	130.2	146.4	162.7	179.0	195.2	211.5	227.8	244.1
Minimal link bitrate	18.5	36.7	54.1	69.8	87.2	103.4	120.1	136.6	152.9	169.4	185.6	202.2	218.3	234.9	250.9
Equivalent channels	1.1	2.3	3.3	4.3	5.4	6.4	7.4	8.4	9.4	10.4	11.4	12.4	13.4	14.4	15.4
Overhead bitrate	2.2	4.1	5.2	4.7	5.8	5.8	6.2	6.4	6.4	6.7	6.6	6.9	6.7	7.1	6.8
Overhead (%)	13.8	12.8	10.8	7.3	7.2	6.0	5.4	4.9	4.4	4.1	3.7	3.6	3.2	3.1	2.8
Loss (%)	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
Cumulative peak loss (kbit)	1.6	3.3	6.5	12.2	12.3	14.3	15.2	16.0	16.9	17.3	18.0	18.2	18.8	18.9	19.4
Duration of peak (s)	10	5	3.333	2.5	2	1.667	1.43	1.25	1.11	1	0.91	0.83	0.769	0.714	0.6667
Peak delay in buffer (ms)	88	89	120	175	142	138	126	117	110	102	97	90	86	81	77

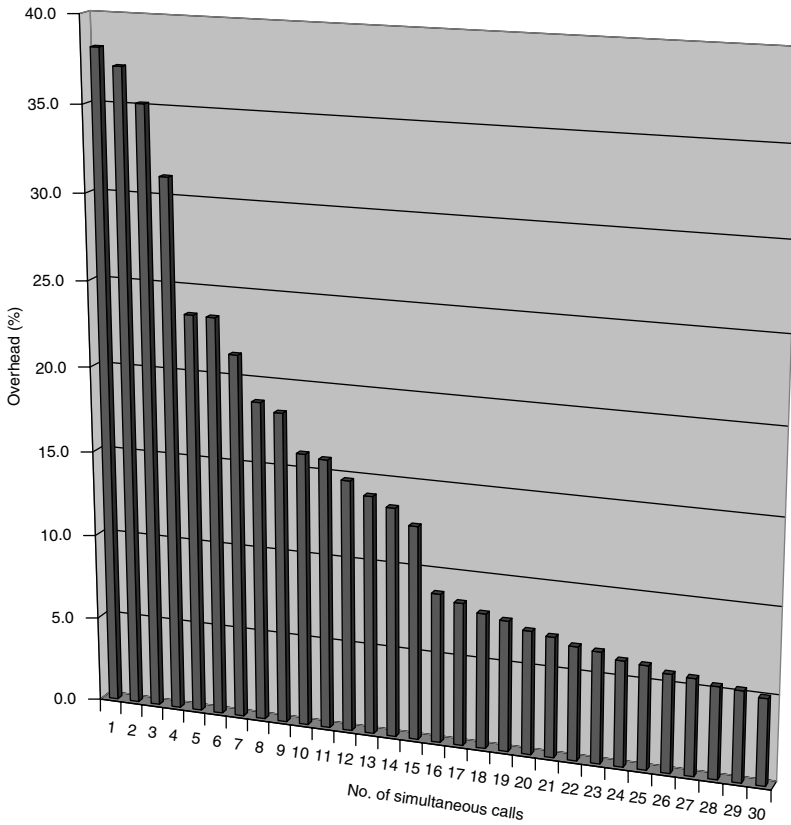


Figure 5.7 Bandwidth overhead required according to number of simultaneous calls, using cRTP.

As discussed above, we should keep in mind that what we call the ‘loss rate’ is an average value: during a single conversation it will be below this level most of the time, but can be significantly above it for short periods of time. This is illustrated in Figure 5.8: for each rectangle i , the x -axis represents the probability that i conversations are active (have a bitrate of M) simultaneously among N and the y -axis represents the loss rate when this occurs. The link itself is dimensioned to carry N conversations with an average loss of 1%.

Figure 5.9 shows the calculated values for $T = 30$, using a G.729 coder with 3 frames per packet. We represent only the upper right corner of each rectangle for clarity. This chart can be interpreted as follows: for a conversation of 1,000 s, there will be a 0% loss rate during most of the conversation (810 seconds), a loss rate between 0% and 5% for a cumulated total of 130 s, a loss rate of 5–10% for a cumulated total of 60 s, and more than 10% for no more than a few seconds. Of course, good, bad, and average quality periods will be interleaved in the conversation, and the periods with high loss rates (corresponding to a high bitrate at this instant) become shorter as the loss rate increases.

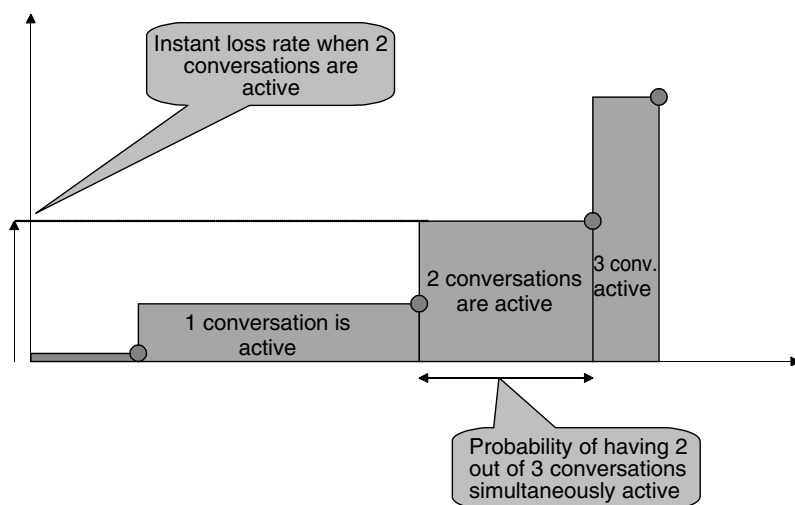


Figure 5.8 Probability of higher loss rates is lower. Loss profile on a link dimensioned to carry 3 conversations with an average loss of 1%, when all 3 conversations are established.

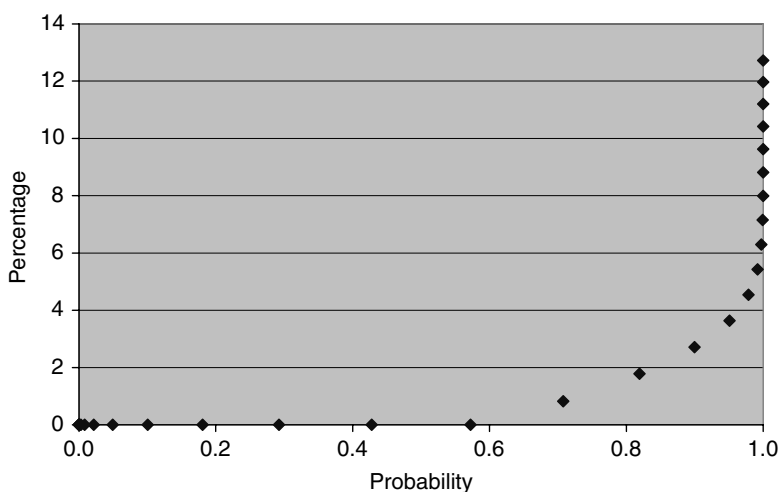


Figure 5.9 Diagram of loss rate probability for 30 active calls (G. 729, 30 ms).

This is important because most IP phones and media gateways recover well from the loss of an isolated packet; therefore, the periods of time corresponding to a very high packet loss rate will go unnoticed if they are very brief. Typically, if the average duration of a period of speech is 10 s, the duration of each occurrence with a bandwidth corresponding to N active streams simultaneously will be of the order of $10/N$ s. During this period, and assuming that loss is evenly distributed across all streams the order of magnitude of

routers do have buffers and therefore the real packet loss rates through an IP backbone would be lower than calculated above. If the provisioned bandwidth is higher than the average bandwidth, buffers convert packet loss into delay, as shown in Figure 5.11.

When, eventually, routers have very large buffers, packets will never be dropped by the network. So in theory we should use a much more complex model that takes into account buffering. This model would also need to take into account the jitter buffer discard policy of receiving terminals. For instance, the following formula holds for a single conversation, where b is the average duration of active periods, C the speed at which the buffer is emptied, and ε the loss rate:

$$0 = \beta \times \exp\left(-\frac{\text{Buffer_size} \times (C - m - a(M - m))}{b \times (1 - a)(M - C)(C - m)}\right) - \varepsilon = f(\text{Buffer_size}, C)$$

$$\beta = \frac{(C - m - a(M - m)) + \varepsilon a(M - C)}{(1 - a)(C - m)}$$

This must be solved numerically to find the appropriate values of C for a target loss rate. Additional details and an approximation of C can be found in ‘Equivalent capacity and its application to bandwidth allocation’, by Guerin et al. (1991).

Figure 5.12 ($M = 9.07$ kbit/s, $m = 3.73$ kbit/s, $b = 10$ s, $\varepsilon = 0.01$, $a = 0.5$) shows that C can be reduced significantly as the size of the buffer increases. But, in our application, real-time telephony, we cannot take advantage of large buffers. The problem is that queued packets get delayed, reducing the interactivity of the conversation, and the increased jitter will cause some packets to be discarded by the jitter buffer of the receiving terminal if the extra delay exceeds a given threshold.

Do we really want to gain a few percent of bandwidth at the expense of reduced interactivity? As we saw in Chapter 3 on voice quality, the greatest constraint of IP telephony is achievable end-to-end delay, which is in most cases at the extreme limit of

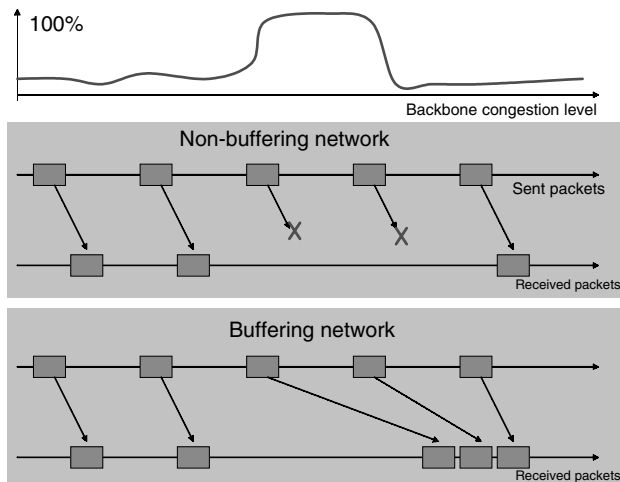


Figure 5.11 Buffers turn packet loss into additional delay.

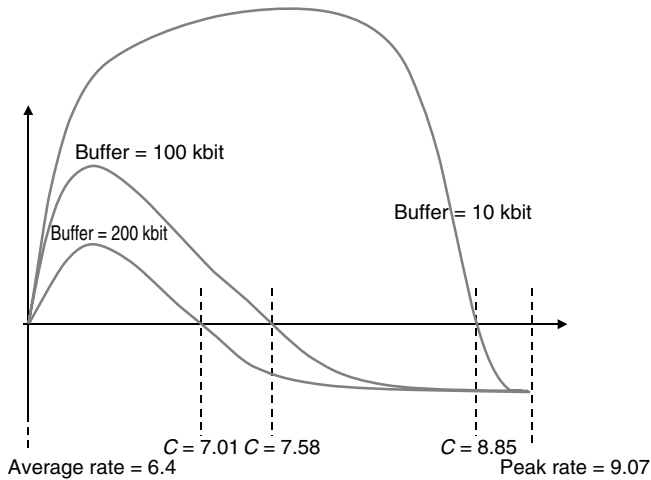


Figure 5.12 Required capacity C for a target loss rate of 1%, according to buffer size.

what is considered acceptable for an average listener. By letting voice packets accumulate in the network, we would increase jitter and force terminals to adapt by building larger jitter buffers: this would lead, inevitably, to additional delays.

Therefore, we think that it is good practice to dimension voice over IP links on the basis of there being no buffering. This leads, for small links, to some overprovisioning, but we will see in the next sections that this overhead can be used by non-real-time traffic.

5.1.4 Packet or frame loss?

Up to now we have calculated the characteristics of the offered bitrate and deduced how much of it could not get through a line of given bandwidth. In doing so we used the ‘fluid approximation’; in fact, routers will not throw away one bit here and there, but whole IP packets. However, this does not change our result which is also valid for the packet loss rate, the bit in excess causes loss of a complete packet (error multiplication), but as a complete packet is pumped from the buffer, other areas that would have experienced bit excess now have available capacity.

Is this the same as the frame loss rate?

If, as we have assumed in our calculations, the frames generated by the coder are simply grouped in IP packets, then the answer is yes. But this is not always true, as some manufacturers use redundancy schemes based on interleaving redundant frames in multiple packets, in order to recover from packet loss. This is often the case for non-standard gateways optimized to work on the Internet, where the bandwidth is ‘free’. The redundancy scheme used by such gateways leads to greater bandwidth usage, but, since it remains only a tiny fraction of overall Internet traffic, it will in most cases not significantly increase the packet loss measured between two gateways.

A basic redundancy scheme would simply be to copy one coder frame in every two IP packets ... if only one IP packet out of two is lost, then the frame can be recovered

and there will be no frame loss. We have seen demonstrations by a manufacturer proudly showing a 50% packet loss with perfect voice quality and doing so using this trick! Of course, a real network is not as simple. The manufacturer's IP network simulator was dropping *exactly* one packet out of 2. The Internet is not so well behaved and will frequently drop entire sequences of packets in a row, so the basic redundancy scheme would not work that well under real conditions!

Our personal advice is to be *very* careful when considering marketing material that proudly announces 'magic' redundancy schemes that beat the competition out of sight. Such material fails to point out that it is most unfriendly to jam the Internet with such a careless use of bandwidth and that there is frequently a delay trade-off when using redundancy.

If, however, you do want to use redundancy, find out how the redundancy scheme performs in the presence of grouped packet loss (or even better, test it). Then look at the packet loss characteristics of the network you will be using (isolated packets or many packets dropped in a row) and decide if the particular redundancy scheme in use will work as advertised. Finally, measure the end-to-end delay that is obtained and check that you are still within the 'acceptable' range.

5.1.5 Multiple coders

So far we have only considered one type of coder, with the same value of M and m for all voice flows. Different types of customers may require different qualities of service, and therefore different coders may be used.

A possible approach to network dimensioning in this case is to rely on the Gaussian approximation to traffic generated by one class of coders. Remember that, in the case of N independent and identically distributed flows, traffic distribution could be considered for large values of N to be Gaussian distribution:

$$P(\text{Bitrate}) = \frac{1}{\sqrt{2\pi}\sigma_b} * e^{\frac{-(\text{Bitrate} - \text{Average_rate})^2}{2\sigma_b^2}}$$

where $\sigma_b = (M - m)\sqrt{Na(1 - a)}$.

An interesting property of the sum of independent variables is that the variance of the sum equals the sum of the variances of each variable ($\sigma^2 = \sum \sigma_i^2$). In addition, if each variable follows a Gaussian law, then the sum is also a Gaussian law.

So an approximation to the traffic distribution in the case of multiple coders is a Gaussian law, with an average value equal to the sum of average values and a variance equal to the sum of variances obtained for each group of conversations with the same type of coder.

At this stage it is easy to calculate what proportion of the traffic is discarded for a given capacity C :

$$\text{Loss_bitrate} = \int_C^\infty B * P(B) dB$$

and the loss rate:

$$\begin{aligned}
 Loss_rate &= \frac{1}{\sqrt{2\pi}\sigma * Average_rate} * \int_C^\infty B * e^{-\frac{(B-Average_rate)^2}{2\sigma^2}} dB \\
 &= \frac{\sigma}{\sqrt{2\pi} * Average_rate} * e^{-\frac{(C-Average_rate)^2}{2\sigma^2}} + \frac{1}{\sqrt{\pi}} \int_{\frac{C-Average_rate}{\sqrt{2}\sigma}}^\infty e^{-x^2} dx \\
 &< \left(\frac{\sigma}{\sqrt{2\pi} * Average_rate} + \frac{\sigma}{\sqrt{2\pi}(C - Average_rate)} \right) * e^{-\frac{(C-Average_rate)^2}{2\sigma^2}} \\
 &= \left(\frac{\sigma(C - Average_rate) + \sigma * Average_rate}{\sqrt{2\pi} * Average_rate * (C - Average_rate)} \right) * e^{-\frac{(C-Average_rate)^2}{2\sigma^2}}
 \end{aligned}$$

For this calculation we used the identity:

$$\int_a^\infty e^{-x^2} dx < \int_a^\infty \frac{x}{a} e^{-x^2} dx = \frac{1}{2a} e^{-a^2}$$

Finding C for a target loss rate ε amounts to calculating the inverse of this function; this can be done numerically or, when the average rate is large compared with the standard deviation (in the first term approximate $(C - Average_rate) \simeq \sigma$), using the following approximation (also given by Guerin et al., 1991):

$$C = Mean_aggregate_bitrate + \alpha \times Standard_deviation_of_aggregate_flow$$

where $\alpha = \sqrt{-2 \ln(\varepsilon) - \ln(2\pi)}$.

5.2 Building a network dedicated to IP telephony

5.2.1 Is it necessary?

We will see in the following sections that it is feasible, and in fact desirable, to mix all types of IP traffic on one common backbone. But, in order to keep a reasonable quality-of-service level for voice flows, it is necessary to have routers with sophisticated queuing capabilities.

5.2.2 Network dimensioning

Suppose we have six sites in three countries that need to communicate using IP telephony. How do we dimension such a network (Figure 5.13)?



Figure 5.13 Sample VoIP network.

5.2.2.1 Traffic matrix

The first step is the same as that in switched telephone networks: we need to know who phones where, how often, and for how long. This information is usually derived easily from existing phone bills during a reference period. We then need to choose an optimal route on the network for each of those calls. For this we need to know the cost of each link per unit of bandwidth (e.g., in the case of a leased line it is monthly fee divided by bandwidth).

The cost of carrying IP traffic from A to B or B to A is not always the same, since a provider could decide to use a satellite link one way. But satellite links add to end to end delay. Let's consider symmetric cost in the context of Table 5.6. This cost matrix could also depend on time, as some providers have discount rates during off-peak hours.

Table 5.6 Cost matrix of the sample network

From/To	1	2	3	4	5	6	A	B	C
1							2		
2							1		
3									1
4									1
5								1	
6								2	
A	2	1						10	10
B					1	2	10		
C			1	1			10	5	

If there is only ‘on-net’ traffic, then the mapping of each phone call to a route in the network is straightforward: we simply take the least costly route at the time of the call (e.g., a phone call from 1 to 6 will be routed through A and B as opposed to A–C–B). For a more complex cost matrix, the cost-optimized path for reaching each destination from a given origin can be calculated recursively by calculating, for each destination, the number of hops needed to reach it and for which price, then repeating the algorithm for each previous hop and summing the total cost. This results in a number of paths that reach each origin, allowing us to select just the lowest cost one.

Once each call is mapped to a route, we need to calculate the maximum number of simultaneous calls over each link (the so-called ‘busy hour’ traffic).

When there is also ‘off-net’ traffic, the optimal route through the network is calculated by minimizing the ‘on-net’ cost added to the cost of the hop from the last ‘on-net’ node and the final destination over the switched telephone network. In most cases the route ends at the ‘on-net’ node closest to the final destination. But, things can get less rational when the last hop crosses national borders! Some calls (e.g., local calls) may not be routed through the IP network at all.

5.2.2.2 Link sizing

Once each call has been mapped to an optimal route, we are able to calculate the number of simultaneous calls on each link at any given time. In general the link bitrate cannot be adjusted dynamically, so we must use the peak number of simultaneous calls (‘busy hour’) as our input to dimension the link and use the Erlang formula (see Section 5.5.4.2 and 5.5.4.3) to plan for variations during this busy hour. At this stage we know the maximum number of simultaneous calls that will be routed on each link.

The difficulty is that the link size also depends on the acceptable average packet loss rate. If we allow for more loss, then the link can be adjusted to a smaller capacity. Table 5.7 shows the bandwidth overhead compared with the average bitrate for a given packet loss (G.729, 3 frames per packet).

