

Torsten Braun, Michel Diaz, José Enríquez-Gabeiras, and Thomas Staub

# End-to-End Quality of Service Over Heterogeneous Networks

February 12, 2008

Springer



# Foreword



# Preface

The Internet has evolved from an academic network for data applications such as file transfer and net news, to a global general-purpose network used for a variety of different applications covering electronic mail, voice over IP, television, peer-to-peer file sharing, video streaming and many more. The heterogeneity of applications results in rather different application requirements in terms of bandwidth, delay, loss, etc. Ideally, the underlying network supports such Quality-of-Service parameters such that applications can request the desired services from the network, and do not need to take actions by themselves to achieve the desired communication quality. Initially, the Internet was not designed to support Quality-of-Service, and only since the last decade have appropriate mechanisms been developed. Those mechanisms mainly operate on the Internet Protocol (IP) level, but also network-specific mechanisms—e.g., targeted to particular wired/wireless access network technologies—are required.

The goal of the European 6th Framework Programme (FP6) Integrated Project "End-to-end Quality of Service Support over Heterogeneous Networks" (EuQoS) was to develop, implement, and evaluate concepts and mechanisms to support QoS end-to-end, meaning that QoS mechanisms in end systems, access networks, inter-domain links and within domains must be supported. The EuQoS project developed an impressive set of innovative solutions and novel scientific ideas to support end-to-end QoS in the Internet. New mechanisms and concepts were designed and implemented in a European-wide distributed testbed. In addition to the rather technical design and implementation work, the project also developed training material introducing basic QoS mechanisms and techniques. Several e-learning modules were developed and are currently being used at several partner universities for teaching on MSc or PhD levels.

The significant technical and educational results achieved during the EuQoS project, motivated us to use the gained knowledge and experiences of the project partners and write this book on end-to-end QoS in heterogeneous IP networks. The book basically consists of three parts. In Chapters 1-4, we discuss QoS mechanisms and protocols such as scheduling schemes, QoS architectures metrics and measurement techniques, traffic engineering and signalling protocols, and the latest

standardisation activities. Chapter 5 describes related work and recent development in the area of transport protocols, in particular how TCP can be optimised towards QoS support and fairness. The EuQoS system presented in Chapter 6 extends and combines the basic mechanisms discussed in the previous chapters. We show how a combination of different QoS enabling mechanisms and protocols can be used and extended to build a comprehensive end-to-end QoS architecture over heterogeneous wired/wireless access networks. To evaluate QoS mechanisms and architectures, appropriate evaluation schemes are required. The two chapters in the annex describe how simulation—in particular the well-known network simulator ns-2—as well as emulation techniques can be used for tests and evaluations.

This book, which is based on the achievements of the EuQoS project, would not have been possible to compile without the funding from the European Commission, as well as the tremendous efforts and enthusiasm of all the people involved in the project. Special thanks to Mark Günter for proof-reading the text contributions to this book.

*Torsten Braun*

*Michel Diaz*

*José Enriquez Gabeiras*

*Thomas Staub*

Bern, Toulouse, Madrid. January 2008.

# Acknowledgements

The book editors and authors would like to thank all people who were involved in the EuQoS project:

- **Telefonica I+D:** José Enríquez Gabeiras, Francisco Javier Ramón Salguero, Gerardo García de Blas, Antonio J. Elizondo Armengol, Francisco, Romero Bueno, Jesús Bravo Ivarez, Jorge Andrés Colás, María Ángeles, Callejo Rodríguez, María Luisa García Osma
- **University of Pisa (CPR/UoPisa):** Enzo Mingozzi, Giovanni Stea, Luciano Lenzini, Luca Bisti, Claudio Cicconetti, Linda Martorini, Abraham Gebrehiwot, Simone Bisogni, Paolo Sozzi
- **Elsag Datamat:** Enrico Angori, Giuseppe Martufi, Marco Carusio, Alessandro Giorgini, Andrea Paselli, Giovanni Saccomandi, Marco Mauro
- **CNRS (LAAS-CNRS, ENSICA):** Michel Diaz, Florin Racaru, Ernesto Exposito, Philippe Owezarski, Patrick Senac , Christophe Chassot , Nicolas Larrieu, Laurent Dairaine, Mathieu Gineste, Nicolas Van Wambeke, Slim Abdellatif, Sébastien Ardon, Roberto Willrich, Guillaume Auriol, Silvia Farraposo
- **France Telecom R&D:** Olivier Dugeon, Walid Htira, Michel Bourbao, Pascal Le Guern, Jean-Louis Le Roux, Stéphane Statiotis, Régis Fréchin, Claire Teisseire
- **Polska Telefonía Cyfrowa (ERA):** Michal Obuchowicz, Robert Parzydo , Adam Flizikowski, Edyta Rafalska, Karol Jez, Krzysztof Horszczaruk, Krzysztof Bronarski, Krzysztof Samp, Maciej Rozowicz, Michal Dudzinski, Pawel Caban, Piotr Zadroga, Slawomir Tkacz
- **Martel:** Martin Potts, Mark Guenter, Sandra Wittwer
- **NICTA:** Emmanuel Lochin, Guillaume Jourjon, Sebastien Ardon, Ernesto Exposito, Feiselía Tan, Laurent Dairaine
- **PointerCom:** Roberto Marega, Stefano Salsano, Donald Papalilo, Gianluca Martiniello, Valeria Calcagni
- **Polish Telecom R&D:** Zbigniew Kopertowski, Jaroslaw Kowalczyk, Tomasz Ciszkowski

- **Portugal Telecom Inovação (PTIN):** Jorge Carapinha, Nuno Carapeto, Paulo Loureiro, Arnaldo Santos, Eduardo Silva, Fernando Santiago, Helena Paula Matos, Hugo Manaia, Isabel Borges, Jacinto Barbeira, Filipe Peixinho.
- **Red Zinc:** Donal Morris, Brian Widger, Diarmuid O Neill, Léa Compin, Oscar Cuidad
- **Silogic:** Laurent Baresse, Benoît Baurens, Jean-Philippe Darmet, Yannick Lizzi, François Meaude
- **INDRA ( previously SOLUZIONA):** Ignacio Fresno, Jaime Orozco, Luis Colantes Abril, Pablo Vaquero Barbón, Rubén Romero San Martín, Jorge Alonso, Maria Lurdes Sousa, Raul Manzano Barroso
- **Telscom AG:** Sathya Rao, Marcin Michalak
- **Technical University of Catalonia (UPC):** Jordi Domingo-Pascual, Loránd Jakab, Marcelo Yannuzzi, René Serral-Gracià, Xavier Masip-Bruin
- **University of Bern:** Torsten Braun, Thomas Staub, Dragan Milić, Marc Brogle, Marc-Alain Steinemann, Thomas Bernoulli, Gerald Wagenknecht, Markus Wulff, Patrick Lauer, Markus Anwander, Matthias Scheidegger
- **University of Paderborn/C-LAB:** Isabell Jahnich, Achim Rettberg, Chris Loeser, Michael Ditze, Kay Klobedanz, Sebastian Seitz, Andreas König, Volker Spaarmann, Matthias Grawinkel
- **University of Rome:** Antonio Pietrabissa, Francesco Delli Priscoli, Sabrina Giampaolletti, Emiliano Guainella, Erasmo Di Santo, Gianfranco Santoro, Ilaria Marchetti, Massimiliano Rossi
- **Universidade de Coimbra:** Edmundo Monteiro, Luís Cordeiro, Bruno Carvalho, Fernando Boavida, Gabriela Batista Leão, Isidro Caramelo, Jian Zhang, Jorge Sá Silva, Marília Curado, Maxwell Carmo, Paulo Simões, Romulo Ribeiro, Vitor Bernardo, David Palma, Rui Vilão, Luís Conceição
- **Warsaw University of Technology:** Wojciech Burakowski, Andrzej Beben, Halina Tarasiuk, Jaroslaw Sliwinski, Jordi Mongay Batalla, Marek Dabrowski, Piotr Krawiec, Robert Janowski
- **Ericsson:** Antoine de Poorter, Julio López Roldan, Miguel Angel Recio, Jesus Renero Quintero, José Luis Agundez
- **Hospital Divino Espirito Santo:** José Manuel Ponte, António Vasco Viveiros, Carlos P. Duarte, Paula Maciel, José M. Jesus Silva, Maura Medeiros, Maria Dulce Raposo

# Contents

<b>1</b>	<b>Motivation and Basics</b> .....	1
1.1	Quality of Service and its Parameters .....	1
1.1.1	Delay and Delay Variations in End-to-End Packet Delivery .	2
1.1.2	Bandwidth and Packet Loss Ratio .....	4
1.2	Applications' QoS Requirements .....	5
1.2.1	Types of Network Applications .....	5
1.2.2	QoS Requirements of Applications .....	7
1.3	Packet Scheduling in Network Elements .....	9
1.3.1	(Non)Work-conserving Scheduling Disciplines .....	9
1.3.2	Fairness .....	10
1.3.3	Scheduling Disciplines .....	11
1.3.4	Packet Dropping .....	12
1.4	Quality-of-Service Architectures .....	13
1.4.1	Integrated Services .....	13
1.4.2	Differentiated Services .....	15
1.4.3	End-To-End QoS Mechanisms .....	17
1.5	Implementation and Performance of QoS-aware Applications .....	19
1.5.1	Prerequisites for Successful QoS Applications .....	19
1.5.2	Media Scaling .....	19
1.5.3	Applications' Performance Gain Due to QoS .....	20
1.5.4	Summary .....	22
1.6	Structure of the Book .....	22
<b>2</b>	<b>QoS Measurements in IP-based Networks</b> .....	25
2.1	Introduction .....	25
2.2	Measurement Metrics .....	26
2.2.1	Network Level .....	26
2.2.2	Call level .....	30
2.2.3	User Level .....	32
2.3	Measurement Techniques .....	36
2.3.1	Previous Considerations .....	36

2.3.2	Base Techniques . . . . .	39
2.3.3	Active Measurements . . . . .	41
2.3.4	Passive Measurements . . . . .	47
2.4	Conclusions . . . . .	51
<b>3</b>	<b>Traffic Engineering . . . . .</b>	<b>53</b>
3.1	Introduction . . . . .	53
3.2	A Motivating Example . . . . .	54
3.3	Multi-Protocol Label Switching Architecture . . . . .	56
3.3.1	The Forwarding Component . . . . .	57
3.3.2	The Control Component . . . . .	59
3.3.3	MPLS Optimisation . . . . .	60
3.4	MPLS-based Traffic Engineering . . . . .	62
3.4.1	Constraint-Based Routing . . . . .	62
3.4.2	Explicit Route Signalling . . . . .	66
3.4.3	Traffic Engineering Practices . . . . .	69
3.5	Traffic Engineering and Quality of Service . . . . .	71
3.5.1	QoS Support over MPLS . . . . .	72
3.5.2	Traffic Engineering Extensions for DiffServ . . . . .	74
3.6	Conclusions . . . . .	78
<b>4</b>	<b>Signalling . . . . .</b>	<b>79</b>
4.1	Introduction . . . . .	79
4.2	Session Initiation Protocol (SIP) . . . . .	80
4.2.1	SIP and Its Value Propositions . . . . .	80
4.2.2	Protocol Components . . . . .	81
4.2.3	SIP Messages . . . . .	84
4.2.4	Session Description . . . . .	87
4.2.5	Establishment of a SIP Session . . . . .	88
4.2.6	SIP's Extension . . . . .	91
4.3	The Next Steps In Signalling (NSIS) . . . . .	91
4.3.1	Background and main Characteristics . . . . .	91
4.3.2	Background and Main Characteristics . . . . .	92
4.3.3	Overview of Signalling Scenarios and Protocol Structure . . . . .	95
4.3.4	The NSIS Layer Transport Protocol . . . . .	96
4.4	Common Open Policy Service (COPS) . . . . .	104
4.4.1	COPS Overview . . . . .	104
4.4.2	Basic Model . . . . .	105
4.4.3	COPS Protocol . . . . .	106
4.4.4	COPS Messages . . . . .	110
4.4.5	Common Operation . . . . .	113
4.4.6	Illustrative Examples, Using COPS for RSVP . . . . .	114
4.5	Conclusions . . . . .	116

<b>5</b>	<b>Enhanced Transport Protocols</b> .....	117
5.1	Introduction .....	117
5.2	State of the Art of Transport Protocols .....	118
5.2.1	TCP and UDP .....	119
5.2.2	TCP Evolution .....	120
5.2.3	SCTP .....	122
5.2.4	DCCP .....	123
5.2.5	Discussion .....	124
5.3	Transport Mechanisms .....	124
5.3.1	Overview .....	124
5.3.2	Congestion Control Mechanisms .....	126
5.3.3	Reliability Mechanisms .....	127
5.3.4	Discussion .....	128
5.4	Enhanced Transport Protocol Mechanisms .....	129
5.4.1	TFRC and gTFRC, a QoS-aware Congestion Control .....	129
5.4.2	Application-Aware Transport Mechanisms .....	130
5.5	Conclusion .....	136
<b>6</b>	<b>The EuQoS System</b> .....	137
6.1	Introduction .....	138
6.2	Architecture .....	139
6.2.1	Goals and Requirements .....	139
6.2.2	Functional Blocks and their Main Functions .....	140
6.2.3	Control Plane Elements: RM and RA .....	143
6.3	Provisioning, Invocation, and Operation, Administration and Management .....	145
6.3.1	Provisioning Process .....	146
6.3.2	Invocation Process .....	152
6.3.3	Operation, Administration and Management .....	156
6.4	End-to-End Classes of Service in Heterogeneous Networks .....	156
6.4.1	End-to-end Classes of Service in EuQoS .....	157
6.4.2	QoS Mechanisms and Algorithms for Specification of e2e Classes of Service .....	160
6.4.3	Implementation of e2e Classes of Service in Underlying Technologies .....	162
6.5	EuQoS Enhanced Transport Protocol .....	168
6.5.1	Introduction .....	168
6.5.2	Enhanced Transport Protocol Services for EuQoS .....	169
6.5.3	Services for Streaming/Non-Streaming Applications .....	170
6.6	Multicast .....	171
6.6.1	Application Layer Multicast .....	172
6.6.2	Application Layer Multicast in the EuQoS System .....	173
6.6.3	Multicast Middleware .....	175
6.6.4	Introducing QoS to Multicast Middleware .....	178
6.7	Telemedicine Application .....	180