For a given coder in a given configuration, it is relatively easy to find the acceptable average packet loss using simple tests. This packet loss can be considered as our end-to-end budget that needs to be split among all the possible paths for a phone call through our network. There are many ways to split this budget, and they are not equivalent. For instance, in Table 5.8 we consider two links: one is a transatlantic line (L_2) with a cost

Table 5.7 Required capacity for a target loss rate

Simultaneous calls	Loss rate (%)					
	3.00	1.00	0.50	0.10	0.01	0
1	9	13	14	15	15	15
5	0	5	7	12	15	15
10	–2	2	4	8	11	15
30	–3	0	1	4	6	15
100	–3	–1	0	2	3	42

Table 5.8 Costly links should consume most of the packet loss budget

$L_1 =$	0.9%	0.7%	0.3%	0.1%
$L_2 =$	0.1%	0.3%	0.7%	0.9%
Channels on link 1	10			
Channels on link 2	100			
Overhead (link 1)	2.4%	3.0%	5.3%	7.7%
Overhead (link 2)	1.6%	0.7%	-0.2%	-0.5%
Distance on link 1	1			
Distance on link 2	100			
Cost on link 1	10			
Cost on link 2	10,000			
Total overhead cost	163.13	71.10	-20.76	-53.73

of 10,000 for each supplementary unit of bandwidth; the other is a local link (L_1) with a cost of 10 for each supplementary unit of bandwidth. This gives us a loss budget of $L_1 + L_2 = 1\%$.² In this example we see that the best way to split the loss rate is to give most of the loss budget to the transatlantic line.

This issue is very complex in general. The same transatlantic link could serve hundreds of smaller local access links; in this case the cumulative overhead cost of all the local lines could become larger than the cost of the transatlantic link ... This is a nonlinear optimization problem and its resolution is outside the scope of this book. Fortunately, the constant drop of bandwidth costs over long-distance links makes this optimization a bit futile in most cases.

5.2.2.3 Fault tolerance

The phone network is a mission-critical system, and the failure of one IP link should not jeopardize corporate communications. This is why a careful network planner will plan ahead for such failures and assess their consequences. This is done by considering the network without this link and rerouting all phone calls according to the new topology (this is done automatically by IP routers). The network must be dimensioned to handle this configuration.

Planning for link outages therefore consists in simulating all possible topologies such that any of their constituent links can fail ($F(\text{link}_i)$) and then dimensioning the remaining links to handle the rerouted flows. The final capacity of a link should be sufficient to compensate for all possible $F(\text{link}_i)$.

Some critical backbones may even be dimensioned to handle the simultaneous failure of two links. Note that this step is not necessary if the layer 2 protocol already has such fault protection (e.g., SDH rings, etc.) and is already dimensioned to support rerouted traffic.

² Loss rates are not additive. For example, in Table 5.8 $(1 - L_2)(1 - L_1) = 0.99$; however, for small values of L_1 and L_2 the $L_1 * L_2$ term is negligible.

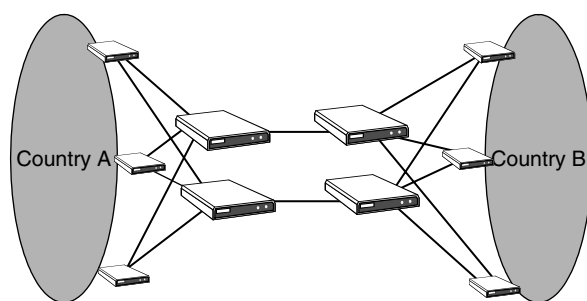


Figure 5.14 Fully fault-tolerant IP network.

A potentially more significant problem is the loss of a router. If the router connects N links, then the situation is the same as if all these links were down. As this is a layer 3 failure, it will not be recovered by layer 2 security mechanisms (SDH rings, etc.). This type of failure can generally be avoided if network providers use redundant configurations such as the one shown in Figure 5.14, where traffic can always be rerouted even if one router goes down.

5.3 Merging data communications and voice communications on one common IP backbone

The previous sections show that it is necessary to plan for some overhead capacity when designing an IP network dedicated to IP telephony. It is very tempting to use this spare capacity for best effort data when it is unused by voice flows. But, what are the consequences?

5.3.1 Prioritization of voice flows

The first condition, in order to carry voice flows over a general purpose IP backbone, is to guarantee that packet loss and network delays and jitter will be kept to a minimum for voice packets. There are several ways of achieving this:

- Do nothing: TCP traffic backs off when facing network congestion. Therefore, UDP traffic (and RTP over UDP) will tend to occupy whatever bandwidth it needs at the expense of TCP traffic. However, this adaptation of TCP traffic is rather slow and works by ‘trial and error’: it sends traffic first and interprets packet loss as congestion. Therefore, TCP traffic will always grow until it congests the network, and having done so backs off again. It will maintain the network in near-congestion state and cause some marginal packet loss on UDP traffic in order to ‘get a feel’ of the state of network congestion. Moreover, there are some TCP stacks that do not respect this back-off strategy. All considered, relying on TCP congestion control algorithms in order to prioritize UDP is not safe to build reliable VoIP networks, even on private networks.

- **Prioritize all UDP traffic:** This is the easiest way to prioritize voice flows, since IP telephony RTP packets are carried over UDP. The side-effect is that all other UDP flows (such as DNS queries) are also prioritized. This is fine in general, because most applications written on top of UDP need minimal delays. In public networks, however, this is a very dangerous practice. Soon, some bright spark will discover that it is actually quite simple to simulate a TCP connection over UDP, and chances are that after a while more and more customers will begin to use all sorts of tricks to send most of their traffic (e.g., peer to peer), not just voice, over UDP! Our advice is to use this method only on corporate backbones, where backbone traffic is well understood and connected networks are well behaved.
- **Use IP precedence levels (DiffServ):** Many routers can be configured to use IP precedence, or DS, information (see Chapter 4 on QoS for more details) in IP packets to prioritize classes of traffic. Routers can be configured in several different ways:
 - Assigning a minimal bandwidth to each class.
 - Assigning a weight to each class and sharing the available bandwidth between classes that have a backlog proportional to these weights.
 - Giving head-of-line priority to a class.

Each method is discussed in detail in Chapter 4. The way in which many of them are used depends on the other flows that are being carried on the network. It is important, when evaluating one method or the other, to verify the behavior of the scheduling algorithm being used regarding delays. We have found it sometimes very difficult to obtain such data from manufacturers. It is not enough just to have the name of the algorithm (e.g., WFQ is now used as a marketing term by many manufacturers for all scheduling algorithms that exhibit selective prioritization properties).

It is also necessary not to ignore the behavior of level 2 multiplexing algorithms: if your IP network is built on top of plain frame relay links (without priority extensions), you may prioritize some IP packets at the IP level... but the frame-relay switches will gladly ignore this information! Connectivity providers using ATM, MPLS, or SDH backbones are likely to offer better support for differentiated classes of service, but it is always useful to check delays and jitter over a shared backbone before trying to use it for IP voice.

5.3.1.1 Example configuration

Let's build a backbone supporting three classes of service:

- Real-time class for voice.
- Committed bandwidth class.
- Best effort.

Each link between our backbone routers is bought from an ATM connectivity provider. We also ask for provision of ABR (available bitrate) capacity on the virtual channels. The minimal cell rate is calculated by adding the bitrate needed for IP telephony traffic

(calculated as above) to the bitrate that we want to have available for the committed bandwidth class. The maximal cell rate depends on how much bandwidth we want to offer for the best effort class.

If our provider does not have ABR, we can use CBR (constant bitrate) mode for the aggregate capacity of the real-time and committed bandwidth class, plus the spare capacity that we always want to have for best effort traffic.

Then we configure our ingress routers to assign a DSCP codepoint = 010000 to all flows generated by IP telephony gateways (e.g., through a route-map) and TOS = 1 to all flows that need a committed bandwidth. The ingress router for such flows must also be configured to check that the bandwidth never exceeds what was reserved, in which case out-of-profile packets are marked as ‘best effort’ traffic (DSCP 000000).

If class-based queuing (CBQ) is available on our routers, we configure the queue for DSCP 010000 to always get priority treatment over 001000 and 000000. Similarly, we configure the queue for DSCP 010000 to get priority over 000000. With this configuration, IP voice packets between our gateways will never have to wait behind non-real-time traffic and committed bandwidth traffic will push back best effort traffic.

The profile of the network load on our link will look like Figure 5.15. On public backbones, this traffic mixing happens to be very favorable: business users will use a lot of bandwidth between 8 a.m. and 6 p.m., but this bandwidth will be freed for the ‘web rush’ of residential users between 8 p.m. and 11 p.m. During the empty hours (12 p.m.–6 a.m.), the provider will be able to offer special pricing for bulk data transfers.

5.3.2 Impact on end-to-end delay

Mixing traffic of different types does have an impact on end-to-end delay. This is because a router needs to wait until it has finished sending a packet before it can service the next

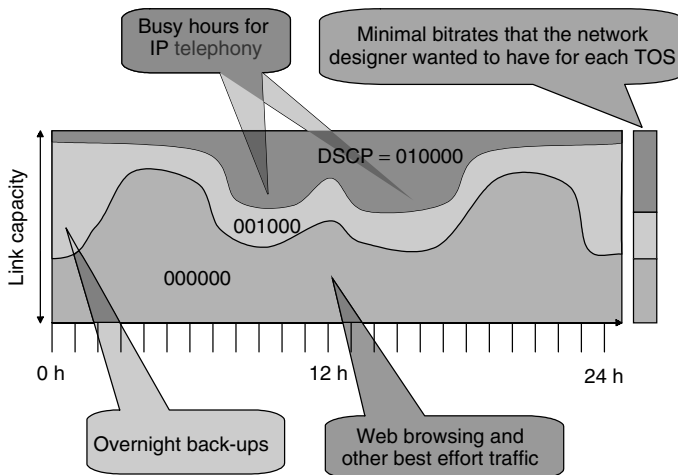


Figure 5.15 Higher priority classes ‘eat’ the best effort budget when necessary.

one. Even if the next packet is a high-priority one, it will have to wait. On backbones dedicated to IP telephony, all packets will be small (unless techniques such as RTP-mux are used) and therefore this waiting delay will be less than on a backbone on which most packets are 1,500 octets long.

This is only worth considering on low-bandwidth links: waiting time quickly becomes negligible as the link bandwidth increases (1,500 octets are sent in 1.2 ms on 10-Mbps links). Moreover, on backbones carrying many more data flows than voice flows, the links used will have a much higher bandwidth than on a backbone dimensioned only for voice. If voice is prioritized, the delays offered on such a backbone may well be lower than those that would be observed on a dedicated voice backbone!

Clearly the trend is for ever-increasing data traffic, especially as voice traffic is limited to 24 hours per day per person! Letting voice take advantage of the high-capacity links of the data backbone is the way forward for future networks.

5.4 Multipoint communications

5.4.1 Audio multipoint conferences

5.4.1.1 *Star topology with centralized flow mixing*

In this section we discuss the typical topology of a conference in multi-unicast mode with a central MCU. For a conference, activity rate a is calculated by considering the conference active when at least one person speaks (Figure 5.16). Parameter a will typically be in the 0.7–0.9 range, depending on whether people are bored or overexcited during the conference!

In this topology each link between the mixing server and each terminal carries asymmetric flows. If there are P participants in the conference, each speaking in turn and for the same amount of time:

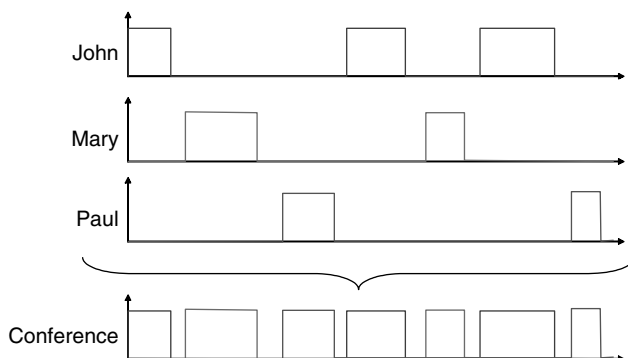


Figure 5.16 Activity rate of a multiparty conversation.

- From the server to each terminal the average traffic will be $aM + (1 - a)m$, with a peak value of M .
- From each terminal to the server, the average traffic will be $(a/P)M + (1 - a/P)m$, with a peak value of M .

The problem with multi-unicast conferences is obvious: if many participants (say, N) are behind the same link, then average downstream traffic from the server is $N \cdot (aM + (1 - a)m)$. Moreover, each flow from the server to a terminal is strongly correlated and, therefore, the peak bitrate $N \cdot M$ will be reached most of the time.

Another major issue, even if there is plenty of bandwidth, is delay. The central server must decode the audio signal it receives from each terminal, calculate the sum of all signals except the signal from the participant himself, and then recode those signals and return them. Before it can sum the signals, the central server must receive enough voice frames from each participant and synchronize them: this requires a jitter and synchronization buffer that will add at least the duration of the longest voice frame to reception delay. All other delays depend on the CPU power available, but can generally be assumed to have a duration of 2 voice frames.

Another source of delay is the nonoptimal route of voice flows, which must pass through the central server instead of reaching the receiving terminal via the shortest route. Finally, the receiving terminal still has a jitter buffer. Figure 5.17 illustrates these delays.

We conclude that multi-unicast conferences work best when installed on non-bandwidth-constrained networks and the multipoint processor is inserted in the optimal path

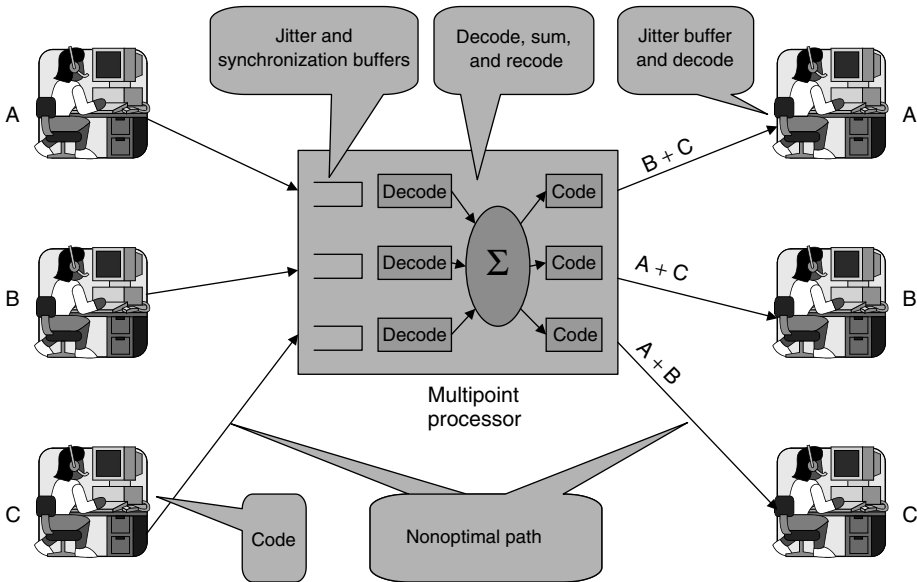


Figure 5.17 Multipoint processors add significant audio delay. For clarity all terminals are represented twice: once in their sending role, once in their receiving role.

between the participants. Codecs should be used in low-delay mode (i.e., minimal number of frames per packet as allowed by bandwidth constraints).

5.4.1.2 Star topology with flow switching

In this configuration the central server does not sum signals, but rather transmits to all participants the signal that was last measured ‘most active’. Traffic evaluation is the same as before, but delays will be lower since the coding stage needed to compress the sum signal is not needed (the decoding stage is generally still needed to evaluate signal energy, unless silence detection is used by senders). This technique allows building very simple and scalable multipoint processors.

However, this comes at a price: the conference becomes very sensitive to parasitic noise sent by senders and conference participants need to be disciplined, since it is generally impossible to interrupt an active speaker.

It is possible to improve things by building multipoint servers that mix only the last two or three most active signals: they remain scalable and yet make it possible to interrupt the active speaker. Most commercial bridges use such a compromise.

5.4.1.3 Using multicast with source-based trees

This is the technique used for conferences on the mBone, the multicast backbone of the Internet. With multicast traffic, measurements depend a lot on the network topology between conference participants (we will examine a few typical situations below). In all cases, end-to-end delay is improved, since all flows generally follow the shortest path and the decode/recode stage is avoided.

5.4.1.3.1 Conferences over an Ethernet LAN

Now, with the same assumptions and definitions as before, each participant generates average traffic of $(a/P)M + (1 - a/P)m$ with a peak bitrate of M . The traffic generated by P participants is superposed on the network, and the average aggregate bitrate is therefore $aM + (P - a)m$, not much higher than the value obtained for a *single conversation*. If the value of a for the conference was the same as for a two-party conversation and the coder was sending no data during silences ($m = 0$), then it would be equal. Note that many coders have a discontinuous transmission feature (DTX), and so we can actually have $m = 0$ during long silences.

However, the possible peak value of aggregate traffic is $P*M$ and is obtained if all voice flows are correlated or if everybody talks simultaneously. Put another way, it is a bad idea to start a karaoke club on the Internet using multicast technology based on source-rooted trees! For normal voice conversations, however, the probability of having such a situation is very low. It is possible to limit the peak rate to $K*M$ deterministically by using terminals that refuse to send data if more than K different SSRCs are detected in incoming RTP flows (or K different source IP addresses).

The previous calculation is valid for coax or thin-coax Ethernet. However, before using a hub or a switch, *always check how they handle multicast traffic*. People usually buy

switches to prevent traffic generated by two hosts from being communicated to other hosts. Unfortunately, some older or cheap switches will turn multicast traffic into broadcast. Although this may work and your multicast conference may be fine, all other connected hosts will also receive the aggregate traffic of the conference. I've heard of networks with thousands of machines suddenly becoming desperately slow just because three guys were having a multicast videoconference!

Switches should have a way of taking into account multicast group membership information. Protocols such as CGMP (Cisco) or GARP can be used; another solution is to use switch/routers.

5.4.1.3.2 LANs connected to a central router

In this topology, multicast packets are duplicated by the central router and sent to all terminals on other LANs. As before, the average traffic of each terminal to the central router is $(a/P)M + (1 - a/P)m$ with a peak bitrate of M .

If there are K participants on a LAN, then the average traffic from the central router to this LAN will be $(P - K)[(a/P)M + (1 - a/P)m]$. This will be added to the traffic that is generated locally on the LAN, so the aggregate traffic is still $aM + (P - a)m$. The same remarks as before apply to the peak bitrate.

5.4.1.3.3 Larger networks

On larger networks each source sends $(a/P)M + (1 - a/P)m$ to a tree that reaches every participant except the source. The tree duplicates traffic according to the topology (for DVMRP, duplication is done, if there are several possibilities, where it leads to the shortest source-to-destination delays).

Comparing this with the multi-unicast situation depends on the topology of the multicast tree built by the multicast protocol. If multicast packets are duplicated very close to final users, then network load is minimized. The issue is that many multicast protocols are optimized to minimize delays and packets are duplicated at the node that minimizes the number of hops to the destination. This arguably is not the best way of minimizing traffic.

However, in some cases the shortest delay solution coincides with the lowest network load solution; that is, where there is only one possible distribution tree: in hierarchical networks (Figure 5.18).

If $N(L)$ is the number of participants located on the part of the distribution tree rooted at link L , then we will have on this link:

- Average downstream traffic of $(P - N(L))[(a/P)M + (1 - a/P)m]$.
- Average upstream traffic of $N(L)[(a/P)M + (1 - a/P)m]$.

If we had been using a multi-unicast server at level i , the average value of downstream traffic would have been:

$$N(L)[aM + (1 - a)m]$$

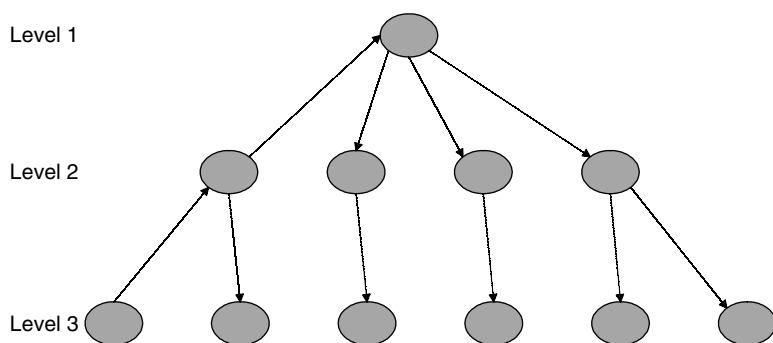


Figure 5.18 Sample hierarchical network. The shortest delay tree is also optimal for lowest bandwidth usage.

and the average value of upstream traffic:

$$N(L) * [(a/P)M + (1 - a/P)m]$$

From these evaluations it is clear that the multicast solution scales better for downstream traffic if there are many participants behind the link and $m \ll M$. For upstream traffic the scalability of both solutions is identical.

5.4.1.4 Using a shared tree multicast technology

In this technique all traffic is sent to a node (this node can be different for each conference), and multicast diffusion starts at this node (Figure 5.19). In this case, for each link L :

- Average downstream traffic is $P * [(a/P)M + (1 - a/P)m]$.
- Average upstream traffic is $N(L) * [(a/P)M + (1 - a/P)m]$.