6.7.1	Telemedicine – the Case for Application-Driven QoS	180
6.7.2	Overview of Medigraf	180
6.7.3	Medigraf Adaptation to EuQoS	182
6.8	Conclusions	184
<b>7</b>	<b>Summary and Outlook</b>	<b>187</b>
<b>A</b>	<b>Implementing Protocols on Network Simulators</b>	<b>189</b>
A.1	Main Simulation Terms and Concepts	189
A.1.1	Simulation Process	190
A.1.2	Simulation Types	190
A.2	Network Simulation	191
A.2.1	Parallel/Distributed versus Serial Execution of Simulations	192
A.2.2	Packet-Level, Fluid-Based and Hybrid Model Simulation	192
A.2.3	Simulation Speedup	193
A.2.4	Network Simulation in Research	193
A.2.5	Simulation for Education Purposes	195
A.3	Network Simulators	195
A.3.1	GloMoSim and Qualnet	195
A.3.2	JiST/SWANS	196
A.3.3	Scalable Simulation Framework (SSF) and SSFNet	196
A.3.4	OMNeT++ and OMNEST	196
A.4	The Network Simulator ns-2	197
A.4.1	The Language Concept	197
A.4.2	Hierarchical Structure	198
A.4.3	First Steps - Simulation Script Template	199
A.4.4	Nodes, Links and Traffic	199
A.4.5	Wireless Networks	202
A.4.6	Implementing Protocols with ns-2	204
A.4.7	Tips for Running ns-2 Simulations	219
A.4.8	Analysing Methods	220
<b>B</b>	<b>Network Emulation Focusing on QoS-Oriented Satellite Communication</b>	<b>221</b>
B.1	Network Emulation Basics	221
B.1.1	Introduction to Network Emulation	221
B.1.2	What is Network Emulation?	223
B.1.3	Why Using Network Emulation?	226
B.1.4	Requirements for Emulation Systems	228
B.1.5	Network Emulation System Approaches	230
B.2	Case Study: Emulation of QoS-oriented Satellite Communication	241
B.2.1	Introduction	241
B.2.2	DVB Satellite Communications	241
B.2.3	QoS Support for Satellite Network Systems	243
B.2.4	Emulation of a DVB-S, DVB-RCS Satellite System	244

B.3 Conclusions .....	253
References .....	255
<b>Index</b> .....	<b>265</b>



# Acronyms

The following list contains acronyms used in the book and their explanation. Most acronyms can be found in the index as well together with a page reference.

ALM	Application layer multicast
API	Application programming interface
CIDR	Classless Internet domain routing
COPS	Common Open Policy Service
DVMRP	Distance-vector multicast routing protocol
IGMP	Internet group management protocol
IP	Internet protocol
IPTV	Internet Protocol Television
IPv4	Internet protocol version 4
IPv6	Internet protocol version 6
ISP	Internet service provider
MM	Multicast Middleware
MOSPF	Multicast open shortest path first
NSIS	Next Step In Signalling
P2P	Peer-to-peer
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PIM	Protocol-independent multicast
QoS	Quality of Service
SDP	Session Description Protocol
SE	Signalling Entities
SIP	Session initiation Protocol
SSQ	Synchronize State Query
TCP	Transmission control protocol
TTL	Time-To-Live
UAC	User Agent Client
UAS	User Agent Server
UDP	User datagram protocol

VLSM	Variable length subnet mask
MPLS	Multi Protocol Label Switching
TE	Traffic Engineering
RIP	Routing Information Protocol
IGP	Interior Gateway Protocol
OSPF	Open Shortest Path First
IS-IS	Intermediate System-Intermediate System
RSVP	ReSerVation Protocol
IETF	Internet Engineering Task Force
FEC	Forwarding Equivalence Class
LSR	Label-Switching Router
LFIB	Label Forwarding Information Base
ATM	Asynchronous Transfer Mode
DLCI	Data-Link Connection Identifier
VPI	Virtual Path Identifier
VCI	Virtual Channel Identifier
BGP	Border Gateway Protocol
LDP	Label Distribution Protocol
LIB	Label Information Base
LSP	Label-Switched Path
CBR	Constraint-Based Routing
TED	Traffic Engineering Database
CSPF	Constrained Shortest Path First
SPF	Shortest Path First
CR-LDP	Constraint-based Routing Label Distribution Protocol
TSPEC	Traffic Specification
ERO	Explicit Route Object
BA	Behavior Aggregate
PHB	Per Hop Behavior
DSCP	Diff-Serv Codepoint
OA	Ordered Aggregate
PSC	PHB Scheduling Class
AF	Assured Forwarding
E-LSP	EXP-Inferred-PSC LSP
L-LSP	Label-Only-Inferred-PSC LSP
BE	Best Effort
DS-TE	Diff-Serv-aware Traffic Engineering
CT	Class Type
BC	Bandwidth Constraint
MAM	Maximum Allocation Bandwidth Constraints Model
RDM	Russian Doll Bandwidth Constraints Model
AC	Access Category
ADSL	Asymmetric DSL
AIFS	Arbitrary Inter-Frame Space
AP	Access Point

AS	Autonomous Systems
ASPB	AS-path Builder
ATM	Asynchronous Transfer Mode
BR	Border Router
BRAS	Broadband Remote Access Server
BRPC	Backward Recursive Path Computation
CAC	Connection Admission Control
CBR	Constant Bit Rate
CoS	Class of Service
CPE	Customer Premises Equipment
CRA	Continuous Rate Assignment
CW	Contention Window
DAMA	Demand Assignment Multiple Access
DCF	Distributed Coordination Function
DSL	Digital Subscriber Line
DSLAM	Digital Subscriber Line Access Multiplexer
DVB-S	Digital Video Broadcasting - Satellite
DVB-RCS	Digital Video Broadcasting - Reverse Channel Satellite
e2e CoS	End-to-end Class of Service
EDCA	Enhanced Distributed Coordination Access
ER	Edge Router
ES	Ethernet Switch
FCA	Free Capacity Assignment
FTP	File Transfer Protocol
GGSN	GPRS Gateway Support Node
HTD	High Throughput Data
IPLR	IP Packet Loss Ratio
IPTD	IP Packet Transfer Delay
IPDV	IP Packet Delay Variation
MAC	Medium Access Control
MT	Mobile Terminal
NCC	Network Control Centre
NRT	Non Real Time
OGGSN	Open GPRS Gateway Support Node
PCC	Path Computation Client
PCE	Path Computation Element
PCEP	PCE Protocol
PQ	Priority Queuing
PR	Peak Rate
RA	Resource Allocator
RBDC	Rate Based Dynamic Capacity
RM	Resource Manager
RNC	Radio Network Controller
RT	Real Time
SHDSL	Symmetrical High Bitrate DSL