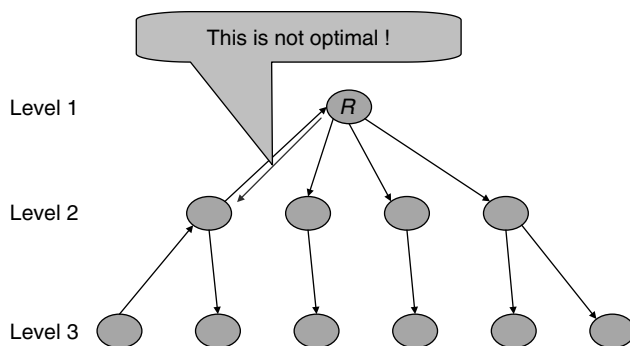


Figure 5.19 Shared tree multicast distribution. All sources send their flows to node R , which initiates multicast traffic.

We see that, due to the fact that local sources are heard through the central node, the result is not as good as before in terms of bandwidth. The delays are not optimal either. However, it remains better than multi-unicast if there are many participants behind that link and $m \ll M$.

5.4.1.5 Using a hybrid unicast-multicast technique

This technique allows every source to send their flows in unicast to a central server, but the central server is allowed to use multicast. The central server can voice-switch and redistribute the most active flow in multicast mode: although this is the most scalable design, the speaker cannot be interrupted. Another solution is to redistribute each source in multicast mode, without any mixing; this design has the same scaling properties as the shared tree multicast technique of Section 5.4.1.4, except that the number of retransmitted streams can be limited to the K most active speakers.

Note that it is impossible to perform mixing at the central node and redistribute the mixed flow in multicast to all conference participants, since the active speaker would hear his own voice echoed by the server. However, when there are a lot of passive participants (e.g., in a ‘panel’ type of conference where there are few speakers and a large audience), this mixing option is very attractive for the audience, while speakers can still use multi-unicast conferencing. This hybrid mode is used, for instance, in H.332 conferences.

5.4.1.6 Conclusion

Using multi-unicast servers for multipoint conferences is only possible for small-scale conferences. This model can be extended to the ‘panel’ service by sending the mixed audio signal to all passive members using multicast.

For larger scale conferences, the cheapest solution is to use audio switching with multicast, but this provides interactivity that is limited to a few active speakers.

High-quality conferences with many speakers can only be provided efficiently using multicast technology. There is a small difference in the bandwidth-scaling properties of shared tree and source-rooted trees, but not significant enough to be a good reason to choose one or another. If delay is a concern, source-rooted trees are a better option; but, overall the major factor of decision will be the scaling properties of such networks regarding the number of simultaneous groups that can be supported by routers (see Chapter 6 on multicast routing for more details).

5.4.2 Multipoint videoconferencing

There is one major difference between audio and video conferences: video data grow linearly with the number of participants, whereas audio data are self-limiting (not everybody talks at the same time!).

In order to limit the amount of video data, most conferencing systems use video switching (i.e., broadcasting the image of the last active speaker only). Another option is to build a composite image ('mosaic') from all the incoming video streams. In this case we reach the same conclusions as for audio traffic:

- If multi-unicast is used, then traffic on each link grows linearly with the number of participants connected via this link, which rapidly leads to scalability problems.
- If the central switching server broadcasts the selected image using multicast, then the conference can grow to accommodate a larger number of active participants (sending video data), but is not limited in the number of passive participants (receivers only).

For high-quality 'continuous presence' conferences, the best solution is to have every participant send their video data to all others using multicast. Network usage is proportional to the number of simultaneous active senders.

5.5 Modeling call seizures

5.5.1 Model of call arrivals: the Poisson process

The Poisson process is also called the 'memoryless' process. It is used to model many situations where the probability of future events does not depend on past events. Put another way, when knowledge of past events tells us nothing about the future (e.g., the Poisson process can be used to model the disintegration of radioactive elements). Under certain conditions it can be used to model the occurrence of incoming calls.

Let's number each new call set-up as 'event number i ' and its timestamp T_i . The time interval between event $i - 1$ and event i is t_i (Figure 5.20).

If the probability of having a new call between t and $t + dt$ does not depend on previous events and is proportional to dt (first-order approximation) with a negligible probability that more than one call arrives when dt is small, then it is a Poisson arrival process.

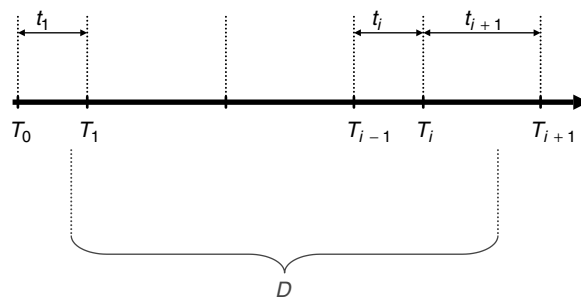


Figure 5.20 The Poisson call arrival model.

If we call $P_k(D)$ the probability of having exactly k calls during a time interval D , then our assumptions can be written:

$$P_1(d) = \lambda \times d + o(d)$$

$$1 - P_0(d) - P_1(d) = o(d)$$

$$P_{k+1}(D + d) = P_{k+1}(D)(1 - \lambda d) + P_k(D) \times \lambda d + o(d)$$

where the last line is obtained because we can have $k + 1$ events at $D + d$ if we have:

(a) $k + 1$ events at D and no event between D and $D + d$.

or

(b) k events at D and one event during d .

This leads to a differential equation whose solution is:

$$P_k(D) = \frac{(\lambda D)^k}{k!} e^{-\lambda D}$$

and if we call $A(t)$ the probability that $t_1 < t$, we have:

$$A(t) = 1 - P(t_1 > t) = 1 - e^{-\lambda t}$$

The probability of not having any event during t is purely exponential, and the Poisson distribution is often called the ‘exponential distribution’:

$$B(t) = 1 - A(t) = e^{-\lambda t}$$

In fact, λ is *the average frequency of the event*, and $1/\lambda$ is *the mean inter-arrival time* ($\bar{t} = \int t A'(t) dt = 1/\lambda$).

Incoming phone calls are often modeled as a Poisson process in large networks, even if one could argue that our hypothesis that an event does not depend on past events is not always true. This is because the superposition of N independent processes P_i with an average event frequency λ_i , even if they are not individually Poisson processes, converges to a Poisson process when N increases (Palm–Kintchine theorem). The average frequency of this Poisson process is the sum of all λ_i . This is exactly what happens for a set of N telephone lines. If f is the frequency of calls for each line, A the total traffic on the network, and d the average duration of a call, call set-up events can be considered as a Poisson process where $\lambda = Nf = A/d$. Figure 5.21 shows the inter-arrival distribution obtained for 100 lines and 1 call/hour per line.

5.5.2 Model of a call server

5.5.2.1 Queue model

The problem of arriving calls that are served, queued, or rejected by a telephone switch is an example of a queuing problem. A queue is modeled as a number S of ‘sources’ (finite or

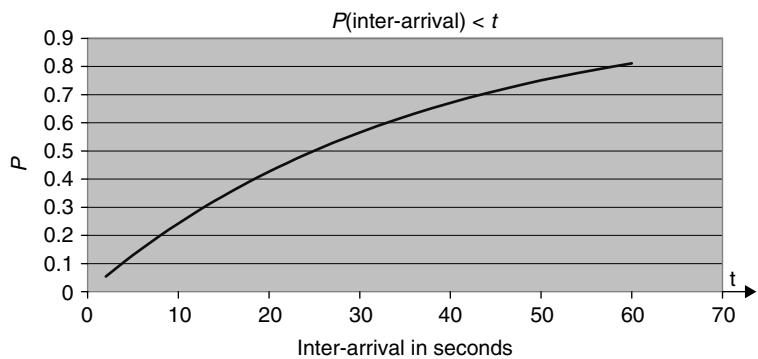


Figure 5.21 Probability of interarrival period to be smaller than t .

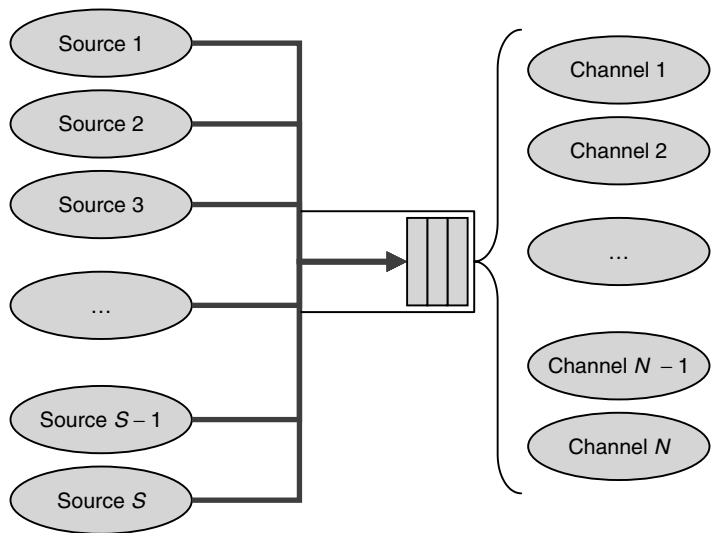


Figure 5.22 Generic queue model.

infinite) which generate requests processed by a number N of ‘channels’ (Figure 5.22). These requests can be served immediately if there is an available channel, queued if allowed by the system, or rejected if the queue buffer is full or does not exist.

Each source is modeled as a two-state process: the ‘idle’ interval is exponentially distributed with intensity λ , and the busy interval is exponentially distributed with intensity μ (as shown in Figure 5.23).

5.5.2.2 Different assumptions lead to various models

The various models listed in Table 5.9 all derive from the same queue model, but with different assumptions. Each model is described in the following sections.

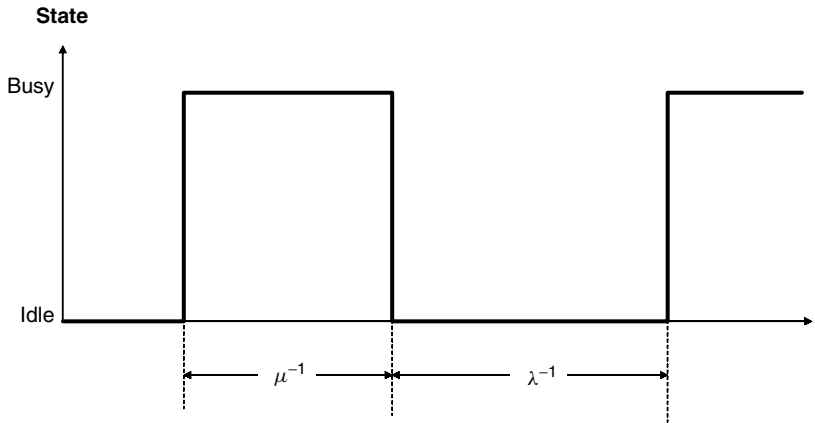


Figure 5.23 Two-state exponential source model.

Table 5.9 Model to use according to traffic assumptions

Call arrival model	Independent sources, two-state model, each with an exponential distribution				
Call arrival intensity	Varies proportionally to the number of on-hook lines		Fixed (variations according to the number of on-hook lines are neglected)		
Congestion	No congestion: full availability $N = S$	Congestion: $N > S$	No congestion: full availability $N = S$	Congestion: $N > S$	
Congested calls	None	Lost	None	Lost	Queued
Service model	Any distribution				Poisson + FIFO
<i>Model</i>	<i>Bernoulli</i>	<i>Engset</i>	<i>Poisson</i>	<i>Erlang B</i>	<i>Erlang C</i>

Neglecting variation in arrival intensity according to the number of lines already off-hook (which cannot generate calls) amounts to considering incoming traffic as a pure Poisson process, whose intensity does not depend on the past history of the system. It is valid only if each line is ‘on-hook’ for a small fraction of the time, which is generally true, and if there are enough lines so that the influence of an off-hook line can be neglected.

The Bernoulli and Engset models do not consider arriving traffic globally as a fixed intensity Poisson process, but instead consider the probability of having a new call seizure for each on-hook line as a Poisson process.

5.5.3 Dimensioning call servers in small networks

5.5.3.1 Overdimensionned network: Bernoulli distribution

This is a special case where there are as many channels as sources ($N = S$). It is also called a 'full availability system', because no client can be rejected or even wait.

We consider many (G) similar environments with S sources and N channels. If we take a snapshot at any time, some of those systems will have only one line busy, some of them will have two lines busy, and some up to S lines busy. We call $g(n)$ the number of systems with n busy lines (' n ' systems). We assume that if the group of systems is large, the number $g(n)$ of systems in each state ' n ' is steady.

We consider the evolution of these G systems over a very short time interval t , so short that there is a negligible probability of having more than one event, like a new call, or completion of an active call for each system: the probability of transiting from state ' n ' to state ' $n + 2$ ' or ' $n - 2$ ' for each system is negligible.

Let's calculate the number of new calls received during a very short interval t by $g(n)$ ' n ' systems, turning them into ' $n + 1$ ' systems. As discussed above, thus is $g(n) * (S - n) \lambda t$. Similarly, calls will be completed in ' $n + 1$ ' systems, turning them into ' n ' systems. In our brief interval t , there will be $g(n + 1) * (n + 1) * \mu t$ such occurrences.

The distribution of systems in each state is steady only if the flow of transitions in both directions is equal (Figure 5.24).

If we want distribution $g(n)$ to be steady, the number of ' n ' systems becoming ' $n + 1$ ' systems must equal the number of ' $n + 1$ ' systems becoming ' n ' systems during t ; therefore, we must have for each n :

$$0 = (n + 1) \mu t * g(n + 1) - g(n) * (S - n) \lambda t$$

In fact, $g(n)/G$ is the probability $P(n)$ of having n lines simultaneously busy and the exact value of t has no importance. Our equation becomes:

$$P(n + 1) = P(n) * (S - n) * \lambda / \mu(n + 1) \quad (\text{eq. } n+1)$$

$$P(n) = P(n - 1) * (S - n + 1) * \lambda / \mu(n) \quad (\text{eq. } n)$$

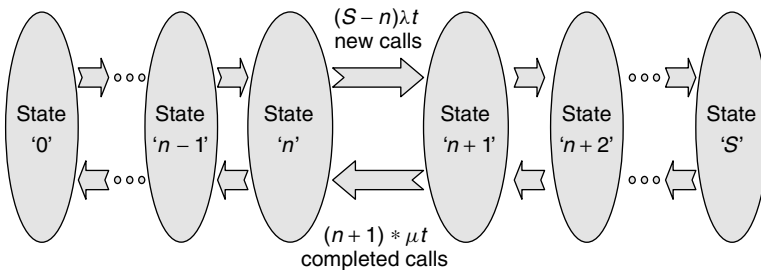


Figure 5.24 Transition model for the Bernoulli distribution.

This leads by recurrence to:

$$P(n) = \frac{S!}{n!(S-n)!} \left(\frac{\lambda}{\mu}\right)^n P(0) = C_S^n \times \left(\frac{\lambda}{\mu}\right)^n P(0)$$

Since the sum of $P(n)$ over all values of n equals 1, and we have:

$$\sum_{n=0}^N P(n) = \left(\frac{\lambda}{\mu} + 1\right)^S P(0)$$

we can find:

$$P(0) = \left(\frac{\mu}{\lambda + \mu}\right)^S = \left(\frac{1}{1 + \beta}\right)^S$$

where $\beta = \lambda/\mu$ is called the offered traffic per idle source:

$$\begin{aligned} P(n) &= C_S^n (\beta)^n \left(\frac{1}{1 + \beta}\right)^S = C_S^n \left(\frac{\beta}{1 + \beta}\right)^n \left(\frac{1}{1 + \beta}\right)^{S-n} \\ &= C_S^n \left(\frac{\beta}{1 + \beta}\right)^n \left(1 - \frac{\beta}{1 + \beta}\right)^{S-n} \end{aligned}$$

In the case where calls are never blocked, traffic a per source equals $\lambda/(\lambda + \mu)$ or $\beta/(1 + \beta)$ and we have:

$$P(n) = C_S^n \times a^n \times (1 - a)^{S-n}$$

Yes, this is the same binomial (Bernoulli) distribution as the one we encountered in the first section 5.1.2 when trying to calculate the probability of having n simultaneous active voice channels. If we consider calling lines as either ‘active’ with probability a or ‘idle’ with probability $1 - a$, and calculate the probability of having n from S lines active, then a denotes the proportion of time the line is busy, not the proportion of time it is sending voice packets when active.

The standard deviation of this distribution is $\sigma = \sqrt{Sa(1-a)}$ (note that it varies proportionally to the square root of total traffic $A = Sa$). Therefore, it is more efficient to process calls from a large number of lines: a large network-based Centrex switch serving 1,000 small enterprises will require a lot less simultaneous call capacity than the sum of 1,000 small IP PBXs! This will also be reflected in the results of Section 5.5.3.2.

5.5.3.2 The PBX model: X clients and N servers (the Engset distribution)

The full availability model is of little use in practice, because we want to use the smallest set of servers (PBX capacity) for offered traffic. We accept losing some calls if congestion occurs, but only a small fraction. We use the same definitions as above:

- S is number of sources.
- N is number of channels (now we have $N < S$).

- λ is call intensity per idle source.
- $1/\mu$ is mean holding time.
- $\beta = \lambda/\mu$.

In addition, let's call B the probability that a new call gets rejected (i.e., that N servers are used when the call arrives).

We calculate the new distribution $P(n)$ using the same method, by considering state ' n ' to ' $n + 1$ ' transitions as in equilibrium. When $n \leq S$ the equation set is the same as above:

$$P(n) = P(n-1) * (S - n + 1) * \lambda / \mu(n) \quad (\text{eq. } n)$$

Therefore, we also have:

$$P(n) = C_S^n \times \left(\frac{\lambda}{\mu}\right)^n P(0)$$

We must have:

$$\sum_{n=0}^N P(n) = 1$$

which leads to:

$$P(n) = \frac{C_S^n * \left(\frac{\lambda}{\mu}\right)^n}{\sum_{i=0}^N C_S^i * \left(\frac{\lambda}{\mu}\right)^i} = \frac{C_S^n * (\beta)^n}{\sum_{i=0}^N C_S^i * (\beta)^i} = E_{n,S}$$

This is simply the truncated form of the binomial distribution. In order to avoid computer overflows when evaluating the Engset distribution, the most efficient method is to use the following recursion formula:

$$\frac{1}{E_{i,S}(\beta)} = 1 + \frac{i}{\beta \cdot (S - i + 1)} \cdot \frac{1}{E_{i-1,S}(\beta)}$$

$$E_{0,S}(\beta) = 1$$

The time congestion probability (the proportion of time the system is blocked for new call attempts) is $P(S)$. The probability of call congestion B is smaller than the time probability of having S servers occupied. This is because average arriving traffic intensity gets smaller as the number of busy servers increases (i.e., fewer on-hook lines, see the $S - n$ coefficient in equilibrium equations), and therefore the arriving intensity in state $n = S$ is below average.

Call attempts arriving in state $n = S$ are blocked:

$$B_{N,S} = \frac{\text{Lost_calls}}{\text{Total_call_attempts}} = \frac{P(N) \cdot (S - N) \cdot \lambda}{\sum_{i=0}^N P(i) \cdot (S - i) \cdot \lambda}$$

$$= \frac{C_S^N \cdot \beta^N \cdot (S - N) \cdot \lambda}{\sum_{i=0}^N C_S^i \cdot \beta^i \cdot (S - i) \cdot \lambda} = \frac{C_{S-1}^N \cdot \beta^N}{\sum_{i=0}^N C_{S-1}^i \cdot \beta^i} = E_{N,S-1}$$

In other words, the probability that a new call attempt gets rejected equals the probability that remaining $S - 1$ sources fully occupy the N channels. This result, called the ‘arrival theorem’ is valid for any queuing system with a limited number of sources, even if congested calls are lost or delayed.

There is the following relation between $E_{N,S}$ and $B_{N,S}$:

$$E_{N,S} = \frac{S}{S - N} \cdot \frac{B_{N,S}}{1 + \beta(1 - B_{N,S})}$$

Carried traffic $A_c = S \cdot a_c$ can be evaluated by substituting $P(i)$ by its expression in the equilibrium equation:

$$\begin{aligned} A_c &= \sum_{i=1}^N i \cdot P(i) = \sum_{i=1}^N \beta \cdot (S - i + 1) \cdot P(i - 1) \\ &= \sum_{i=0}^{N-1} \beta \cdot (S - i) \cdot P(i) = \beta S(1 - P(N)) - \beta A_c + \beta N \cdot P(N) \\ A_c &= S a_c = \frac{\beta}{1 + \beta} \cdot [S - E_{N,S} \cdot (S - N)] = S - \frac{S}{1 + \beta \cdot (1 - B_{N,S})} \\ a_c &= 1 - \frac{1}{1 + \beta \cdot (1 - B_{N,S})} \end{aligned}$$

Usually, the problem is to find the blocking probability for a given number of servers N when offered traffic A_o is known, not when β is known. Here the definition of offered traffic varies from author to author. Some authors simply define offered traffic A as $A = A_c/(1 - B)$ or the equivalent $(A - A_c)/A = B$. In this case there is no easy analytical solution.

For very small values of B , it is possible to approximate $(1 - B)$ by 1. Another way of solving the problem is to draw (A, B) parametrically as a function of β for the given values B and N . The desired result is the value of B when $A = A_o$. This definition of offered traffic is used by most traffic calculators available on the Internet.

The ITU question 16 teletraffic group defines offered traffic as the traffic that would be carried in the total accessibility situation (no blocking). This is more logical, as with this definition offered traffic does not depend on the number of servers. With this more rigorous definition we have:

$$A = \frac{\lambda}{\lambda + \mu} = \frac{\beta}{1 + \beta}$$

With this definition it is easy to find β from A , and derive the time-blocking probability $P(N)$ and the call-blocking probability. With this definition, however, we no longer have

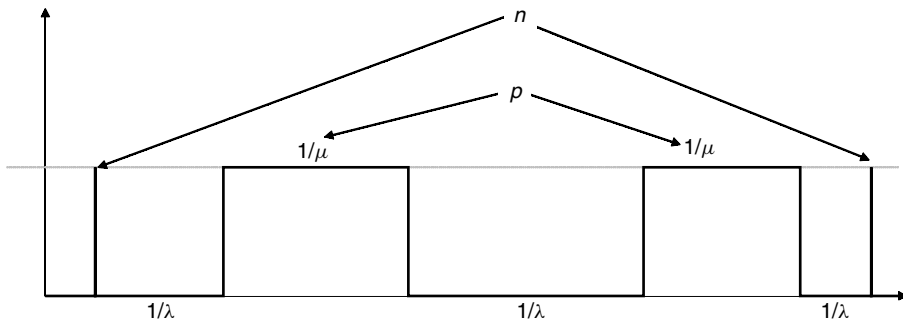


Figure 5.25 Traffic profile when some calls are rejected.

$(A - A_c)/A = B$, but instead:

$$C = \frac{A - A_c}{A} = \frac{B}{1 + \beta(1 - B)}$$

where C is called traffic congestion (representing approximately the loss of revenue for a service provider). While this difference seems counter-intuitive, it can be understood by considering real traffic with lost calls for a source (Figure 5.25).

The number of active periods is p , the number of failed call attempts is n , and we have $B = n/(n + p)$. Over a period of time T , the line transmits p/μ seconds of traffic. Therefore, we have:

$$a_c = \frac{P/\mu}{T} = \frac{P/\mu}{p(1/\mu + 1/\lambda) + n/\lambda} = \frac{\beta p}{p(1 + \beta) + n} = \frac{\beta(1 - B)}{(1 - B)(1 + \beta) + B}$$

$$C = \frac{a - a_c}{a} = 1 - a_c \cdot \frac{\beta + 1}{\beta} = \frac{B}{1 + \beta(1 - B)}$$

If we consider $C = B$, we are overlooking the fact that when a call cannot complete it does not spend any time active, and therefore on average the next call arrives after $1/\lambda$, not after $1/\lambda + 1/\mu$, which explains why we have $C < B$.

5.5.4 Dimensioning call servers in large networks

5.5.4.1 Full availability model

In a large network, the total number of end points will usually be unknown; only the total traffic A to be processed will be available. Although S is very large and unknown, and a is small and unknown, we know that $S \cdot a = A$, the total traffic (in Erlangs) generated by the sources. Therefore, we cannot use the Bernoulli distribution as presented above, but

the Bernoulli distribution $P(n)$ can be simplified³ into a Poisson law:

$$P(n) = \frac{A^n}{n!} e^{-A}$$

and we approximate $\sigma = \sqrt{A}$, which is slightly larger than the exact value.

Note that the average of both the Bernoulli and the Poisson laws is obviously A (the amount of arriving traffic, since there is no congestion). However, the variance of the Poisson law is higher than the variance of the Bernoulli distribution: so when using the Poisson law to dimension a network with a large, unknown number of clients, we obtain a worst case estimate.

5.5.4.2 The Erlang B formula

We can derive the Erlang B law from the Engset distribution if each line is not very active, there are many lines, and we only know the total traffic ($Sa \rightarrow A$). In this approximation the variation of incoming traffic intensity with the number of off-hook lines can be ignored ('infinite' number of sources) and the equilibrium equation converges to:

$$P(n) = Sa/n * P(n-1)$$

and we have:

$$P(n) = \frac{\frac{A^n}{n!}}{\sum_{i=0}^N \frac{A^i}{i!}}$$

This is a truncated Poisson distribution. The probability of call loss is:

$$B = P(S) = E_{1,S}(A)$$

This result is the first Erlang law (also called the Erlang B law). It allows calculation of the number N of servers (e.g., external lines) that must be installed to ensure a given rejection rate B , if the excess calls are *immediately* rejected. This is similar to the Poisson law, with a scaling factor.

Since the Erlang law is obtained simply by considering that arrival intensity does not depend on the number of clients already 'off-hook', time congestion probability $P(S)$ is now equal to call loss probability B .

The Erlang B law is valid only if the law of arrival calls is a Poisson process and if congested calls are cleared, but makes no assumption on call duration distribution, which can be exponentially distributed (Poisson law) or fixed.

³ Note that this limit of the binomial distribution is different from that of the Moivre approximation used in Section 1.2: it is obtained when S increases, but with a constant value for a .

In order to compute the Erlang B law without overflow errors, the following recursive method can be used:

$$\frac{1}{E_{1,N}(A)} = 1 + \frac{N}{A} * \frac{1}{E_{1,N-1}(A)} \quad \text{with } E_{1,0}(A) = 1$$

It is always possible to compute $P(N)$ with increasing values of N until we reach the desired B , but there is also a simple approximation that is useful to make quick order-of-magnitude estimations (known as the ‘Rigault rule’): if the desired B is 10^{-k} , then $N_{\max} \approx A + k\sqrt{A}$.

5.5.4.3 Model for a limited set of servers and phone lines that can wait for a server (Erlang C formula)

A call server that can process a maximum of N calls simultaneously, instead of dropping all calls that arrive while the servers are busy, can decide to queue the incoming calls for a little while. This model can also be used for call centers.

Let’s now model the traffic generated by a single phone line with an average pick-up frequency per line of λ , an average duration $1/\mu$, and an average waiting time of w , as illustrated in Figure 5.26.

A line is served immediately it is picked up, if possible, but if there are already N lines busy then the user needs to wait to be served. The waiting queue size is infinite and the users are served in FIFO mode.

The method used to obtain the probability of each state is similar to that used previously. In addition to the N states, corresponding to the number of channels used, there are additional states corresponding to all N channels used, and 1, 2, 3, ... lines waiting for service in the queue. For these states we use the notation ‘ $N + 1$ ’, ‘ $N + 2$ ’, ‘ $N + 3$ ’, etc. (Figure 5.27). In all waiting states only N channels process calls and, therefore, call-processing intensity remains fixed at $N\mu t$.

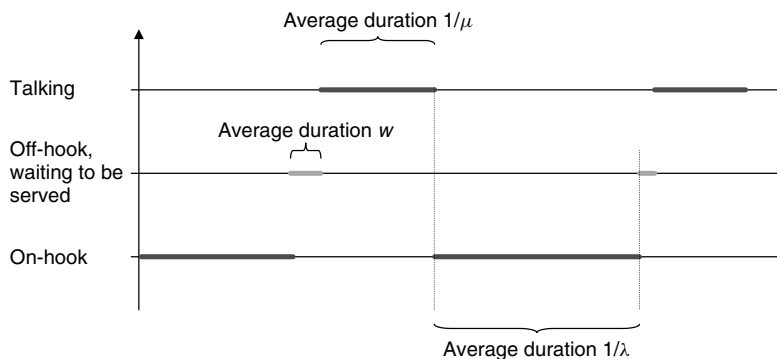


Figure 5.26 Call states for the Erlang C model.

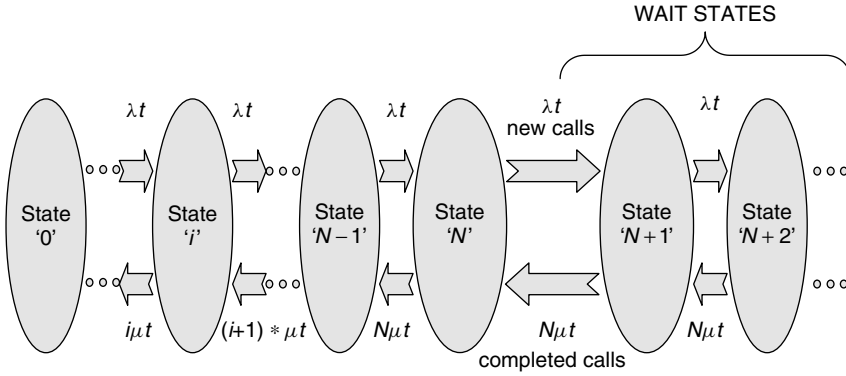


Figure 5.27 State transitions for the Erlang C model.

Under the same approximation conditions as for the Erlang B law (i.e., many lines served, each with relatively little traffic), the offered traffic is $A = \lambda/\mu$ and we obtain:

$$P(j < N) = \frac{A^j}{j!} P(0) = \frac{\frac{A^j}{j!}}{\left(1 + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \dots + \frac{A^N}{N!}\right) + \frac{A^N}{N!} \times \left(\frac{A}{N-A}\right)}$$

$$P(N+j) = \left(\frac{A}{N}\right)^j P(N) = \frac{\left(\frac{A}{N}\right)^j \cdot \frac{A^N}{N!}}{\left(1 + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \dots + \frac{A^N}{N!}\right) + \frac{A^N}{N!} \times \left(\frac{A}{N-A}\right)}$$

The time probability of waiting called $E_{2,N}(A)$ can be obtained summing all $P(N+j)$. It can be shown that in the case of a Poisson arrival process this is also the call-waiting probability (the PASTA, or Poisson Arrivals See Time Averages, property). This formula is the Erlang C law:

$$E_{2,N}(A) = \sum_{j=0}^{\infty} P(N+j) = \frac{N}{N-A} P(N) = \frac{\left(\frac{N}{N-A}\right) \frac{A^N}{N!}}{1 + A + \frac{A^2}{2!} \dots + \frac{A^N}{N!} + \frac{A^N}{N!} \left(\frac{A}{N-A}\right)}$$

It is also interesting to note that the Erlang C law can be expressed as a function of the Erlang B law:

$$E_{2,N}(A) = \frac{N E_{1,N}(A)}{(N-A) + A E_{1,N}(A)}$$

The probability of waiting more than t_w seconds can be obtained because, in each state ' $N+j$ ', this probability is the same as having fewer than j departures in t_w seconds, if

the service discipline is FIFO. For a Poisson service process, this is:

$$\sum_{r=0}^j \frac{\left(\frac{N}{\mu} t_w\right)^r}{r!} e^{-\frac{N}{\mu} t_w}$$

The weighted sum on all waiting states gives:

$$P(w > t_w) = E_{2,N}(A) \cdot e^{-\frac{(N-A)t_w}{\mu}}$$

The average number of calls waiting in the queue, also called waiting time traffic, is:

$$A_w = \sum_{j=0}^{\infty} j P(N+j) = \frac{AN}{(N-A)^2} P(N) = \frac{A}{N-A} E_{2,N}(A)$$

and Little's theorem (valid in most queuing systems) tells us that the average waiting time t_{wait1} equals the average queue length divided by the arrival intensity:

$$t_{wait1} = \frac{A}{(N-A) \cdot \lambda} E_{2,N} = \frac{1}{\mu(N-A)} E_{2,N}$$

This is an average for all calls, including those that do not wait. The average for calls that wait t_{wait2} can be obtained by writing total waiting time as:

$$\begin{aligned} Total_wait_time &= t_{wait1} * Total_calls \\ &= t_{wait2} * Waiting_calls = t_{wait2} * Total_calls * E_{2,N}(A) \end{aligned}$$

Therefore the average waiting time for calls that effectively wait is simply:

$$t_{wait2} = \frac{1}{\mu(N-A)}$$

5.6 Conclusion

Systems with few users need to be dimensioned for peak values; systems with many users need to be dimensioned for average values. This allows network use to become more efficient as the number of users increases.

In a large TDM telephony system, the network can be dimensioned for slightly more than the average number of simultaneous calls. It is possible to be even more efficient with IP telephony systems, because audio coders with voice activity detection make it possible to perform some statistical multiplexing between active channels and idle channels: when there are many channels, the link needs to be dimensioned for slightly more than the average bitrate of the coder taking into account the activity rate of the conversation. Another potential advantage of VoIP is that it is able to use many low-bitrate coders, not

just the 64-kbit/s G.711 codec used in traditional telephony. Some of these low bitrate coders have a peak rate of less than 10 kbits/s.

With an additional level of multiplexing (voice activity detection) and more efficient coders, it would seem that IP telephony is far more efficient than the PSTN. In fact, this comment must be viewed with skepticism. The tremendous overheads of RTP+UDP+IP+the physical layer ‘eat’ much of what was gained with compression and VAD (many systems do not even have VAD). Overheads can be reduced at the edge of the network using compressed RTP (cRTP), but within the core network overhead reduction requires stacking many frames per RTP packet and this degrades delay performance.

Overall, it seems that claiming that *the* advantage of IP telephony is its bandwidth efficiency is misleading. Many other techniques can achieve the same or even better efficiency (e.g., think of DCME equipment used on transatlantic lines). The key advantages of VoIP lay elsewhere:

- Much better any-to-any connectivity in large networks (compared with frame relay or ATM where too many ‘virtual channels’ are required).
- Companies and service providers can reduce wiring costs, as all communications are multiplexed on the data network.
- Carriers can merge all communications on a single backbone, reducing maintenance and operations costs.
- VoIP voice-switching equipment does not need to route media streams, unlike TDM switches. Because of this they can be located much farther away from the end-customer, reducing the need for local POPs and reducing both CAPEX and ongoing OPEX.
- The commoditization of hardware used for service platforms, because the hard, real-time requirements of TDM are no longer required in VoIP (the media stream bypasses the service equipment); so, standard computers can be used instead of purpose-built TDM systems.

With the ever-increasing availability of broadband connections at an affordable price and the constant lowering of the cost of long-distance bandwidth for carriers, service providers now have a tendency to focus on services, instead of spending a lot of time optimizing the need for network resources. More and more carriers are now deploying voice over IP networks with the G.711 voice coder, because they find the gain in bandwidth is not worth the reduction in perceived voice quality. Service providers may even, in the near future, introduce broadband coders in their networks. VoIP started with a focus on cheaper voice and prepaid telephony, but it is now clear that the direction is toward the high-end market, advanced services, and a richer multimedia experience.

5.7 References

- R. Guerin, H. Hamadi, and M. Naghshineh (1991). Equivalent capacity and its application to bandwidth allocation. *IEEE Journal on Selected Areas in Communications*, 9(7), September.
- L. Kleinrock. *Queueing Systems*, Wiley Interscience (ISBN 0 471 49110 1).

6

IP Multicast Routing

Multicast is a real-time, network-level information distribution technology. It does not need any central server to distribute information at the application level. Like many other IP technologies, multicast was originally designed in a university. It grew from an overlay network called the mBone (Figure 6.1) which is built on top of regular Internet links. Today, multicast seems to have reached a critical level of maturity which makes it capable of supporting commercial services such as television broadcast over IP, real-time financial data distribution, and videoconferencing. These applications will soon trigger a need for IP multicast-enabled intranets.

6.1 Introduction

The chapter explains the advantages of network-level data distribution and describes the protocols currently used, and their limits. There is also a description of some widely used applications. As multicast is an evolving technology, we also cover the current work at IETF regarding group address allocation and multicast interdomain routing.

6.2 When to use multicast routing

6.2.1 A real-time technology

There are already many techniques that are used to distribute information to many recipients on the Internet. They were developed to solve specific problems that were encountered during the development of the Internet:

- The domain name system (DNS) is used to distribute the mapping of domain names to IP addresses. DNS defines an efficient caching and replication mechanism for use between DNS servers.

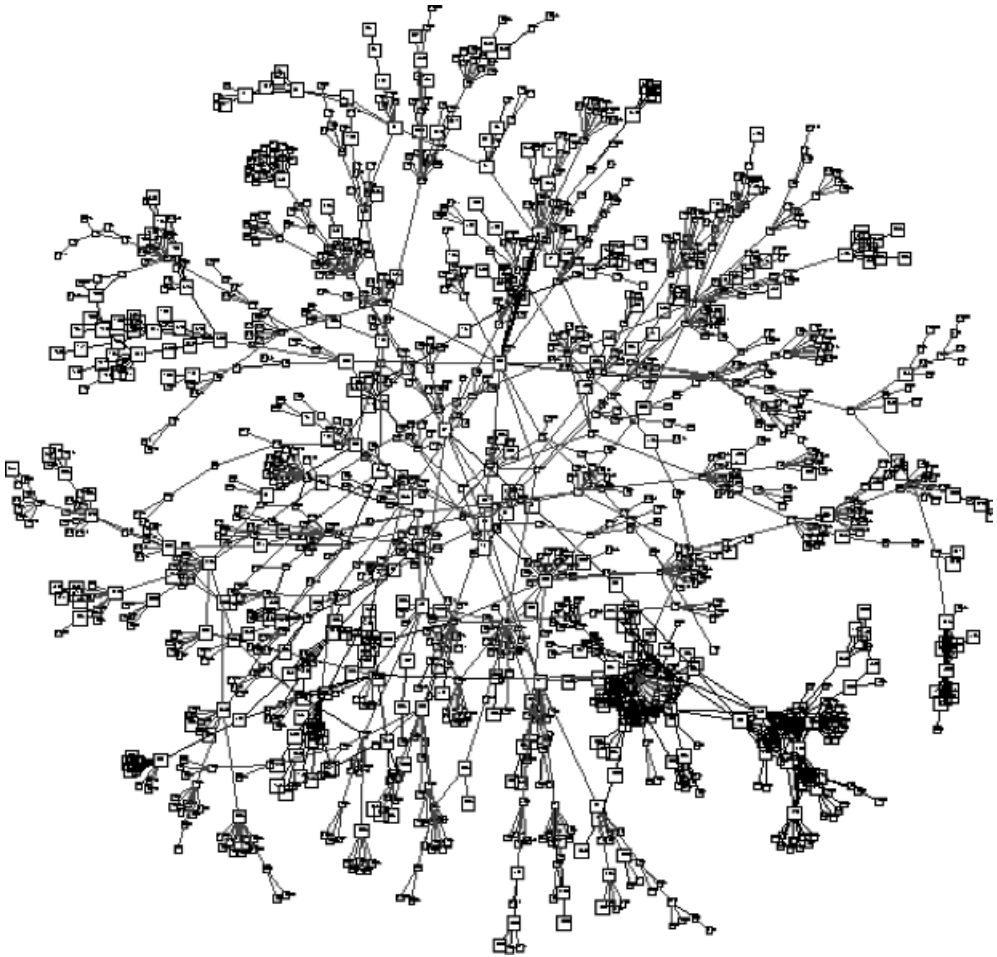


Figure 6.1 The mBone as of August 5, 1996. Reproduced from the University of California at Berkeley.

- NNTP, the Network News Transfer Protocol, is used to send newsgroup messages to news servers worldwide.
- IRC, the Internet Relay Chat, is a text chat protocol optimized to immediately send any sentence typed by any participant of a forum to all other relevant chat servers, which in turn send this sentence to all members of the forum that they host.
- Even HTTP, the protocol used to transfer web documents, was designed to let cache servers know how long they can keep a page in memory, in order to minimize unnecessary network traffic.