SLA	Service Level Agreement
ST	Satellite Terminal
STD	Standard
TERO	Traffic Engineering and Resource Optimization
TOS	(Type of Service)
UTRAN	UMTS Terrestrial Radio Access Network
VBDC	Volume Based Dynamic Capacity
VBR	Variable Bit Rate
VDSL	Very High Bitrate DSL
VoD	Video on Demand
VoIP	Voice over IP
VTC	Video Teleconference
WFQ	Weighted Fair Queueing
WMM	WiFi Multi-Media
WRED	Weighted Random Early Detection
WRR	Weighted Round-Robin
CLI	Command Line Interface
EQ-BGP	Enhanced QoS Border Gateway Protocol
QoS NLRI	QoS Network Layer Reachability Information
DoP	Degree of Preference
SNMP	Simple Network Management Protocol
TMN	Telecommunications Management Network
SAAA	Security, Authentication, Authorization and Accounting
QoS SR	Quality of Service Routing
xDSL	Digital Subscriber Line
UMTS	Universal Mobile Telecommunications System
LAN	Local Area Network
NREN	National REsearch Network
GEANT	Multi-gigabit pan-European data communications network
NTI	Network Technology Independent
NTD	Network Technology Dependent
AQ-SSN	Application Quality Signalling and Service Negotiation
CHAR	CHARging module
QCM	Quality Control Module
MMS	Monitoring and Measurement System
EQ-SAP	EQ-Service Access Point
PQ-WFQ	Priority Queueing - Weighted Fair Queueing
SCTP	Stream Control Transmission Protocol
DCCP	Datagram Congestion Control Protocol
ETP	Enhanced Transport Protocol
gTFRC	TCP-Friendly Rate Congestion Control
TC	Time Constraints
SACK	Selective ACKnowledgement
PCMA	Pulse Code Modulation a-law
PCMU	Pulse Code Modulation mu-law

CIF	Common Intermediate Format
QCIF	Quarter Common Intermediate Format
SQCIF	Sub Quarter Common Intermediate Format
e2e	end-to-end



## Chapter 4

# Signalling

**Ilaria Marchetti, Antonio Pietrabissa, Massimiliano Rossi, Fernando Boavida, Luís Cordeiro, Edmundo Monteiro and Marilia Curado**

**Abstract** Although the Internet does not rely on a connection-oriented paradigm as the PSTN does including connection establishment, data transfer, and connection termination, several signalling protocols are required as well. The need for signalling protocols is manifold. Three of such routing protocols with rather different purposes and goals are the Session Initiation Protocol (SIP), the Next Steps In Signalling (NSIS) framework, and the Common Open Policy Service (COPS) protocol. SIP supports application level signalling to establish, maintain and terminate Voice over IP calls, while NSIS and COPS rather operate at the network level. NSIS is a signalling framework supporting network-level signalling of QoS parameters between network elements such as routers. COPS supports policy-based network management and configuration of network elements.

### 4.1 Introduction

Several approaches to signalling exist on the current Internet. All of them have the common objective of providing some form of control over—and support of—user traffic and services, thus contributing to the smooth operation of the network.

For many years in the past, Internet users have looked at signalling-based approaches as something to avoid. Ideally, signalling on the Internet should be reduced to a minimum and performed by end-systems in order to keep the network as simple as possible. The current use of the Internet is showing that this is not possible anymore and that, for several reasons, some forms of signalling must be used.

Although the current Internet is still data-driven—as opposed to the signalling-driven nature of, for instance, the telephone network—signalling is present in virtually all its components for network operation support, quality of service support, management and application/user support. Routing protocols, such as OSPF and BGP, can be considered operation-oriented signalling protocols, as they are indispensable to the operation of the current Internet. RSVP, MPLS and NSIS are examples of signalling protocols for the support of Quality of Service. SNMP and

COPS are examples of management protocols. On the other hand, SIP and H.323 are examples of application/user support signalling protocols.

The purpose of this chapter is to address and explain three of the most used and most promising signalling protocols/frameworks used in the current Internet: the Session Initiation Protocol (SIP), the Next Steps In Signalling (NSIS) framework, and the Common Open Policy Service (COPS) architecture and protocol.

This chapter starts off with a description of the Session Initiation Protocol. The description addresses the protocol components, SIP messages, session description, session establishment and SIP extensions. The next section is dedicated to the NSIS framework providing some background and main characteristics, presenting the overall architecture and an overview of the protocol structure, as well as describing the main aspects of the NSIS Layer Transport Protocol and of the two main NSIS Signalling Layer Protocols. COPS is described in the last section of the chapter. The description includes architectural and protocol overviews, message formats, common operation and some examples.

## **4.2 Session Initiation Protocol (SIP)**

### ***4.2.1 SIP and Its Value Propositions***

The SIP protocol has been developed with the main purpose of establishing a session with two or more clients wishing to communicate with each other. It is similar to the two major Internet protocols—HTTP (World Wide Web) and SMTP (e-mail)—in that it uses symbolic addresses to represent persons in a session. Like both of them, SIP is a textual client-server protocol, in which the client issues requests and the server returns responses. SIP uses much of the syntax and semantics of HTTP: its response code architecture, many message headers and its overall operations. Also like HTTP, each SIP request is an attempt to invoke some method at the server. There are six SIP methods to establish a session: INVITE, ACK, CANCEL, REGISTER, OPTION and BYE. The most basic is the INVITE method used to initiate a call between the client and the server. Unlike HTTP and SMTP, SIP can run on top of either the Transmission Control Protocol (TCP) or the User Datagram Protocol (UDP). To assure the Quality of Service during the transmission of the messages, the SIP protocol uses the time-out and request-response mechanisms. When the server sends a request, if the answer does not return in a pre-established time (time-out), it assumes that the request was lost and it re-issues the request. In the same way, the caller must receive an ACK message when the response is received. But, if the request is unsuccessful for several times, the client can decide to open the connection with TCP. However, the preferred transport protocol for the SIP is UDP, in order to avoid the time spent in TCP connection set-up and tear-down. When used with TCP, SIP allows many requests and responses to be sent over the same TCP connection, as in HTTP. The SIP protocol allows a perfect integration with other protocols

developed for the IP network (for example: SAP, RTP/RTCP, RTSP, RSVP, SDP, MEGACO), but it does not depend on these, so it can be used with another signalling protocols. In the following the most important capabilities of SIP protocol will be analysed, such as:

- Identification of the locations of the target endpoint: SIP supports address resolution, name mapping and call redirection.
- Identification of the media capabilities of the target endpoint: conferences can be established only knowing the media capabilities supported by all endpoints.
- Identification of the target endpoint availability: if the called party is already connected to a call or did not answer in the allotted number of rings, the SIP protocol sends a message indicating why the target point was unavailable.
- Establishment of a session with two or more parties.
- Handling of transfer and termination of calls: during a call transfer, SIP can establish a session between the transferee (a proxy server in-between) and a new endpoint and terminates a session between transferee and the transferring party.
- Supporting of the advanced personal mobility service: if an endpoint has two addresses (for instance, one for the office and one for home), the user can set up his/her computer to automatically forward the calls to the address where he/she is.

### ***4.2.2 Protocol Components***

The SIP protocol is able to create, modify and terminate voice, video and multimedia sessions thanks to messages sent between two or more parties (clients and servers). SIP is a client-server protocol that involves the following four basic entities: User Agents (that contains both a client protocol, called User Agent Client (UAC) and a server protocol called User Agent Server (UAS)), Registrar, Proxy Server and Redirect Server.