These techniques are very efficient at what they do, but they share a common characteristic: they are not real time. Because they duplicate and distribute information at the application

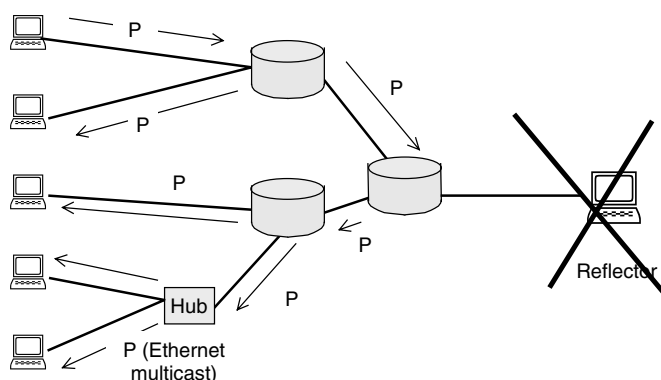


Figure 6.4 Multicast optimizes the distribution of information.

reflector must be able to carry all the generated traffic, which is proportional to the number of clients. A more scalable solution would ideally send only one copy of each packet over each link and would not need a special machine to handle all the work: this is exactly what IP multicast is doing, as illustrated in Figure 6.4. The drawing also illustrates that IP multicast is supported natively by some transport networks, in this case an Ethernet hub.

6.2.3 Resource discovery

Another application for multicast is the discovery of resources on a network. Many applications today rely on broadcasts (sending information to all hosts on a network of computers linked together by means of a network layer like Ethernet) of an interrogation message to find network resources. The Windows[®] operating system is one of them. Broadcast is fine when just a few workstations share a small LAN, but in bigger networks where hundreds of workstations are connected using hubs and switches it becomes a real problem. Because network managers want to avoid broadcast storms as much as possible, they usually configure their routers to not forward broadcasts across subnets. This limits the practical usefulness of broadcast discovery to just the subnet of the broadcasting host.

Multicast is one possible solution to these limitations of broadcast; there are other useful approaches (e.g., the IEEE 802.1 WG defined the notion of VLANs for distributed working groups). Multicast is a way of distributing information to a group which can easily span several subnets and yet reach only the hosts that have requested to be members of the group. Moreover, multicast can be configured to carry out expanding ring searches, so a host can query its immediate neighborhood for a resource without flooding the universe in the first place. The H.323 protocol uses this type of resource discovery to find gatekeepers on the network.

6.3 The multicast framework

6.3.1 Multicast address, multicast group

A ‘multicast IP address’ format has been introduced in IPv4 and IPv6 to support multicast applications, in addition to the existing unicast (pointing to a single destination) and

broadcast (pointing to all hosts on a subnet) addresses. Multicast addresses should not be confused with anycast addresses, which have been added in IPv6: a packet sent to an anycast address must reach one and only one host in a group, while a packet sent to a multicast address must reach all members of the group identified by the multicast address.

In IPv4, a multicast address is a class D address, which ranges from 224.0.0.0 to 239.255.255.255 (all addresses starting with the bit pattern '1110'). Addresses 224.0.0.0 to 224.0.0.255 are reserved for multicast-routing protocols. With the remaining addresses, combined with a port number (from 1,024 to 65 535) in the case of UDP multicast, there are still more than 16 000 billion possibilities for distinct multicast conversations. However, only the IP address part is relevant when building the distribution tree, so applications using distinct ports must share the same distribution trees. In IPv6, multicast addresses will have a high-order octet equal to FF.

There is the same difference between a regular email address and a mailing list address as between a unicast address and a multicast address (Figure 6.5). Clients who subscribe to a particular multicast address will receive all datagrams sent with this multicast address in the destination address field.

A multicast group is a set of hosts that subscribed to the same multicast address. The subscription is done using a protocol called IGMP (Internet Group Membership Protocol). These hosts are called the group members. A group is completely dynamic: at any time a machine can leave or join a group. There is no restriction to the number or location of members in the group.

Note: A client is not required to be a member of a group to send a message to its members. In fact, there is only one significant difference between a mailing list and a multicast group. In the first case, the complete list of members is known to a central server. For multicast, the routers in the network only know if they have at least one member on each interface, without knowing who the members are.

Since groups are completely dynamic, multicast addresses need to be obtained dynamically. The main issue is to choose an address that is not already in use. On the mBone, the addresses already in use can be obtained via the SDR application (see Section 6.7.3.2), but some applications simply choose a random address. The second issue is to make this address known to potential listeners: here again it is possible to use SDR (this has the

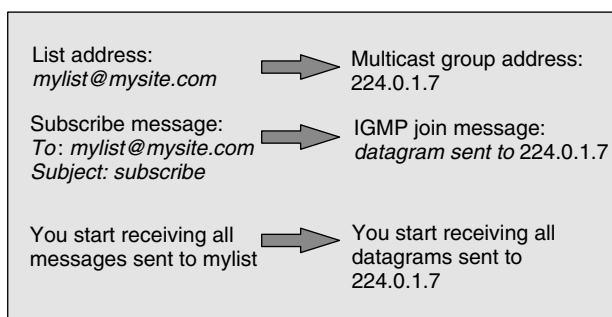


Figure 6.5 Mailing list address versus multicast group address.

advantage of letting everyone know that you are using this address), but a simple web page also serves the purpose if it is known to the potential audience.

A permanent group is just a group with a well-known address (registered by the **Internet Assigned Numbers Authority for IANA**) which is used for a particular application. It does not imply that there is some permanent member in that group. Table 6.1 lists some well-known groups.

Note: TCP cannot be used for multicast communications, and multicast datagrams have to be standard UDP or RAW datagrams, which are delivered to group members with no guarantee. Other reliable transmission mechanisms can be implemented on top of UDP.

6.3.2 Multicast on ethernet

In addition to reserved class D IP addresses, the IANA owns a block of Ethernet addresses reserved for IP multicast, which in hexadecimal begins with 01:00:5E (the first byte of any Ethernet address must be 01 to specify a multicast address). The IANA allocates half of this block for mapping class D IP multicast addresses to IEEE-802 multicast addresses; so, the Ethernet addresses corresponding to IP multicasting are in the range 01:00:5E:00:00:00 through 01:00:5E:7f:ff:ff.

There is no one-to-one mapping. The reason for this can be explained by a bit of history: when Steve Deering first designed IP multicast, he figured out that he would need to buy 16 blocks of 24 bits from IEEE to map all IP multicast addresses. Each block was worth \$2,000, so he was only allowed to use half of a 24-bit block, which accounts for the 23 bits we have today.

This allocation allows for 23 bits in the Ethernet address to correspond to the IP multicast group ID. The mapping places the low-order 23 bits of the multicast group ID into these 23 bits of the Ethernet address (Figure 6.6). Since the upper 5 bits of the multicast address are ignored in this mapping, there is no one-to-one relationship: 32 different multicast group IDs map to each Ethernet address.

Because there is no one-to-one mapping between Ethernet and IP multicast addresses, an Ethernet card can receive and forward to the device driver wrong packets. The device driver or the IP stack of the host must filter out these datagrams by checking the IP

Table 6.1 Some well-known multicast address groups

All systems on this subnet	224.0.0.1
All routers on this subnet	224.0.0.2
All DVMRP routers	224.0.0.4
All MOSPF routers	224.0.0.5
Routing Information Protocol (RIP)—Version 2	224.0.0.9
Network Time Protocol (NTP)	224.0.1.1
Audio news	224.0.1.7
IETF audio	224.0.1.11
IETF video	224.0.1.12

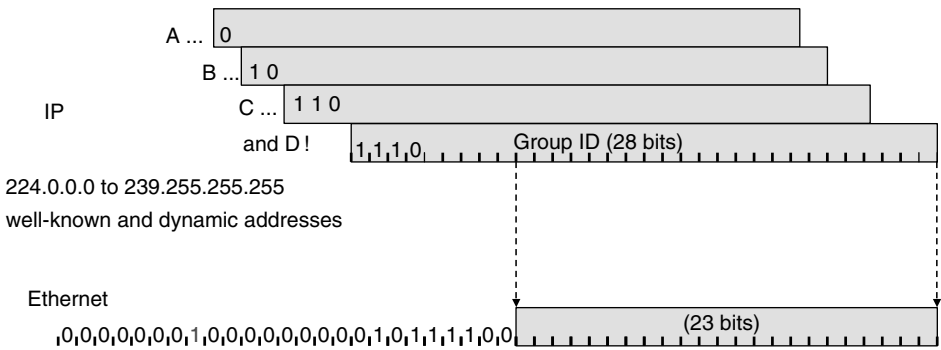


Figure 6.6 Mapping of IP multicast addresses to Ethernet multicast addresses.

destination address. The receiving processes must notify their IP layers that they want to receive datagrams destined for a given multicast address, and the device driver must enable reception of these multicast frames. This process is handled by joining a multicast group.

IP multicasting on a single physical Ethernet network is simple. The sending process specifies a destination IP address that is a multicast address and then the device driver converts this address to the corresponding Ethernet address and sends it.

6.3.3 Group membership protocol

6.3.3.1 IGMPv1

The Internet Group Membership Protocol (IGMP) version 1 is specified in RFC 1112. In the same way as a special form of email is sent to the list server to subscribe to the list, a host sends a group membership protocol datagram to the group IP multicast address in order to become a member of a multicast group. IGMP has been assigned protocol number 2 (RFC 1700).

When a host first subscribes to a multicast group, a couple of IGMP reports are sent to the group address to which the host subscribes with a TTL of 1 (Figure 6.7). Since multicast routers promiscuously receive all multicast traffic (the network interface forwards all packets to the device driver), they get informed of the new member. Because of the TTL, an IGMP message is never forwarded out of the subnet.

On each link, a multicast router is elected to be the ‘querier’ and periodically (every minute, typically) sends an IGMP query message to the all-hosts group (224.0.0.1) with a TTL of 1 (Figure 6.8). All hosts on directly connected subnets are supposed to issue an answer along with an IGMP report sent to each group address to which it belongs. To avoid a synchronized storm of messages, these reports are sent after a random delay. When a host hears a report for a group and is also a member of that group, it resets the timer and keeps silent to avoid duplicate messages. The router will consider that there is no member left for group *G* on a link if it doesn’t hear reports for group *G* after several queries on this link.

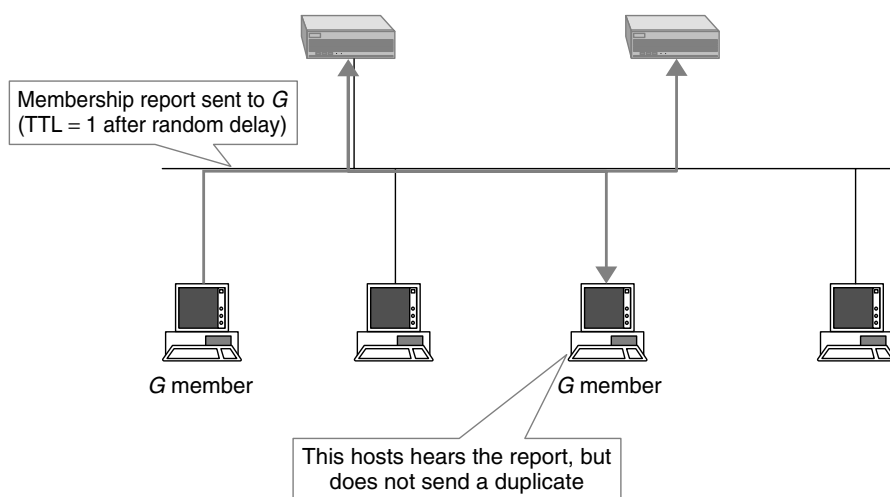


Figure 6.7 Avoiding unnecessary membership reports. When first joining a group, two reports or more are sent without waiting for a query.

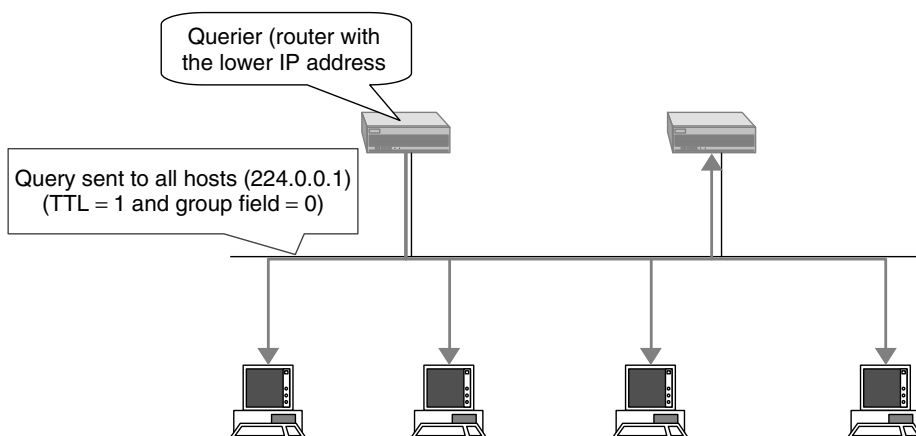


Figure 6.8 Periodic group membership queries by the querier router. Queries are sent every 60–90 s.

In the IGMPv1 format (Figure 6.9), message type 1 is used for queries and message type 2 is used for reports. The group address is either the multicast group concerned by the report or 0 in queries.

Note: IGMP only operates over broadcast LANs or point-to-point links, but there are some ways to extend the subscription mechanism over NBMA (non-broadcast, multiple access) networks, the ‘MARS’ protocol is an example of such a solution over ATM networks.

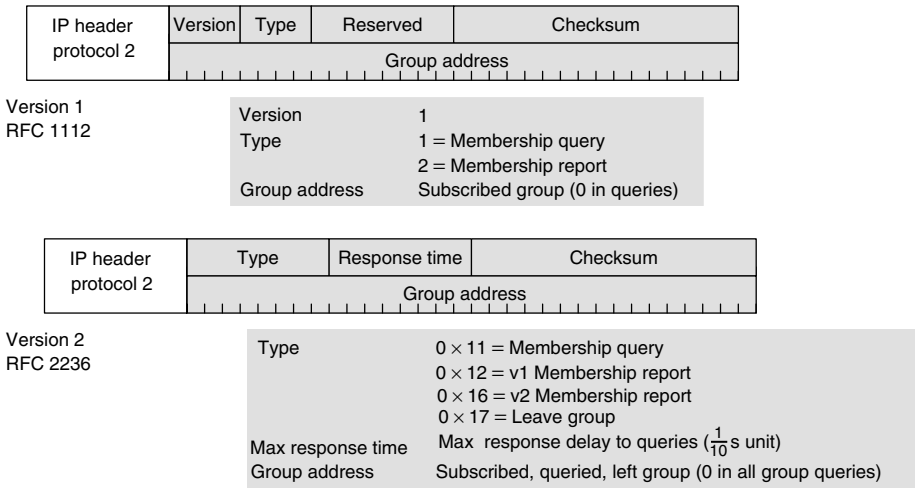


Figure 6.9 IGMPv1 and IGMPv2 message format.

6.3.3.2 IGMPv2

In IGMPv1, a router considers a group has no members left if it does not receive IGMP reports addressed to the group after a number of queries. In the meantime it will keep forwarding useless and bandwidth-consuming datagrams. In IGMPv2, an additional ‘leave group’ message has been defined to reduce the latency of hosts leaving a group (Figure 6.9). IGMPv2 is specified by RFC 2236 and is backward-compatible with v1.

The message fields ‘type’ and ‘version’ have been merged into a new 8-bit-type field (0x11 membership query, 0x12 v1 membership report, 0x16 v2 membership report, 0x17 leave group). The group address now indicates either the group being queried or reported to, or the one that has left. It is left to 0 to query all groups.

The reserved space has been allocated to indicate a maximal response delay in tenths of a second. The ‘leave’ message for a group is sent by a leaving host only if this host is the last one to have effectively sent a report membership for that group (otherwise it knows that there still are other members on the LAN). The querier router then sends a couple of group-specific queries with a small max response time to check no one else is still a member. If no report is heard for the group, then the router considers there are no more members on the LAN.

The querier election for IGMPv2 is very simple: initially all routers send queries and then only the router with the smallest IP address keeps sending queries. If the other routers do not hear queries for some time they restart the election process.

6.3.3.3 IGMPv3

IGMPv3, defined in RFC 3376, adds source selection possibilities, such as listening to some sources only or to all but a set of unwanted sources. This can be used, for instance,

to exclude from large conferences some users who send background noise (e.g., ones who do not know how to switch off their microphones). This also helps to prevent ‘denial of service’ attacks where the hacker sends a stream conflicting with the original session on the same multicast group and port. Because of this, some IGMP query and report messages have been extended to include a list of sources and a new IGMPv3 report type (0x22) has been introduced.

6.4 Controlling scope in multicast applications

6.4.1 Scope versus initial TTL

Like any other IP packet, a multicast datagram has a **TTL** (time to live) field. The TTL is decremented at each hop. When the TTL reaches 0, the packet is discarded by routers. For a unicast packet, this TTL is always set to a rather high value (127, typically) and is just used to prevent routing loops. The TTL field of a multicast datagram is also decremented at each multicast router. But, in addition of preventing routing loops, it is also an indication of how large the scope of the datagram is. If the IP multicast sender is considered to be like a radio station, the initial value of TTL defines the power of the emitter. The larger the TTL, the larger the range that can be reached (Figure 6.10). Therefore, multicast datagrams are usually sent with a small initial TTL.

The TTL can therefore be used as some basic form of ‘power control’ for a multicast session. A multicast session sent using a TTL of 2 can only span a disk centered on the sender with a diameter of 4 routers. Increasing the TTL to 6 would expand this diameter to 12 multicast routers. The broadcast area depends on the source.

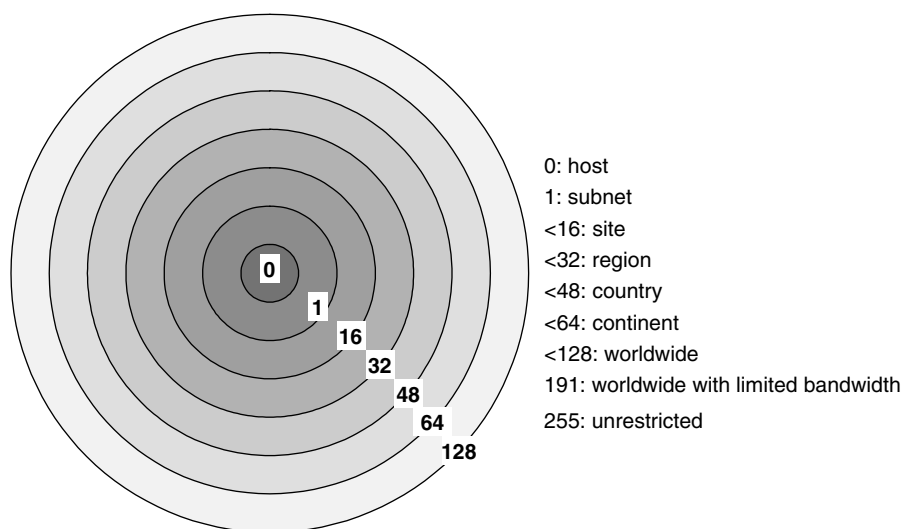


Figure 6.10 Classic TTL conventions.

6.4.2 TTL threshold

The TTL can also be used to set a virtual administrative boundary to a domain which does not depend on the source. All multicast interfaces can be configured to only forward packets having a TTL greater than a preset value (Figure 6.11). If an administrative domain can be approximately defined by a disk of diameter D , then setting the minimal forwarding threshold of all edge routers higher than D will prevent all sessions originating in the domain with a TTL of D to propagate to the outside world. Such sessions with an initial TTL of D will cover the whole domain but stay within the boundary of the edge routers.

This scheme also applies to nested domains (e.g., an internal subdomain could be configured with a threshold of 16 and the parent domain would then have a TTL of 32).

This method of limiting the scope of a multicast broadcast using TTL has a serious limitation: it does not allow administrative domains to overlap. For instance let us take the case of a company that has an engineering department A and an accounting department B, two bookkeepers are in charge of the engineering department and belong to both domains. We want to be able to make engineering-only conferences, accounting-only conferences, and company-wide conferences from any desktop in the relevant domains.

In the set-up shown in Figure 6.12 a conference sent from domain A with a TTL of 16 will stay in domain A. A conference with a TTL of 32 will be company-wide. But

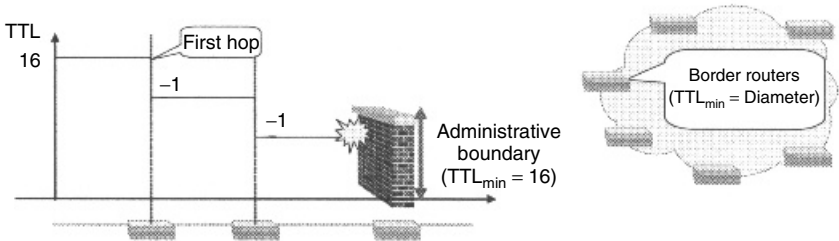


Figure 6.11 Using a TTL threshold to restrict multicast packets to a given domain.

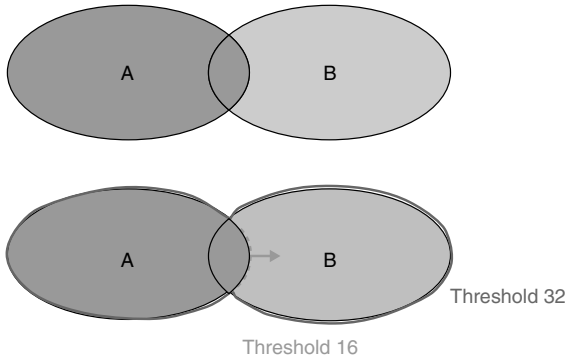


Figure 6.12 TTL threshold cannot be used with overlapping domains.

how can we make a conference for domain B only? If we set the outgoing threshold of the remaining common interfaces (left) to $X > 16$, then it will be impossible to initiate an 'A-only' conference from the bookkeepers' desktops (an initial TTL ≤ 16 would stay in the intersection domain, an initial TTL > 16 would leak in domain B). A threshold X below 16 creates the same impossibility for B-only conferences.

The multicast address range 239.0.0.0 to 239.255.255.255 (administratively scoped addresses) has been reserved to allow administrators to have better control over the scope of a session. Administrators can now configure all edge routers to not forward some addresses in this range. All sessions sent using a multicast address in this range will stay within the domain, regardless of the initial TTL. Overlapping multicast domains can now be configured simply by using different administratively scoped addresses in each of the domains.

Administrative scope is bidirectional: it prevents all 239.x.x.x traffic from getting out and getting in. This is useful since many site administrators on the mBone forget to set the administrative scope and still use software that is set to send 239.x.x.x datagrams.

6.5 Building the multicast delivery tree

With IP multicast, routers are responsible for duplicating the packets and sending them to appropriate interfaces. But, which interfaces are they? In fact the construction of the multicast delivery tree is the most complex issue of the multicast technology. Several techniques can be used, the most common are discussed below.

In the following text we will call a 'source router' any router directly connected to a subnetwork with an active source station.

6.5.1 Flooding and spanning tree

The simplest way to send a packet to every member of a group is flooding. In this technique each router of the IP network replicates every inbound multicast packet to all interfaces except the inbound interface. If the same packet arrives more than once, it is discarded. This is simple and robust (hence its use in some military networks), but clearly not scalable.

An improvement of the flooding algorithm is to select just a subset of Internet routers, but a subset that can still reach any destination. This subset should form a 'spanning tree' of interconnected routers, in which two distinct routers are interconnected by one and only one active path (Figure 6.13). This topology ensures there will be no routing loop, making it unnecessary to detect duplicate packets and making flooding much more efficient. Unfortunately, it is computationally difficult to build a spanning tree for large networks. There are two main types of spanning trees: shared trees and source-rooted trees.

6.5.2 Shared trees

Shared tree techniques use only one spanning tree for the group, independently of the source. A simple way to build a common spanning tree is to choose a 'rendezvous' point. Then all routers willing to receive the datagrams sent to the group send a message toward

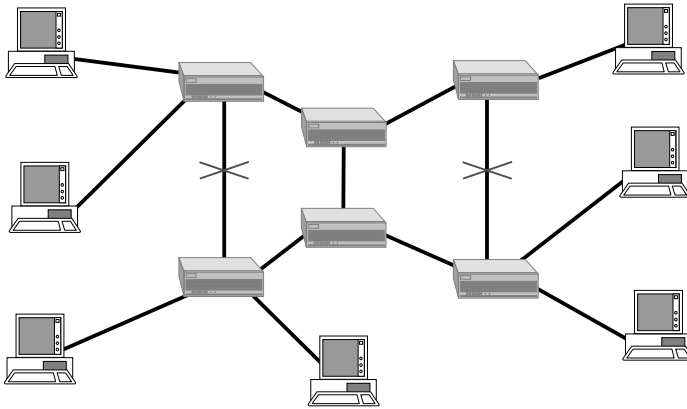


Figure 6.13 Spanning tree (there is exactly one path between any pair of nodes).

the rendezvous point, and each multicast router seeing this message on its way marks the interface from which it arrived and the outgoing interface. Now, any multicast datagram received at the outgoing interface will be copied to all other marked interfaces.

For a source router, sending a datagram to the group is just a matter of sending an encapsulated datagram to the rendezvous point, which unwraps it and forwards a copy to all of its marked interfaces.

6.5.3 Source-based trees

Some algorithms build a different tree for each source router. When a host sends a datagram to the group, the datagram will be duplicated according to the spanning tree rooted at the host's router (Figure 6.14). This leads to more efficient paths and shorter delivery delays.

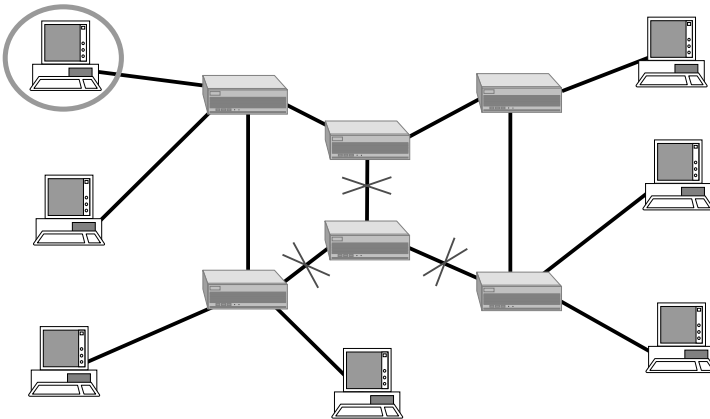


Figure 6.14 Source-based tree.

6.5.3.1 Reverse path broadcasting and truncated reverse path broadcasting

RPB (reverse path broadcasting) is a technique used to build source-based spanning trees. For each source, if the packet arrives on the link that the router believes to be the shortest way back toward the source (this information is derived from the protocol's own routing table in the case of **DVMRP** or from the unicast-routing table in the case of **PIM**), then the router duplicates the packet and forwards it to every interface except the originating one. Otherwise (i.e., if the packet comes from a link that is not the shortest way back to the source), the packet is dropped (Figure 6.15).

The algorithm in Figure 6.15 has one main limitation: it includes all routers and subnets in the tree, even if some of them are not part of the destination multicast group.

A possible enhancement of RPB is **truncated reverse path broadcasting (TRPB)**: here routers use the information obtained with IGMP to avoid sending multicast datagrams to leaf subnets in which no host is a member of the destination multicast group. However, the delivery tree between routers still makes no use of IGMP information, even though some parts of the tree might be useless.

DVMRPv1 (Distance Vector Multicast Routing Protocol), the original mBone-routing protocol, used the TRPB forwarding algorithm. The DVMRP multicast-routing protocol is very similar to the RIP unicast-routing protocol, except that it tracks distances to the source, not the destination.

6.5.3.2 Reverse path multicasting

RPM builds source-based trees that span only subnets with group members and builds routers along the shortest path to subnets with group members. The first packet is forwarded using the TRPB algorithm, but if edge routers see that none of their leaf subnets is a member of the destination group, they send a 'prune' message to the parent router.

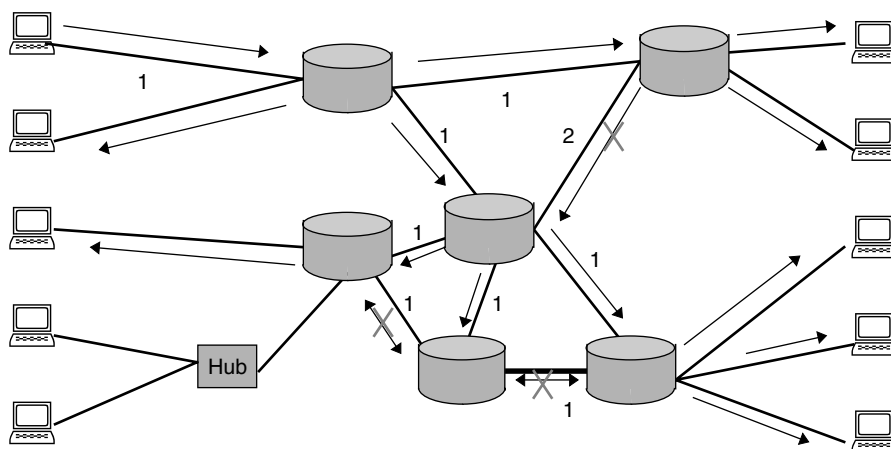


Figure 6.15 Distribution tree with RPB.

The parent router stores this information and disables this child interface for this source and this group. If all child interfaces are disabled for a given source and group, then this router itself sends a prune message upstream.

In order to allow dynamic group expansion, prune information has a limited lifetime, and therefore the network is periodically flooded again with TRPB. RPM is a big improvement over simple TRPB, but it requires routers to store a lot of prune information (for each active [source, group] pair) and periodic flooding wastes some bandwidth. RPM is well suited for networks with a large proportion of edge subnets which have members of multicast groups: it is a ‘dense-mode’ multicast-routing algorithm.

6.6 Multicast-routing protocols

6.6.1 Dense- and sparse-mode protocols

Multicast-routing techniques fall in two broad categories: sparse-mode protocols and dense-mode protocols. Sparse-mode protocols are optimized for large networks where only a small portion of all connected hosts are members of each group. Dense-mode protocols are optimized for networks where most hosts are members of active multicast groups. This is not necessarily small networks (e.g., at an exchange between large ISPs, it is very likely that there will be at least one member in each ISP domain for all active groups).

Technically, sparse-mode protocols tend to use a shared tree, and a router needs to subscribe to a group to become a member. Dense-mode protocols tend to use source-rooted trees and include by default all multicast routers in the distribution tree. Routers need to send prune messages if they are not interested.

The most popular sparse-mode protocols are PIM-SM and CBT. The most popular dense-mode protocols are DVMRPv3 and PIM-DM.

6.6.1.1 DVMRPv3

DVMRPv3 (Distance Vector Multicast Routing Protocol) is a routing protocol that uses an RPM algorithm to forward multicast packets. It is the dominant protocol of the mBone.

As we saw in Section 6.5.3.1, when a router *R* running an RPM algorithm receives a multicast datagram, it needs to know:

- whether the packet was received on the interface closest to the source (**reverse path forwarding**, or **RPF, check**) from the multicast topology perspective. If it is not, then the packet should have been received first by the interface closer to the source, so this packet is probably a duplicate and must be dropped. Note that in most cases all links on the network are not multicast-enabled, so the interface closest to the source from the unicast topology perspective and the interface closest to the source for the multicast topology perspective will often differ. For this reason, DVMRP runs its own routing protocol in order to take multicast topology into account.

- whether the source of this datagram is closer to R or closer to R neighbor routers. If neighbor routers are closer, they will receive the datagram first, so there is no need to forward the current packet to these routers.

In unicast-routing protocols, such as RIP, each router advertises its best route *from* the router *to* each destination for the unicast topology. The result is that each router knows the unicast distance *from* it *to* each destination.

Here, what we really want to know in order to build an optimized distribution tree is the distance *from* the source *to* the router in the *multicast* topology. This is very often the same, but not always, as in the case of asymmetric links or when using tunnels. All current multicast-routing protocols including DVMRP assume that links are symmetric, so the link symmetry issue is currently ignored. DVMRP solves the issue of multicast-specific topology by using its own routing protocol running over multicast-enabled interfaces.

For each directly attached subnet S , a DVMRP router R advertises the distance from S to R to each neighbor router N_i (in the case of Figure 6.16 just one hop). When N receives the notification that S can reach R in h hops, it first checks whether any other router Z has sent a message saying that S is closer to it. (If this is the case, the accessibility notification from R is not forwarded). Otherwise, N will send a message to each of its neighbors saying that S can reach N in $h + d$ hops, where d is the administrative distance

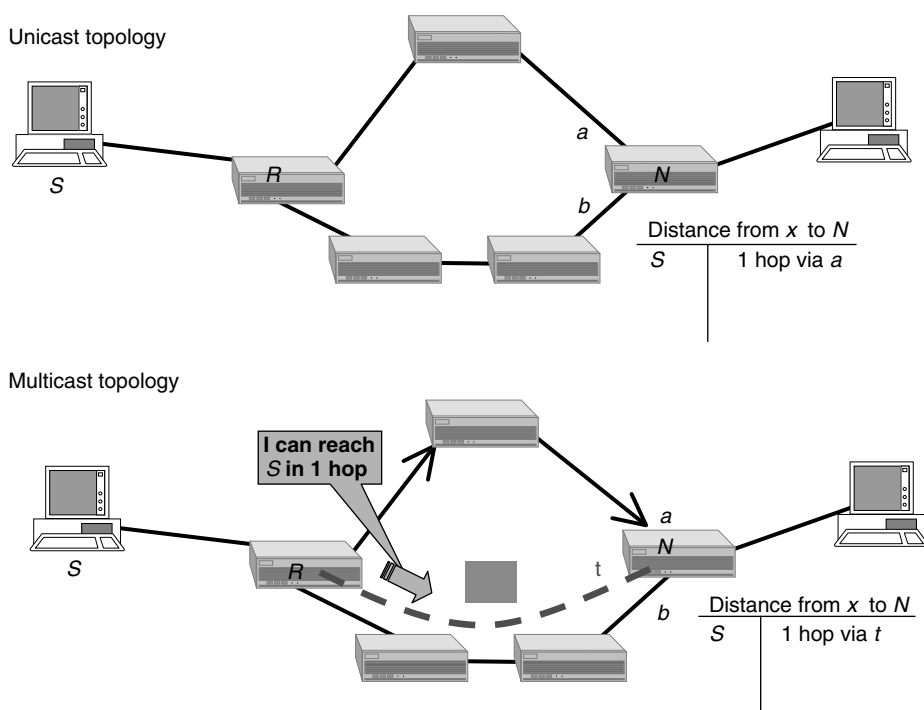


Figure 6.16 Because of tunnel t , the routers see a different topology at unicast and multicast level.

associated with the interface connected to *R*. The interface can be a physical interface or a virtual tunnel interface as in Figure 6.16.

A DVMRP routing table might look like Table 6.2. DVMRP also builds a group-specific forwarding table (Table 6.3) since the routing table does not include group membership information. This table includes by default all interfaces connected to neighbor DVMRP routers (including virtual tunnel interfaces). After prune messages have been received some interfaces are pruned for certain groups (Figure 6.17). On interfaces with directly attached hosts, the forwarding information is based on IGMP queries and reports. Prune states have a lifetime of about 2 hours on the mBone. The number of prunes that routers

Table 6.2 A DVMRP routing table

Source prefix	Subnet mask	From gateway	Metric	Status	Entry lifetime (s)
128.1.0.0	255.255.0.0	128.7.5.2	3	Up	200
128.2.0.0	255.255.0.0	128.7.5.2	5	Up	150
128.3.0.0	255.255.0.0	128.6.3.1	2	Up	150
128.3.0.0	255.255.0.0	128.6.3.1	4	Up	200

Table 6.3 DVMRP forwarding table

Source subnet prefix	Multicast group	TTL In interface (prunes sent)	Out interface(s) (prunes received)
128.1.0.0	224.1.1.1	200 1	2–3
	224.2.2.2	100 1	2–3
	224.3.3.3	250 1	2
128.2.0.0	224.1.1.1	150 2	2–3

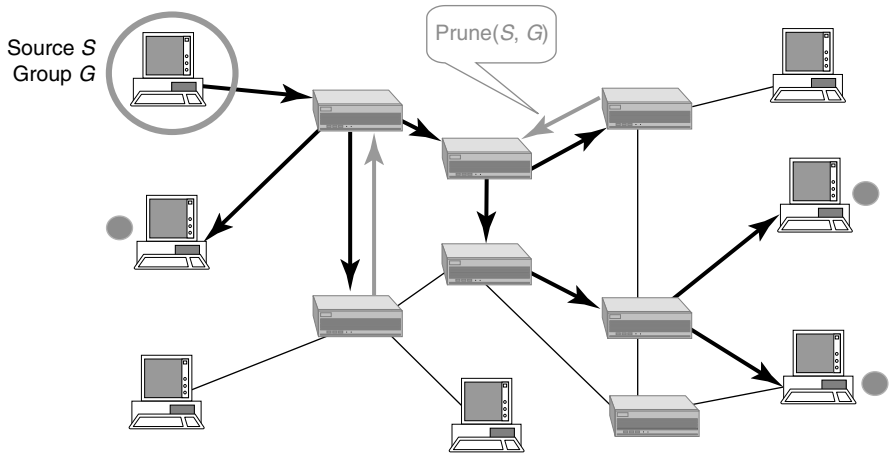


Figure 6.17 Use of the prune message. Subnets with at least one host willing to receive packets of group *G*.

need to maintain (per source, group, and interface) is the main limitation of the scalability of DVMRP. This is the paradox of DVMRP: as the number of listeners increases for a source, the amount of state required in the router decreases. So DVMRP is really a dense-mode protocol!

DVMRPv3 also has a notion of ‘graft’ messages. These graft messages (for each active [source, group] pair) are sent by a router to indicate that it is willing to reattach to a multicast tree for which it had previously sent a prune message (Figure 6.18).

All messages exchanged by DVMRP routers are encapsulated in IP datagrams with protocol number 2 (IGMP) and IGMP packet type 0x13.

Some further improvements of DVMRP are underway, such as **CIDR**-like aggregation. The main issue with the scalability of DVMRP is the periodic flooding that occurs when prune states expire. All DVMRP routers will receive unwanted multicast traffic until they have returned a prune. However, measurements made on the mBone show that this is not yet a real problem. Figure 6.19 is a graph of flooding activity for two pruned sessions (one audio and one video) which can be found on <http://ganef.cs.ucla.edu/~mbone/tunnel.html>. The graph in Figure 6.19 shows that most of the time the session is pruned back immediately after the first packet of the session reaches the router; so, flooding activity is really minimal! The aggregate flood/prune rate for all sessions typically never exceeds 40 packets/s.

6.6.2 Other protocols

6.6.2.1 MOSPF

6.6.2.1.1 Description of operation in a single MOSPF area

The multicast extension to OSPF is described in RFC 1584. **MOSPF** uses the link state information built by OSPF to calculate a shortest path tree on the fly for each

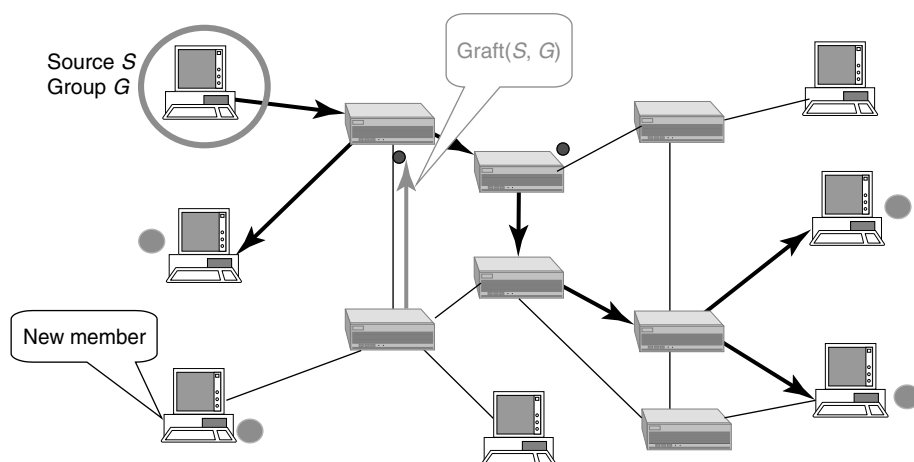


Figure 6.18 Use of the graft message.

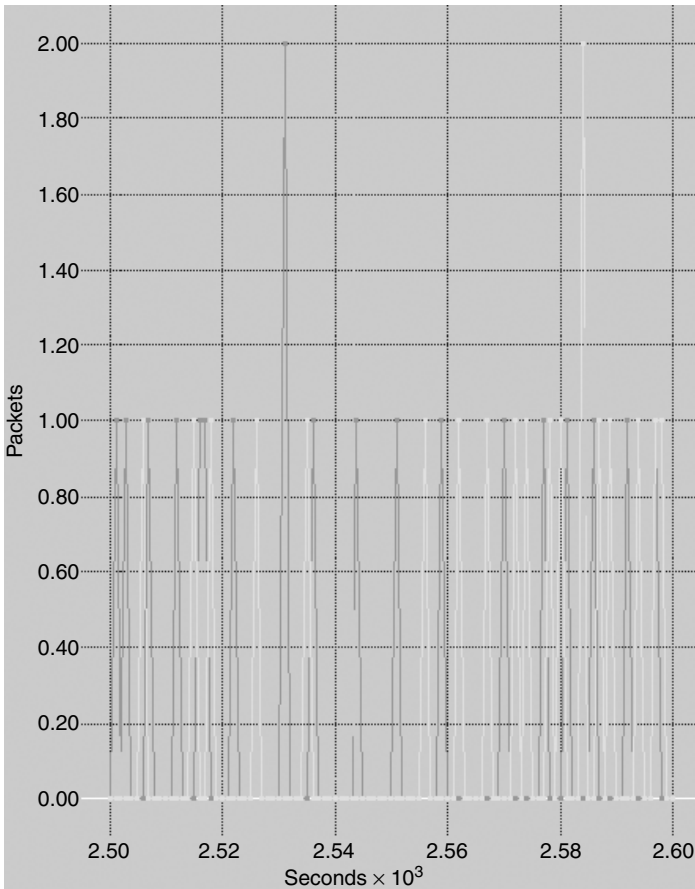


Figure 6.19 Periodic flooding on a typical mBone access.