#### **4.2.2.1 User Agent Client (UAC)**

The User Agent Client is an entity that allows initiating a session by sending a request. When a client has to forward a call by the UAC SIP component, the UAC determines the essential parameters to establish a conversation, which is the protocol, the port and the IP address of the UAS to which the request is being sent. The UAC is also capable of using the information in the request URI to establish the path of the SIP request to its destination, as the request URI always specifies the host, which is essential.

#### 4.2.2.2 User Agent Server (UAS)

The UAS is the server that hosts the application responsible for receiving the SIP requests from a UAC, and on reception it returns a response to the request back to the UAC. The UAS can answer more requests from the UAC. The communication between the UAC and the UAS is client-server and peer-to-peer.

#### 4.2.2.3 Proxy Server

Proxy Servers are SIP routers. They receive a message and forward it to a user agent or another proxy along the path. The Proxy Server is the most important entity to understand how the SIP protocol routing works. A typical SIP session is initiated by the user agent client thanks to the services provided by the proxies. Users have their UA configured to be directed to their respective proxy servers. The proxy servers then communicate with each other to convey the message. If the proxy server is used, the caller UA sends an INVITE message to the proxy server. When the proxy server determines a path to send the call and then forwards it Figure 4.1(a), the callee responds to the proxy server, which forwards the response to the caller Figure 4.1(b). Once the proxy server forwards the acknowledgments of both parties, a session is then established between the two clients Figure 4.1(c).

A proxy server can be also used for name mapping. A proxy server can ask a location service and map an external SIP identity to an internal SIP identity. These proxy servers are not firewalls, they are independent servers on the Internet that proxy the request on behalf of the user for various possible reasons. The Proxy Server can use an inter-organisational configuration through which SIP communications are routed. This configuration is used when the messages are routed through the proxy servers before the messages are relayed to the destination SIP client. This can be useful for internal communications where security over an Internet link can be a problem.

#### 4.2.2.4 Registrar

The Registrar server allows redirecting a call to where the client is usually reachable. This is possible by sending a request to change the address to the registrar server. When the server receives these messages, it forwards the requests to the registered address. For instance, a client (that we will call Alice) can be reachable by two addresses:

- at home, where she has a computer - sip: alice@pc1.home.com
- and on her computer at the university laboratory - sip: alice@pc2.lab.uni.edu

Alice registered both addresses with the registrar message at domain.com. She wants to receive the calls at the university between 9:00 and 13:30 and at home from

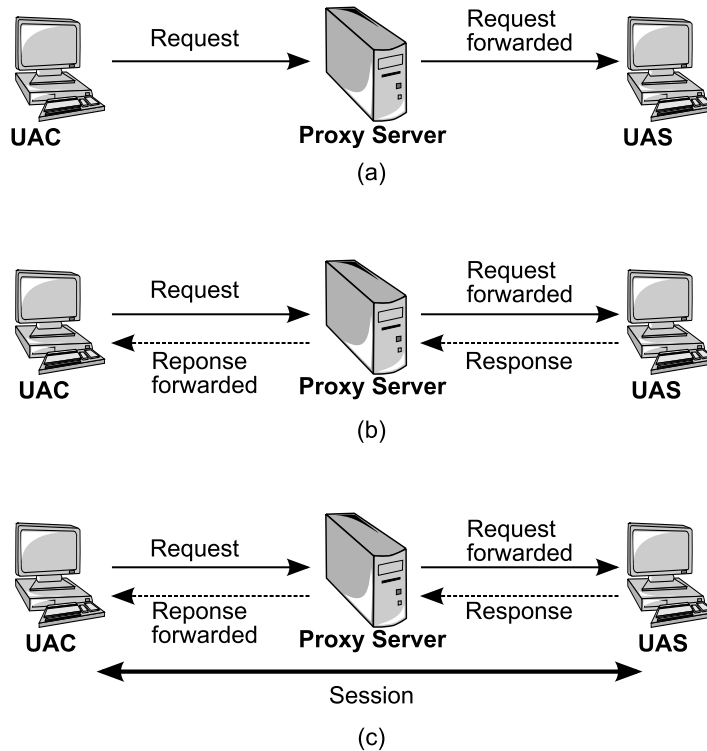


Fig. 4.1 Proxy Server Functionality.

16:00 to 20:00. When the registrar server receives a message addressed at the Alice's public URI (`sip:Alice.Bianchi@domain.com`) it decides where to forward the request by checking when the message will be sent. In this example, we saw that the registrar server routed the messages to Alice's user agent. However, the entity responsible for routing the message is the proxy. Proxies and registrars only play logical roles. In the previous example, the registrar and proxy servers are within the same box: when Alice registered her current location it acts as a registrar, when the server routes the SIP messages it acts as the proxy. However, we can find a box for each server to separate the roles.

#### 4.2.2.5 Forking Proxies

An important feature of SIP is the possibility of forwarding single request calls to multiple destination addresses during session establishment. This function supports the provision of several advanced telephony services, such as automatic call forwarding to Voice Mail or simpler user location. The SIP proxy servers that route messages to more than one destination are called forking proxies. For instance, if

all the telephones ring at the same time in a house, the client has the time to pick up the call in any rooms of the house. This is called parallel forking. A forking proxy can also route a message sequentially. For instance, if a client has two addresses, the server can let a user agent ring for a certain time at the first location. If the client does not pick up the call, the forking proxy forwards the call to the second address.

#### 4.2.2.6 Redirect Server

The Redirect Server helps the UAC to find a new location of the receiver of the message. When it receives a message, it contacts the location server to determine the path to the callee and then sends a reply, which provides the client with information about the next hop (or hops) that the message should take. The caller then sends a message directly to the device indicated in the redirection information. Note that the redirect server does not forward the message to its destination as proxies do.

### 4.2.3 SIP Messages

As previously mentioned, SIP is based on HTTP, thus it is a textual request-response protocol. In order to establish a session, the clients send the requests and the servers answer with responses. The format of SIP messages is composed by *Start Line*, a number of *header fields*, an *Empty Line* and an *Optional message body*.

#### 4.2.3.1 Start Line

*Start Line* differs dependent on a *request* or a *response*. Indeed, the *Request Line* consists of a method name (indicating the purpose of the request), the *Request-URI* (containing the address of the callee), and the protocol version (e.g., SIP2.0). For instance, if the client wants to send an INVITE message to Alice Bianchi, it must compile the *Status Line* as shown in the following table.

Method Name	Request-URI	Protocol Version
INVITE	sip:Alice.Bianchi@domain.com	SIP/ 2.0

**Table 4.1** Start Line

The *response* to the *Start Line* is named *Status Line* and it contains the protocol version (SIP2.0) and the status of the transaction, which is in numerical (status code) and in human-readable (reason phrase) format. For instance, Bob can answer the previous invite request as follows:

In responses, the status code is described with an integer of three ciphers that indicate the meaning of the response. The first cipher defines the class of the re-

Protocol Version	Status Code	Reason Phrase
SIP/ 2.0	200	OK

**Table 4.2** Invite Request

sponse and the others have no particular meaning. Some possible response classes are shown in the following table:

Status code	Meaning
1XX (100 – 199)	Provisional: the request has been received but at the server there is not definitive response so it is continuing to process the request.
2XX (200 – 299)	Success: the request has been successful
3XX (300 – 399)	Redirection: the callee has a new location
4XX (400 – 499)	Request Failure: the request has a syntax error so the server does not understand the message
5XX (500 – 599)	Server Failure: the server does not implement the request that appears to be good
6XX (600 – 699)	Global Failure: the request is not identified from any server

**Table 4.3** Possible Responses for the example

#### 4.2.3.2 Header Field

All messages have the mandatory header fields. In the SIP messages there are different header fields. The most used are:

- *To*: contains the URI of the destination of the request. However, this address cannot be definitive; in fact, as previously said, a client can change its location. In the presence of forking proxies, this field includes a tag value to distinguish among the different user agents that are identified with the same URI.
- *From*: contains the URI of the caller. Like the *To* header field it can specify a tag value.
- *Call-ID*: provides a unique identifier for a SIP message exchange.
- *Cseq*: contains a number and a method name. They are used to match requests and responses.
- *Contact*: contains a list of addresses where the callee can find the caller.
- *Via*: indicates the request's path among proxies. The response uses this header field to keep the same proxies traversed by the request. So it returns more quickly.
- *Max-Forwards*: is used to avoid routing loops. Every proxy that handles a request decrements its value by one, and if it reaches zero, the request is discarded.

Some header fields contain information on call services, addresses and protocol features to establish a session, and a set of header fields provides information about the message body, such as:

- *Content-Type* indicates which is the protocol used in the message body (in general it uses the SDP protocol).
- *Content-Length* contains the length of the body expressed in bytes.

#### 4.2.3.3 Message Body

The message body is separated from the header field by an empty line. SIP messages can carry any type of body, also multi-part bodies using MIME (Multipurpose Internet Mail Extensions) encoding. The MIME is a format that allows sending a message with multiple attachments of different formats. For example, an e-mail message can carry a JPEG picture and an MPEG video. The most important aspect of the Message bodies is that they are transmitted end-to-end. In fact, the proxy server does not need the message body to route the message. Note that the body can be empty.

#### 4.2.3.4 SIP Methods

SIP defines some several methods that are summarised in the following table:

Method name	Meaning
INVITE	Establishes a session inviting a user to a call
ACK	Confirms reliable message exchanges
CANCEL	Terminates a pending request
INFO	Transports PSTN telephony signalling
OPTIONS	Solicits information about the capabilities of the callee, but does not set up a call.
REGISTER	Conveys location information to a SIP server. Allows a user to tell a SIP server how to map an incoming address into an outgoing address that will reach that user
BYE	Terminates a session
NOTIFY	Notifies the user agent about a particular event
PRACK	Acknowledges the reception of a provisional response
PUBLISH	Uploads information to a server
SUBSCRIBE	Requests to be notified about a particular event
UPDATE	Modifies some characteristic of a session
MESSAGE	Carries an instant message
REFER	Instructs a server to send a request

**Table 4.4** SIP Methods

### 4.2.4 Session Description

So far, we have seen the SIP body message where the information about the client and the server are contained. The details of the session are not described using SIP. Rather, the body of a SIP message contains a description of the session, encoded in some other protocol format, called Session Description. SIP can use different formats to describe the multimedia session. It is independent of the format of the objects it transports.

An example of session description is:

- *Subject*: Conference about Internet Communication
- *Time*: 12 April from 9:00 to 12:00
- *Location*: La Sapienza

The most common format used by SIP, however, is the Session Description Protocol (SDP), a textual format for describing unicast and multicast multimedia sessions. It contains information about codec, ports and protocol to be used for transmitting media to the caller. A caller can use this information to invite a callee to participate in an existing multicast session. For example, the information found in SDP is sufficient to allow a caller to send and receive audio immediately. SDP enables a user to indicate the ability to send and receive with multiple audio and video codecs, and SDP can also rank those codecs in preference of usage. The SDP lines consist of *type = value*, where *type* is always one character long. The next table shows all the types defined by SDP:

Type	Meaning
v	Protocol version
b	Bandwidth information
o	Owner of the session and session identifier
z	Time zone adjustments
s	Name of the session
k	Encryption key
i	Information about the session or the media line
a	Attribute lines
u	URL containing a description of the session
t	Times when the session is active and will be repeated
e	E-mail address to obtain the information about the session
p	Phone number to obtain information about the session
m	Media line
c	Connection information

**Table 4.5** SDP types

For example, Alice's client wants to send an INVITE message to Bob to talk about the holidays. Alice's IP address is 123.4.5.6 and she wants to receive audio via the port number 4561. The codec audio that Alice supports is G.711, which corresponds to the number 0 in the example:

```

v =0
o =Alice 2649376915 734564836585 IN IP4 123.4.5.6
s =Talk about the next holidays
c =IN IP4 123.4.5.6
t =0 0
m=audio 4561 RTP/AVP 0
a =sendrecv

```

As we can see in this example, an SDP description consists of two parts: session-level information and media-level information. The first part (before the m-line) provides version and user identifiers (v-line and o-line), the subject of the session (s-line), Alice's IP address (c-line) and the time of the session (t-line). Note that in this case the session is supposed to take place at the moment this session is received, which is why the t-line is t = 0 0. The second part is stream-specific and consists of an m-line and a number of optional a-lines that provide further information about the media stream. The a-lines in the example indicates that the stream is bidirectional (user and server receive media).

### 4.2.5 Establishment of a SIP Session

SIP supports five ways of establishing and terminating multimedia sessions:

- *User location*: determination of the end system to be used for communication
- *User availability*: determination of the willingness of the called party to engage in communications
- *User capabilities*: determination of the media and media parameters to be used
- *Session setup*: "ringing", establishment of session parameters at both called and calling party
- *Session management*: including transfer and termination of sessions, modifying session parameters, and invoking services

To establish a session the caller sends an INVITE message addressed to the callee's SIP URI, which he/she wants to contact. The INVITE message contains a number of header fields that provide additional information about the message (the destination address, the caller's address, session information, etc.) and the session description. The session is established when the caller receives the OK response by the callee. The 200 (OK) message contains a message body with the SDP media description of the type of the session that the callee is willing to establish with the caller. As a result, there is a two-phase exchange of SDP messages - the first in the request and the second in the response. This two-phase exchange provides basic negotiation capabilities and is based on a simple request/response model of SDP exchange. Now that we have introduced the generic information about the session establishment, let us give some examples of how SIP works.

### 4.2.5.1 Message Flow for Session Establishment

First of all, to establish a multimedia session, Alice must register her current location with the REGISTER message at the `domain.com` by sending a REGISTER request, indicating that all messages addressed to the URI specified in the *To* header field

```
sip:Alice.Bianchi@domain.com
```

must be forwarded to the URI specified in the CONTACT header field

```
sip:Alice@pc1.home.com
```

The OK response by the `domain.com` confirms that the request has been registered. If Bob wants to contact Alice to talk about a meeting, he sends an INVITE message to Alice's public URI. The proxy at the `domain.com` routes the INVITE request at Alice's current location. Alice accepts the invitation by sending a *200 OK* response. The proxy at the `domain.com` routes the *200 OK* message at the Bob's URI.

Note that in the *200 OK* response Alice specifies her current location in the Contact header field. This header field is used by Bob to send subsequent messages directly to Alice, bypassing the proxy server. As soon as the session has been established, Bob and Alice can talk about whatever they want. If, in the middle of the session, they want to make any changes to the session (e.g., add video), they should issue another INVITE request with an updated session description.