[source, group] pair. The router knows the multicast topology because link-state advertisements (LSAs) comprise a multicast-capable bit (Figure 6.20), so the tree spans only MOSPF routers.

In addition to the regular OSPF-routing table, each MOSPF router maintains a group membership table. On each subnet, one or two MOSPF routers maintain multicast group memberships in a local group database using IGMP: the designated router (DR) performs IGMP queries on each subnet, and both the DR and the backup designated router (BDR) listen to IGMP host membership reports. The DR then floods the entire OSPF area with ‘group membership link-state advertisements’.

Since each MOSPF router has all the necessary information locally, the multicast tree built using Dijkstra’s algorithm only spans subnetworks that have members of the group, so it does not have to be pruned. This is the major difference with DVMRP (i.e., DVMRP floods the networks whenever there is a new multicast flow and when the prune state expires).

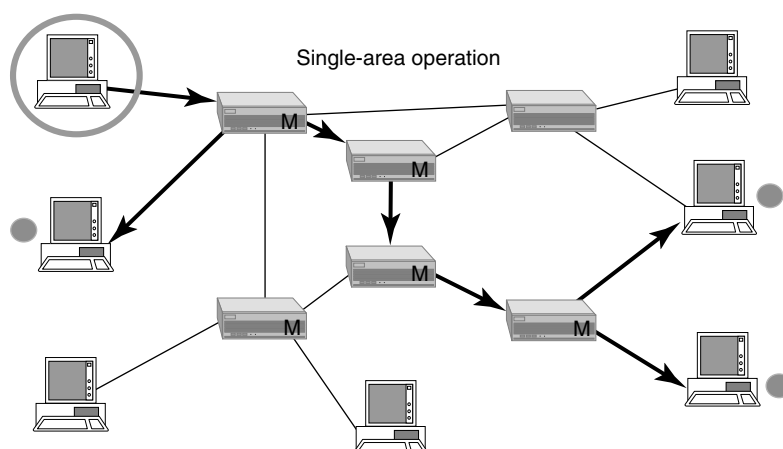


Figure 6.20 MOSPF distribution tree.

6.6.2.1.2 *Inter-area routing*

In OSPF, **area border routers (ABRs)** are used to forward datagrams outside the OSPF area (Figure 6.21). In MOSPF, some are also configured to act as inter-area multicast forwarders. An inter-area multicast forwarder sends new group membership LSAs to the backbone area for each group that has at least one member within the local OSPF area. The inter-area multicast forwarder is a ‘wild card multicast receiver’ for the local OSPF area (i.e., it receives all multicast traffic generated within that OSPF area and decides whether to forward it to the backbone based on the LSAs received from the backbone).

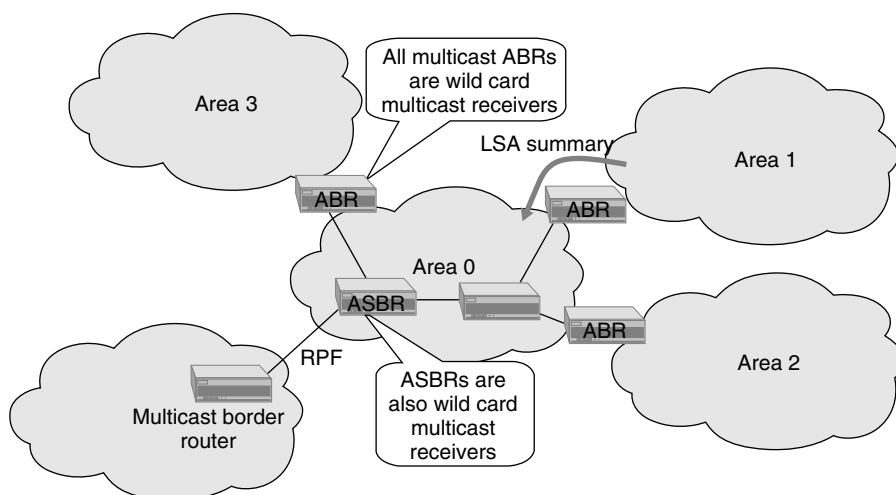


Figure 6.21 MOSPF with multiple areas. Area multicast border router and AS boundary routers handle inter-area and inter-AS multicasting.

6.6.2.2 PIM

The Inter-Domain Multicast Routing working group of the IETF is tasked with developing a set of standards describing multicast-routing protocols. For the moment the working group has defined PIM (protocol-independent multicast), which comes in two flavors: PIM dense mode and PIM sparse mode. PIM-DM and PIM-SM must be used in separate multicast domains; however, packet forwarding and control messages operate seamlessly between the two.

6.6.2.2.1 PIM-DM

PIM-DM (dense mode) relies on the routing tables established by any unicast-routing protocol. This topology information is used to find the route back to the source and build a spanning tree using the reverse path multicasting algorithm. PIM-DM forwards the multicast packets to all downstream interfaces (flooding) until a prune message is received (Figure 6.22). By comparison, DVMRP determines ‘child’ interfaces (i.e., interfaces that are known to be on the shortest path back to the source from the downstream router).

PIM-DM also uses graft messages to reattach a pruned part of the delivery tree if a new member joins the group.

6.6.2.2.2 PIM-SM

PIM-SM (sparse mode) is specified in RFC 2362. By design, PIM-SM is suited for WAN nets that have limited bandwidth and scarce group members. With this constraint, it is impossible to use flooding; so, DVMRP would not scale well.

With PIM-SM, designated routers must explicitly join a group by sending a ‘join’ message to a rendezvous point (RP) for that group (Figure 6.23). There is only one RP per group; this is determined among candidate routers by a deterministic hash function of the group address. Each multicast router in the path of the join message to the RP creates a forwarding entry for that group.

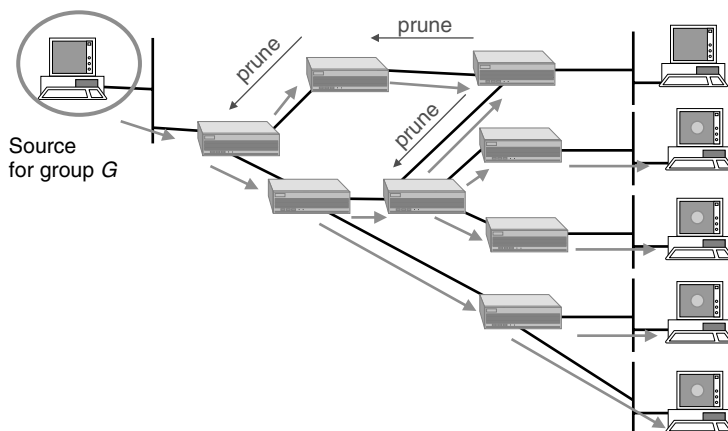


Figure 6.22 PIM-DM also uses prune messages.

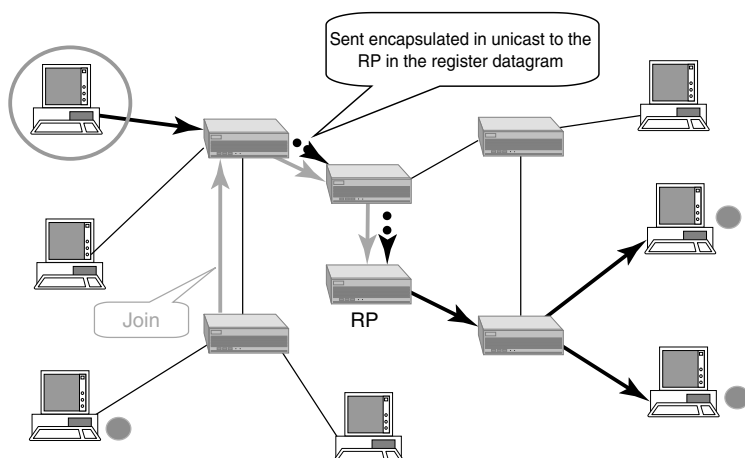


Figure 6.23 PIM-SM uses a rendezvous point for each group.

The first packet of a new multicast stream is sent encapsulated in a unicast 'register' packet to the RP. Each router in the path of this register packet creates a forwarding entry so that future multicast datagrams for this group can be sent unencapsulated.

If the traffic from a source exceeds a certain threshold, the last hop router has the option (it is in no way mandatory) to stop using the RP for that source and build a source-based shortest path tree by sending a join message toward the source of the stream. Once the tree is built, the last hop router sends a prune message for that source to the rendezvous point (Figure 6.24).

6.6.2.3 Core-based trees

CBTs (RFC 2201) have been designed to be used in the context of very large networks, where scalability issues can prevent the use of other RPM-based multicast techniques.

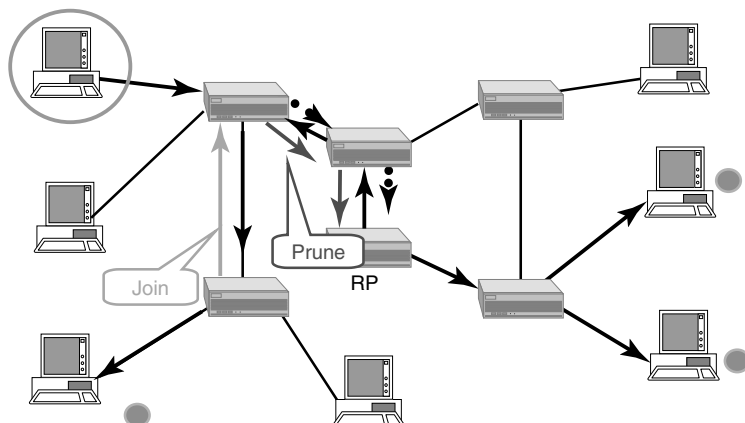


Figure 6.24 PIM-SM allows last-hop routers to switch to a source-based tree.

CBTs use a bidirectional shared tree. Many features are identical to PIM-SM, including the notion of a rendezvous point (called a core router) and an election mechanism. However, the CBT design does not allow shortcuts to be established; this was presented by CBT designers as a feature to preserve the scalability features of CBT. Because the tree to the rendezvous point is bidirectional, routers that are already attached to a group do not need to encapsulate their multicast messages sent to the same group.

6.7 The mBone

6.7.1 An experimental network that triggered the deployment of commercial multicast networks

The mBone started as an experimental network with just 40 subnets in 4 countries in 1992; by January 1998 there were nearly 6,000 subnets. It was composed of islands of multicast routers interconnected by tunnels over regular Internet links, in which case multicast datagrams are conveyed on the tunnels as IP over IP datagrams (protocol 4).

Today, many service providers offer multicast support as a commercial service, not just for experiments, either for their own TV over IP offerings or for corporations looking for efficient broadcast over IP capabilities.

6.7.2 Routing protocols and topology

Most routers run DVMRP (MOSPFv2 does not handle tunnels), but the islands themselves may run MOSPF, PIM, or CBT. The mBone was structured around main nodes, often universities or research labs, which in turn offered multicast connectivity to smaller networks. Figure 6.25 shows the main nodes of the mBone in France back in 1996.

6.7.3 mBone applications

Today, many commercial multicast applications exist; however, for a first contact with multicast, the tools developed for use on the mBone offer a useful introduction. These applications help us to better understand the issues and limitations of SDP for use in VoIP. SDP was really designed for multicast conferences.

6.7.3.1 Videoconferencing with RTP on multicast networks

On unicast networks, RTP can be used for point-to-point communications, but it requires a mixer or multi-unicast for multipoint communications. On a multicast network, such as the mBone, RTP and RTCP packets can be broadcast to all participants, and mixing is done locally by the receiving software.

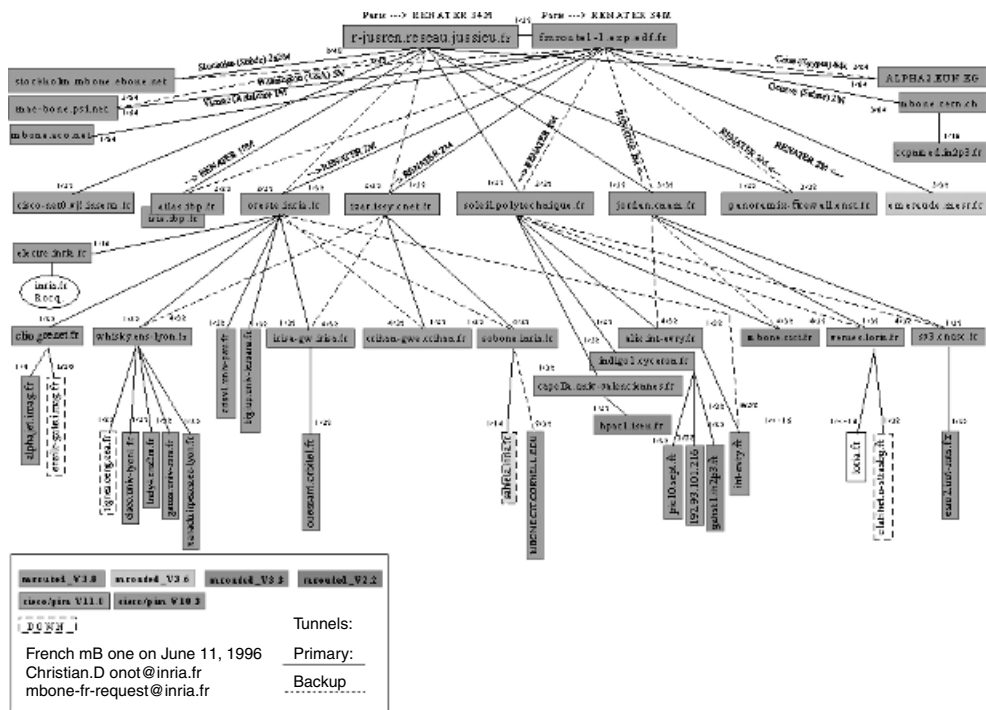


Figure 6.25 A map of the mBone in France in 1996. Reproduced from INRIA.

For all media, two UDP ports are allocated: one for RTP and one for RTCP; however, a single multicast address can be used for the whole conference. For public mBone sessions, this multicast address and port are encoded using SDP session descriptions transmitted using the Session Announcement Protocol.

A receiver knows who originated an RTP packet from the SSRC identifier of the RTP packet. This SSRC can be mapped to a CNAME as soon as an RTCP sender report is received. A receiver should also try to synchronize audio and video streams whose SSRCs correspond to a common CNAME.

Using separate multicast addresses for audio and video allows receivers to choose to receive only audio if they do not have sufficient bandwidth. If a common multicast address is used, multiplexing can still be achieved using a UDP port or payload type; but, since the protocol used to subscribe to a multicast group, IGMP, cannot distinguish between payload types or UDP ports, a potential member of the conference can only receive all media streams or none (unless TTL-based scoping is used, with different TTLs for audio and video).

In case of congestion during the conference, participants will become aware of it with incoming receiver reports and can decide to reduce the number of frames per second of

their video streams. They can also dynamically change the audio codec used, since the codec used can be learned from the value of the RTP packet payload type.

6.7.3.2 SDR

SDR (session directory) is a tool based on the Session Announcement Protocol (SAP). It is used to list announced sessions on the mBone and could be used to advertise new sessions (Figure 6.26). Depending on the version, the SDR tool can also launch and automatically configure some multicast applications from SAP data.

SDR uses 239.255.255.255 for local scope groups and 224.2.127.254 for global scope groups. For administratively scoped groups, the highest address in the scoped range should be used. Any UDP port is suitable, but the tradition is to use port 9875.

SAP is a simple text-based protocol, most of whose data fields are self-explanatory. For the media description portion, it uses the Session Description Protocol (SDP). For more details about SDP, refer chapter 3 in *IP Telephony: Deploying Voice-over-IP Protocols*. The multicast origin of SDP in a send once, receive many and loose coupling context explains its shortcoming when used in duplex, one-to-one, interactive applications such

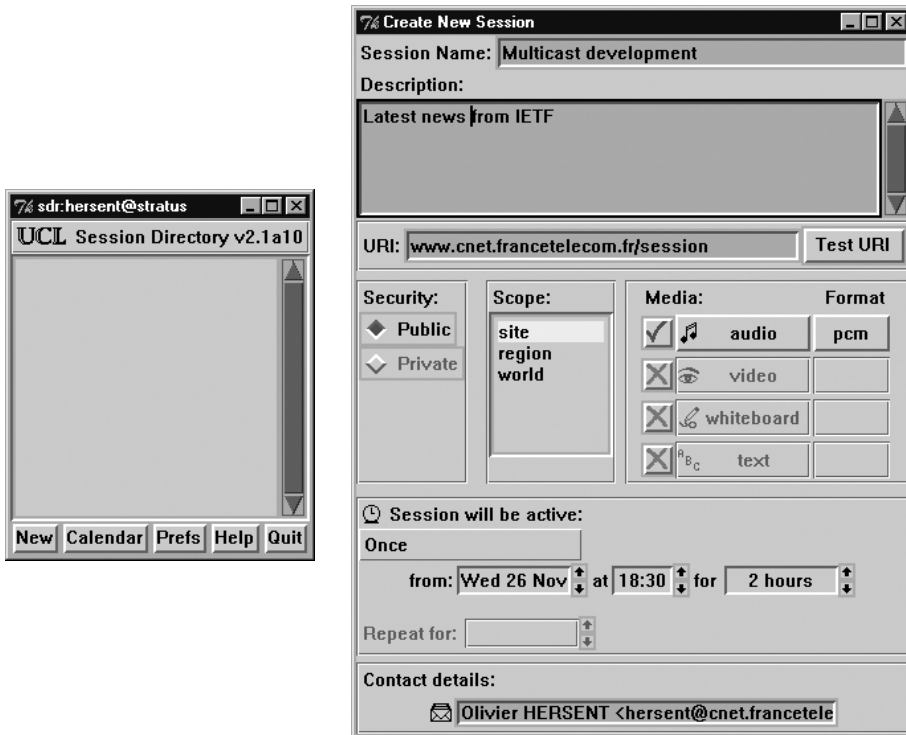


Figure 6.26 The SDR tool. SDR is used for address assignment, scoping session advertisement, and automatic application launching.

a SIP-based videoconferences: SIP had to add the offer/answer model to SDP, which was not used originally. Listed below is an example announcement using SAP with SDP encoding:

```
SAP: 596 bytes
version: 0
message type: 0
encrypt: 0
compress: 0
auth length: 0
msgid: 8192
address: 130.240.64.20
v=0
```

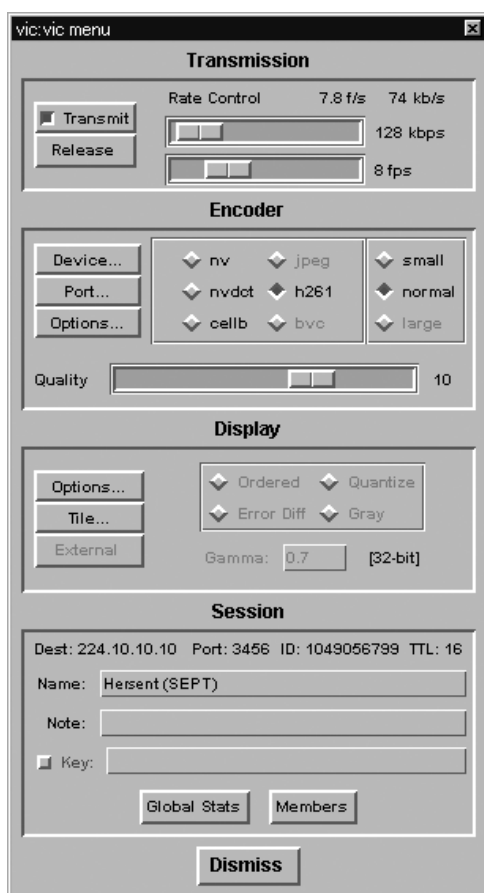
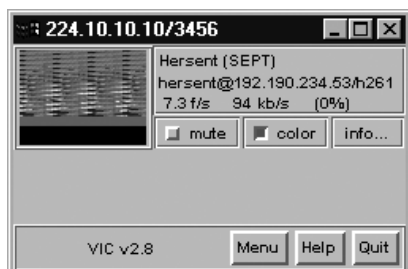


Figure 6.27 Some of the tools that were commonly used on the mBone: VIC video tool, VAT audio tool, wide-band shared whiteboard, and network text editor.

```
o=demo 3066564173 3066564269 IN IP4 130.240.64.67
s=Places all over the world
i=Low bandwidth video (10 kb/s) with views from all over the
world. It is probably wise to limit the overall bandwidth
to 100 kb/s (that is, a maximum of ten 10 kb/s streams).
Audio is primarily for feedback for the senders of video.
e=John Doe <Doe@mydomain.org>
c=IN IP4 224.2.172.238/127
t=0 0
a=tool:mStar 1.0beta1
a=type:broadcast
m=video 51482 RTP/AVP 31
c=IN IP4 224.2.172.238/127
m=audio 20154 RTP/AVP 0
c=IN IP4 224.2.213.113/127
a=rtppred1:5
a=ptime:40
a=rtppred2:5
a=rtpmap:121 red/8000
```

6.7.3.3 VIC and VAT

The VIC and VAT tools are among the very first interactive audio and video applications (Figure 6.27). The limited number of hosts connected to the mBone are mostly confined to universities. Still, even today the VIC and VAT tools are much better suited for large-scale conferencing and broadcasting than most commercial applications.