When Bob and Alice finish their conversation, Bob sends a BYE request directly to Alice without interaction with the proxy. Alice confirms the request with an OK response. Session description is summarised in Figure 4.2.

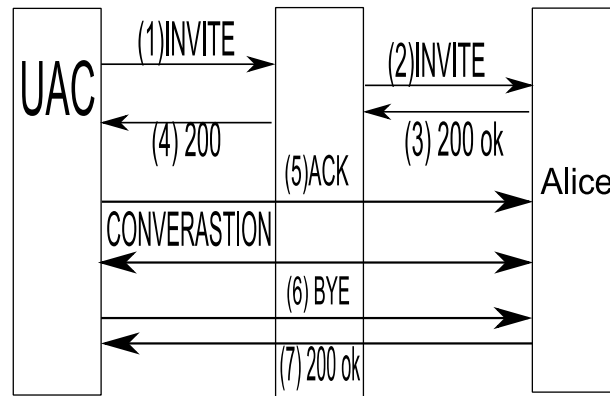


Fig. 4.2 Session Description.

### 4.2.5.2 Home Phone

One of the most interesting technical challenges for IP telephony is to mimic standard residential phone services. In particular, IP telephony requires the following features:

- All phones in a home must ring if someone calls its number.
- When one of the lines is picked up, all other phones in the home must stop ringing.
- A user can join an existing call by picking up any other telephone in the home.
- A home can have multiple lines, enabling a user on another handset to initiate a new call while one or more are in progress.
- All users involved in a single call are essentially involved in a multi-party conference call, and are thus able to hear each other.

A simple example further explains these features: Alice sends an invite message to Bob's parents. When the network server receives the message, it consults the database to find the list of contact addresses, each of which constitutes a single extension of the line. The server thus forks and sends out the INVITE messages at all members of the family and the lines ring at each address. When a user picks up the call (Member 3), an acceptance message is sent back to the server. When a forking proxy receives a call acceptance, it should send a cancel request on all unanswered members of the family (Member 1 and Member 2). Then, the server forwards the call acceptance. After that, if another callee picks up the call, an acceptance message is acknowledged by the server and the new participants join the call. To terminate the session, the caller sends a BYE message that is forwarded by the proxy to the two currently active lines.

### 4.2.5.3 Personal Mobility

Alice is a university researcher. So, in addition to having a computer at home (sip: Alice.Bianchi@domain.com), she also has two location addresses at the university, at the laboratory (sip: Alice@lab.uni.com) and at the office (sip: Alice@office.uni.com). When Alice is at the university, she sends a REGISTER message to the domain.com server listing the university address (sip: Alice@uni.com) as a forwarding address. Once at the university, Alice registers both her lab and office machine with the registration server of the university. Now, if someone sends a request to Alice's public URL (sip: Alice.Bianchi@domain.com), the domain.com server checks the current location of Alice in the database and forwards the request to the university server. As soon as the request arrives, the university server looks up the database and determines that Alice has two potential means of contact. The server then forks and sends the request to both the lab and office machines at the same time, causing the ringing of the phones. When Alice picks up the call, an acceptance message is sent back to the server and the session is established, as we have seen above. We have a different situation if Al-

ice forgets to register the lab machine, for instance, and restarts her user agent in the lab to forward the call to her `domain.com` server. When the lab phone receives the request, according to its outdated configuration, it forwards it to the `domain.com` server. Using the loop detection capabilities in SIP, this server determines that an error has occurred and returns an error response to the lab machine. In turn, it returns an error code to the university server. If, in the meantime, Alice responds to the call from the office, the university server also receives an acceptance message. Having now received both responses, the server forwards the acceptance answer establishing a session.

#### ***4.2.6 SIP's Extension***

SIP's extension negotiation mechanism uses three header fields: *Supported*, *Required*, and *Unsupported*. When a SIP session is being established, the user agent client lists in the *Required* header field all the names of the extensions it wants to use for that session, and all the names of the extensions it supports are not listed previously in a *Supported* header field. When the user agent server receives the *Required* header field, it checks the list and, if it does not support any of the extensions listed, it sends back an error message specifying that the session could not be established. This error response contains an *Unsupported* header field listing the extensions the user agent server did not support. If the user agent server supports all the required extensions, it should decide whether or not it wants to use any extra extensions and, if so, it includes the option tag for the extension in the *Required* header of its response. If this option was included in the *Supported* header field of the request, the session will be established. Otherwise, the user agent server includes the extension that is required by the server in a *Required* header field in the error response.

### **4.3 The Next Steps In Signalling (NSIS)**

#### ***4.3.1 Background and main Characteristics***

The IETF's Next Steps in Signalling Working Group is responsible for standardising an IP signalling protocol with QoS signalling as the first use case. This working group will concentrate on a two-layer signalling paradigm. The intention is to reuse, where appropriate, the protocol mechanisms of RSVP, while at the same time simplifying it and applying a more general signalling model. The existing work on the requirements, the framework and analysis of existing protocols will be completed and used as input for the protocol work. The NSIS WG is developing a transport layer signalling protocol for the transport of upper layer signalling. In order to support a toolbox or building block approach, the two-layer model will be used

to separate the transport of the signalling from the application signalling. This allows for a more general signalling protocol to be developed to support signalling for different services or resources, such as NAT, firewall traversal, and QoS resources. The initial NSIS application will be an optimised RSVP QoS signalling protocol. The second application will be a middle box traversal protocol. Security is a very important point for NSIS and the working group will study and analyse the threats and security requirements for signalling. Compatibility with authentication and authorisation mechanisms such as Diameter, COPS for RSVP (RFC 2749) and RSVP Sessions will be addressed. NSIS is a signalling protocol framework for conveying information about data flows along their path in the network, and interacting with nodes along the data path. Moreover, the NSIS messages pass directly through the same nodes as the data and only unicast data flows are considered. The intention is that the components of the NSIS protocol suite will be usable in different parts of the Internet, for different needs, without requiring a complete end-to-end deployment (signalling is intended not only for QoS). This flexibility is achieved by dividing the signalling protocol stack in two layers: a generic (lower) layer and an upper layer specific for each signalling application.

### ***4.3.2 Background and Main Characteristics***

In the signalling architecture, the messages are only received, processed and sent by Signalling Entities (SE) that can be placed on all devices on the data path (to support signalling purposes) or on some devices not included on the data path. Two different signalling architectures can be realised (distributed and centralised) and will be analysed in the following sections. In the distributed signalling architecture, SEs are placed on all devices (e.g., routers) on the data path. Thus, all devices are signalling-aware and take part in signalling. In the centralised signalling architecture, signalling is managed by a centralised SE in a domain. This architecture reduces the signalling charge on the interior devices in the domain.