6.8 MULTICAST issues on non-broadcast media

6.8.1 Bridged LANs

Modern LANs use bridges to reduce the number of collisions. A bridge forwards a packet only to the segment on which the machine with the destination MAC address has been detected. Packets with multicast MAC addresses are traditionally forwarded on all interfaces, which is wasteful. There are several solutions to improve the situation.

6.8.2 IGMP snooping

This solution requires the bridge to inspect all multicast frames in order to decode IGMP reports. This allows the bridge to discover where the receivers are. In addition it decodes router messages like IGMP queries, DVMRP probes, and MOSPF and PIM hellos to discover the position of multicast routers (connected multicast routers need to receive all multicast traffic).

Because hosts never send duplicate IGMP reports, the bridge does not forward a report heard on one segment to another segment in order to see which hosts are receivers on each segment.

This solution has some potential drawbacks: because it relies on the content of IP multicast messages, it does not work for non-IP multicasts and even for IP it may stop working for new IP multicast algorithms. In addition, the inspection of all multicast frames possibly has an impact on performance.

6.8.3 Cisco group management protocol (CGMP)

There is currently no public specification of this proprietary protocol. The idea is to let the router add forwarding entries to the bridge's tables. The router sends CGMP control messages to the bridges. The bridge datagram-forwarding mechanism is left untouched only multicast MAC addresses are added to the forwarding tables for the segments on which the router has detected a member of the multicast group.

6.8.4 IEEE GMRP

GMRP (GARP¹ Multicast Registration Protocol), defined by IEEE 802.1p, is analogous to IGMP at the MAC layer. Hosts wanting to receive frames with a particular multicast MAC address send a GMRP message to the bridge. The bridge propagates this information to the other bridges.

Therefore, a host compatible with GMRP must, after sending the IGMP message to the IP layer, send a GMRP message at the MAC layer.

6.9 Conclusion

VoIP was born on multicast networks with the help of tools like VIC and VAT. Since then VoIP has grown independently on unicast networks, adding to protocols like UDP the missing features required to fully support telephony.

Now that residential VoIP networks increasingly frequently include video on demand and television over IP offerings ('triple play'), VoIP has come face to face with multicast again.

We believe that the combination of VoIP and multicast-enabled tools will open a whole new range of applications for education, remote learning, reporting, and gaming. In the coming years, we will get used to communicating using video and will generate more and more video content (3 G phones, etc.): the combination of multicast and VoIP will enable us to interact and communicate more efficiently using video content.

¹ GARP stands for Generic Attribute Registration Protocol (formerly Group Address Resolution Protocol).

6.10 References

- IGMP version 1: RFC 1112.
- IGMP version 2: W. Fenner, *Internet Group Management Protocol, Version 2*. RFC 2236, November 1997.
- DVMRP: original RFC 1065, 1075, DVMRPv3 *Distance Vector Multicast Routing Protocol* <draft-ietf-idmr-dvmrp-v3-06.txt>.
- MOSPF: RFC 1584, 1585.
- CBT: A. Ballardie, *Core based trees (CBT version 2) multicast routing*: Protocol specification. RFC 2189, September 1997.
- PIM-SM: Estrin, Farinacci, Helmy, Thaler, Deering, Handley, Jacobson, Liu, Sharma, and Wei. *Protocol independent multicast-sparse mode (PIM-SM)*: Protocol Specification. RFC 2117, June 1997.
- Domain interoperability: *Interoperability Rules for Multicast Routing Protocols*
- <draft-thaler-multicast-interop-02.txt> (Exp Sept 98).
- Border Multicast Protocol: *Border Gateway Multicast Protocol* <draft-ietf-idmr-gum-01.txt>.
- HDVMP: A.S. Thyagarajan and S.E. Deering. Hierarchical distance-vector multicast routing for the MBone. In: *Proceedings of the ACM SIGCOMM*, pp. 60–66, October 1995.
- MBGP: *Border Gateway Multicast Protocol* <draft-ietf-idmr-gum-02.txt> (Thaler, Estrin, and Meyer).
- MASC: *Multicast-Address-Set advertisement and Claim mechanism* <draft-ietf-idmr-masc-00.txt> (Estrin, Handley, and D. Thaler).
- Multicast address allocation extensions to the Dynamic Host Configuration Protocol <draft-ietf-dhc-mdhcp-03.txt> (S. Patel).
- Multicast address allocation extensions options <draft-ietf-dhc-multopt-02.txt> (S. Patel).
- AAP: Multicast Address Allocation Protocol <draft-handley-aap-00.txt> (Handley).
- IPv6: R. Hinden and S. Deering, *IPv6 Multicast Address Assignments*, <draft-ietf-ipngwg-multicast-assgn-04.txt>, July 1997.
- BGP4+: T. Bates, R. Chandra, D. Katz, and Y. Rekhter, *Multiprotocol Extensions for BGP-4*, RFC 2283, February 1998.
- BGP4 Y. Rekhter and T. Li, *A Border Gateway Protocol 4 (BGP-4)*, RFC 1771, March 1995.
- Multicast and firewalls: <draft-finlayson-mcast-firewall-00.txt> *IP Multicast and Firewalls*.
- Multicast: S. Deering, *Host Extensions for IP Multicasting*, RFC 1112, August 1989.
- Firewalls: N. Freed, and K. Carosso, *An Internet Firewall Transparency Requirement*, Work-in-Progress, Internet-Draft <draft-freed-firewall-req-02.txt>, December 1997.
- RTP: H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, *RTP: A Transport Protocol for Real-Time Applications*, RFC 1889, January 1996.
- SAP: M. Handley, *Session Announcement Protocol*, Work-in-Progress, Internet-Draft <draft-ietf-mmusic-sap-00.txt,ps>.
- Administrative scoping: D. Meyer, *Administratively Scoped IP Multicast*, Work-in-Progress, Internet-Draft <draft-ietf-mboned-admin-ip-space-04.txt>, November 1997.
- Domain-wide reports: B. Fenner, *Domain Wide Multicast Group Membership Reports*, Work-in-Progress, Internet-Draft <draft-ietf-idmr-membership-reports-00.txt>, November, 1997.
- UMTp: R. Finlayson, *The UDP Multicast Tunneling Protocol*, Work-in-Progress, Internet-Draft <draft-finlayson-umtp-02.txt>, February 1998.
- SOCKS: M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, and L. Jones, *SOCKS Protocol Version 5*, RFC 1928, April 1996.
- SOCKS: D. Chouinard, *SOCKS V5 UDP and Multicast Extensions*, Work-in-Progress, Internet-Draft <draft-chouinard-aft-socksv5-mult-00.txt>, July 1997.

IP Telephony

IP Telephony

Deploying Voice-over-IP Protocols

OLIVIER HERSENT, Netcentrex, France

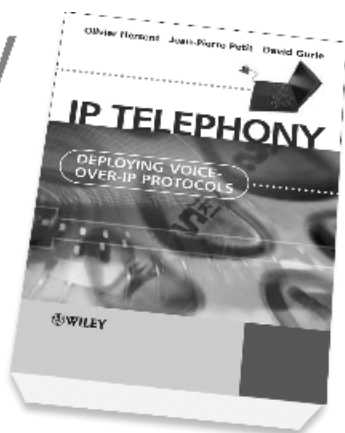
JEAN-PIERRE PETIT, France Telecom

DAVID GURLE, Reuters, France

IP (INTERNET PROTOCOL) TELEPHONY, ENABLED BY SOFTSWITCHES, IS GOING TO USHER IN A NEW ERA IN TELECOMMUNICATIONS. BY PUTTING VOICE AND DATA OVER ONE IP NETWORK, OPERATORS CAN ENJOY LOWER COSTS AND CREATE NEW, REVENUE-GENERATING "MULTIMEDIA" SERVICES.

This valuable reference offers a comprehensive overview of the technology behind IP telephony and offers essential information to network engineers, designers and managers who need to understand the protocols and explore the issues involved in migrating the existing telephony infrastructure to an IP-based real time communication service.

Drawing on extensive research and practical development experience in VoIP from its earliest stages, the authors give access to all the relevant standards and cutting-edge techniques in a single resource.



IP Telephony: Deploying Voice-over-IP Protocols:

- Assumes a working knowledge of IP and networking and addresses the technical aspects of real-time communication over IP
- Presents a high level overview of packet media transport technologies, covering all the major VoIP protocols - SIP, H323 and MGCP
- Details specific strategies to design services for public networks where endpoints cannot be trusted and can be behind firewalls
- Explores the problems that may arise from incomplete protocol implementations, or architectures optimized for private networks which fail in a public environment.

This amply illustrated, state-of-the art reference tool will be an invaluable resource for all those involved in the practical deployment of VoIP technology.

0-470-02362-7 | December 2004 | Hbk | 360pp | £45.00, \$60.00, \$85.00

HOW TO ORDER...

EUROPE, MIDDLE EAST,
AFRICA & ASIA

John Wiley & Sons Ltd
Tel: +44 (0)1245 812700
Fax: +44 (0)1245 812756
e-mail: cs.books@wiley.co.uk
www.wiley.com

NORTH CENTRAL &
SOUTH AMERICA

John Wiley & Sons Inc
Tel: 877 762 2674
Fax: 800 398 3226
e-mail: cs.books@wiley.com
www.wiley.com

GERMANY, SWITZERLAND
& AUSTRIA

Wiley-VCH Verlag GmbH
Tel: +49 (0)7141 676 700
Fax: +49 (0)7141 676 704
e-mail: cs.books@wiley.de
www.wiley.de

All books are available
for purchase at the
discounted price of
£30.00
Please add mailing
costs
Permitted price: £30.00

 **WILEY**

60714

Beyond VoIP Protocols: Understanding Voice Technology and Networking Techniques for IP Telephony

by O. Hersent, J.P. Petit, D. Gurle

Copyright ©2005 John Wiley & Sons, Ltd. ISBN: 0-470-02362-7

Index

- ABS, 38
- ADPCM, 40, 48
- A-law, 16, 17
- aliasing, 12, 13
- AMR, 71
- AMR-WB, 71, 73
- analog to digital conversion, 10
- anti-aliasing filter, 14
- auditory perception, 37

- behavior aggregate, 133
- Bernoulli distribution
 - model, 216
 - traffic assumptions, 215
- bundling, 103

- CableLabs, 154
- call management server, 155
- call seizures
 - call server model, 214
 - Poisson source, 212
- CBT, 248
- CGMP, 254
- class based queuing, 120
- class selector codepoint, 133

- CMS, 155
- CMTS gate, 154, 155
- codec
 - AAC, 38
 - AC3, 37
 - activity bitrate, 184
 - ADPCM, 40, 48
 - analysis by synthesis, 38, 56
 - bit error rate sensitivity, 50
 - CELP, 61
 - Dolby-Digital, 37
 - embedded, 51
 - G.722, 52
 - G.722.1, 55
 - G.723.1, 66
 - G.726, 40
 - G.727, 51
 - G.728, 69
 - G.729, 64
 - GSM 06.10, 58
 - GSM 06.90, 72
 - GSM FR, 58
 - LPC 10, 36
 - MELP, 36, 37
 - MPEG-2 AAC, 37

- codec (*continued*)
 - network overhead, 185
 - NICAM, 39
 - quality, 74
 - REL P, 57
 - wide-band, 52, 71
- codepoint DiffServ, 133
- coder waveform, 47
- coding
 - entropy, 47
 - Huffman, 47
 - perceptual, 37
- companding, 17
- compressed RTP, 175
- conferences
 - audio multicast, 208
 - audio multi-unicast, 207
 - video, 211
- continuous-time, 11
- controlled load service, 140
- COPS
 - commands, 163
 - profile for DQoS, 161
 - protocol, 160
- critical band analysis, 37
- CRTP, 104, 175, 176
-
-
- data stream, 118
- DCME, 48
- DCS, 155
- DECT, 48
- delay versus echo, 90, 107
- Dense mode multicast, 241
- DiffServ, 3
 - as core of RSVP network, 150
 - DS byte, 133
 - DS codepoint, 133
 - IP precedence, 131
 - issues, 135
 - per hop behavior, 133
- dimensioning
 - audio multipoint communications, 206
 - average waiting time, 224
 - call arrivals, 212
 - converged voice-data network, 203
 - large networks and full availability, 220
 - large networks with Erlang B, 221
 - loss rate, 190
 - multiple coders, 198
 - multiple streams, 188, 198
 - overhead, 192
 - queue with finite number of servers, 222
 - single coder, 188
 - single stream, 184
 - small network with Erlang C, 222
 - time probability of waiting, 223
 - traffic matrix, 200
 - video multipoint communications, 211
 - with fault-tolerant network, 202
- discrete time, 11
- discrete-to-continuous conversion, 11
- DOCSIS, 155
- DQoS, 154
 - commands, 163
 - gate, 157
 - sample call flow, 168
 - session, 155
- duplexer, 93
- DVMRP routing table, 241, 243
-
-
- echo
 - acoustic, 94
 - canceller, 96
 - electric, 93
 - hybrid, 91, 93
 - listener, 91
 - suppressor, 96
 - talker, 91
 - versus delay, 107
- embedded MTA, 155, 159
- E-model, 113
- EMTA, 155, 159
- Engset distribution

- model, 217
- traffic assumptions, 215
- Erlang first law *see* Erlang B
- Erlang B distribution
 - model, 221
 - traffic assumptions, 215
- Erlang C distribution
 - model, 222
 - traffic assumptions, 215
- exponential distribution, 213
- fair queuing, 120
- fair queuing bitwise round robin, 121
- FIFO queue, 120
- Filter
 - bandpass, 30
 - bank, 37
 - digital, 21
 - high pass, 30
 - long term predictor, 45
 - low pass, 30
 - LPC analysis, 32
 - LPC synthesis, 32
 - LTP, 45
 - notch, 30
 - perceptual weighting, 61
- filters
 - finite impulse response, 25
 - infinite impulse response, 25
- FIR, 26
- formants, 33
- frame
 - per IP packet, 103
 - size, 39, 103
- frequency aliasing, 13
- G.107, 113
- G.131, 107
- G.165, 94, 96
- G.168, 96
- G.711, 16
- G.722, 52
- G.722.1, 55
- G.722.2, 73
- G.723.1 discontinuous transmission,
 - 66, 68
 - G.726, 40, 42, 47
 - G.727, 51
 - G.728, 69
 - G.729, 64
 - GARP, 254
 - gate controller, 155
 - gate DQoS states, 157
 - generalized processor sharing,
 - 122
 - GPS, 122
 - GSM
 - 06.10, 58
 - 06.60, 61
 - 06.90, 72
 - EFR, 71
 - FR, 71
 - Full Rate, 58
 - HR, 71
 - guaranteed service, 140
 - Huffman, 47
 - IGMP
 - snooping, 253
 - v1, 233
 - v2, 235
 - v3, 235
 - IIR, 26
 - impulse response, 25
 - finite, 26
 - infinite, 26
 - integrated services, 140
 - interactivity, 111
 - interleaving, 105
 - IntServ, 4, 140
 - IP precedence field, 130
 - jitter, 90, 120
 - LD-CELP, 69
 - leaky bucket regulator, 119
 - linear prediction, 30
 - linear prediction coefficients, 44
 - look-ahead, 103
 - LPC analysis, 44

- masking frequency, 37
- mBone, 249
 - SDR, 251
 - VAT, 253
 - VIC, 253
- mean opinion score
 - CMOS, 80
 - common coders, 50
 - DMOS, 80
 - G.722, 54
- modulation pulse amplitude, 10
- MOSPF inter-area routing, 244
- MPEG 2 AAC, 37, 38
- MTA, 154
- multicast, 5, 227
 - address, 230
 - CGMP, 254
 - dense mode, 241
 - DVMRP, 241
 - group, 231
 - IGMPv1, 233
 - IGMPv2, 235
 - IGMPv3, 235
 - mbone, 249
 - MOSPF, 244
 - PIM, 247
 - pruning, 240, 243
 - reverse path broadcasting, 240
 - reverse path multicasting, 240
 - shared tree, 238
 - source based tree, 239
 - spanning tree, 238
 - sparse mode, 241
 - truncated reverse path
 - broadcasting, 240
 - TTL, 236
- multilink-PPP, 177
- multimedia terminal adapter, 154
- NCS, 155
- Nyquist rate, 13
- Nyquist theorem, 13
- overhead compression, 174, 175
- packet by packet generalized
 - processor sharing, 124
- packet fragmentation, 177
- packet loss, 90
- PacketCable, 154
- PAM, 11
- PCM, 14, 49
- PDP, 156
- PEP, 156
- per hop behavior, 133
- perceptual coding, 37
- PGPS, 124
 - delay bound, 126
 - network, 129
 - packet ordering, 128
- PHB, 133
- PIM, 247
 - DM, 247
 - SM, 247
- Poisson distribution traffic
 - assumptions, 215
- Poisson process, 212
- policy decision point, 156
- policy enforcement point, 156
- post-filter, 38
- prediction
 - linear, 44
 - long term, 45
 - temporal, 44
- pulse amplitude modulation, 11
- pulse code modulation, 14, 49
- quantization, 10, 14
- quantizer
 - adaptive, 39
 - backward, 40
 - differential, 42
 - forward, 39
 - linear, 15
 - nonlinear, 16
 - optimal, 17
 - predictive, 42
- random early detection, 120, 173
- RED, 120, 173

- redundancy media transmission, 105
- resource allocation protocol, 156
- reverse path broadcasting, 240
- reverse path forwarding, 241
- reverse path multicasting, 240
- RFC 791, 130
- RFC 1349, 131
- RFC 1990, 177
- RFC 2474, 133
- RFC 2475, 134
- RFC 2686, 178
- RFC 2753, 156
- RSVP, 139
 - controlled load service, 140
 - guaranteed service, 140
 - PATH message, 141
 - RESV message, 142
 - scaling, 145, 148
 - soft states, 145
 - tunneling over DiffServ, 150
- RSVP+, 155, 158
- RTP compressed, 175
- sampling, 10
- sampling Theorem, 12
- scheduling
 - class-based, 120
 - FIFO, 120
 - fair queuing, 120
 - weighted fair queuing, 120
 - policy, 120
- session, 119
- signal residual, 44
- signal processing, 9
- signal-to-noise ration, 15
- SNR, 15
- sound
 - plosive, 33
 - unvoiced, 33
 - voiced, 33
- sparse mode multicast, 241
- speech production, 32
- system
 - causal, 26
 - physically realizable, 26
- TCP slow start, 171
- token bucket regulator, 119
- TOS octet, 130
- traffic-class octet, 130
- traffic matrix, 200
- transfer function, 25
- Truncated Reverse Path Broadcasting, 240
- TTL multicast, 236
- Type of Service, 130
- μ -law, 16, 17
- UMTS, 72, 73
- VAD Intro, 4, 113
- vocoder, 36
- voice activity rate, 185
- voice activity detection, 4, 113
- voice quality
 - ACR test, 77
 - average level, 112
 - CCR, 80
 - clipping, 112
 - CMOS, 80
 - DCR, 80
 - DMOS, 80
 - E-model, 113
 - interactivity, 111
 - MOS, 77
 - objective measurement, 75
 - subjective measurement, 75
- vocoder LPC 10, 36
- waveform coder, 47
- weighted fair queuing, 120
- weighted random early detection, 120, 173
- WRED, 120, 173
- Z transform
 - table, 23
 - unilateral, 22