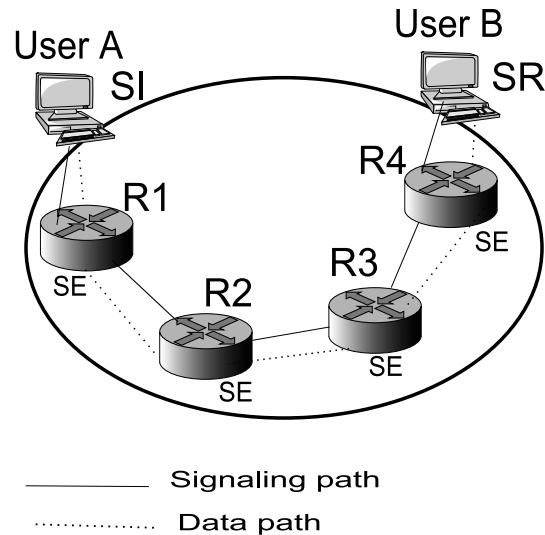
#### **4.3.2.1 Signalling Entities (SE)**

A Signalling Entity (SE) is the function that implements the signalling protocol. It can be placed in a network device (e.g., router, policy server) or in end systems. SE can support many signalling applications (for instance resource reservation). A Signalling Initiator (SI) is the SE that initiates the signalling, while Signalling Responder (SR) is the SE that is at the end of the signalling path and terminates the signalling. Signalling Forwarder (SF) is a SE that is on the signalling path between SI and SR. SF receives, processes and forwards signalling messages from SI to SR. Finally, Signalling Controller (SC) is the centralised SE in a domain. A SC is responsible for receiving, processing and sending signalling messages in a domain

and it is used in the centralised signalling architecture to reduce the charge on the edge routers and interior routers.

#### 4.3.2.2 Distributed Signalling Architecture

In the distributed signalling architecture, SEs are placed in the network devices on the data path and the signalling messages pass through the same network devices as the data packets. Figure 4.3 shows a simple example of the architecture. After a trigger from an application, the SE in user A initiates the signalling for a data flow from A to B via nodes R1, R2, R3 and R4. Signalling entities are installed on all routers between two end points to support the signalling. In the case the request of the user must be done by multi-domains, signalling will be done between the devices at the edge of the domain and the interior devices. The advantage of this architecture is that nodes on the data path can receive, process and send signalling messages. When there is a change on the network (e.g., overload, errors), the nodes can notify this to other entities in the network and rapidly adapt themselves to the change.



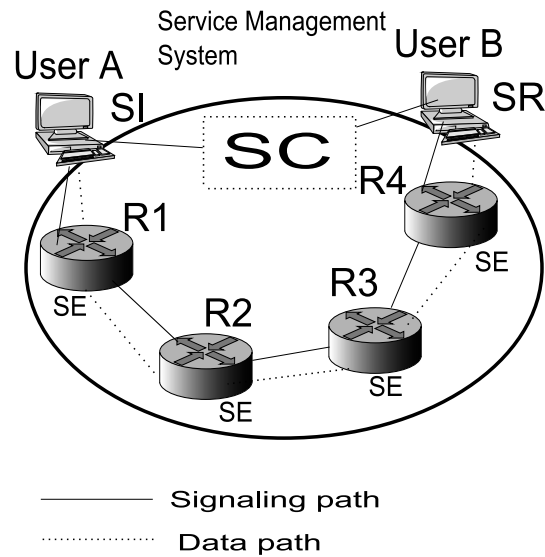
**Fig. 4.3** Distributed signalling architecture.

Every router uses the same routing mechanism for data packets to route signalling messages. As a result, signalling message propagation does not require global information for routing signalling messages. When a routing change happens (e.g., handover in the case of a mobile user), the network devices can detect this change and rapidly establish a new signalling path that follows the data path. The disadvantage of the distributed architecture is that an SE must be installed on all network

devices on the data path with consequently increased load of signalling on the intermediate nodes between two end points. Moreover, when the number of user flows increases, the information of state per flow contained in intermediate nodes can also increase. This could cause serious scalability problems.

#### 4.3.2.3 Centralised Signalling Architecture

In the centralised signalling architecture, a particular SE, called SC, (Signalling Controller) is in charge of managing the signalling in the domain. A SC can be installed on a centralised service management system (SMS), which manages services in a domain. It can also be installed in resource management server, on a policy server, or in any server in a domain. Figure 4.4 shows the centralised signalling architecture.



**Fig. 4.4** Centralised signalling architecture.

User A can make direct contact with the SC to ask the network to support a signalling application. When the SC receives signalling messages from user A, it transfers them to the SMS that knows the key information on the domain (e.g., topology of domain, routing table, resource availability of devices in the network) and is able to answer the request from the user. Finally, the SMS can invoke signalling entities or other entities implementing the request of user by configuring devices in the network.

### 4.3.3 Overview of Signalling Scenarios and Protocol Structure

#### 4.3.3.1 Layer Model for the Protocol Suite

In order to achieve a modular solution for the NSIS requirements, the NSIS WG proposed a split-layer protocol suite structured in two layers:

The NSIS Transport Layer Protocol, named General Internet Signaling Transport (GIST) [Schulzrinne2006], is responsible for moving signalling messages among network entities. This process should be independent of the signalling applications. The NSLP (NSIS Signalling Layer Protocol) contains the specific functionalities of the signalling applications. As described in the following subsection, this two-layer protocol model allows the support of various signalling applications, such as QoS [Manner2006] and Network Address Translation (NAT) & Firewall (FW) [Stiemerling2006] (see Figure 4.5).

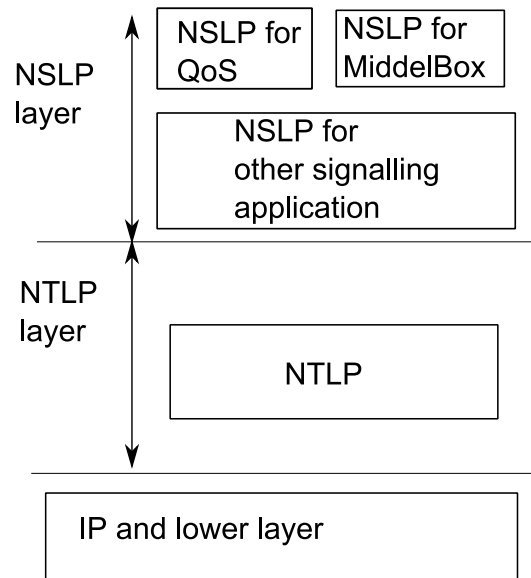


Fig. 4.5 Protocol Signalling Architecture.

#### 4.3.3.2 Signalling Application Properties

For some signalling applications and scenarios, signalling may only be considered for a uni-directional data flow. However, in other cases there may be interesting relationships between the signalling for the two flows of a bi-directional session. An example is QoS for a voice call. Note that the path may be different for two

directions, due to asymmetric routing. In the basic case, bi-directional signalling can simply use a separate instance of the same signalling mechanism in each direction. In constrained topologies where parts of the route are symmetric, it may be possible to use a more unified approach to bidirectional signalling, e.g. carrying the two signalling directions in common messages. This optimisation might be used for example to make mobile QoS signalling more efficient. In either case, the correlation of the signalling for the two flow directions is carried out in the NSLP. The NTLP would simply be enabled to bundle the messages together. Moreover, to provide additional flexibility in defining the objects carried by the NSLP such that only the objects applicable in a particular setting are used. One approach for reflecting the distinction is that local objects could be put into separate local messages that are initiated and terminated within one single domain; an alternative is that they could be "stacked" within the NSLP messages that are used anyway for inter-domain signalling. We are assuming that the NTLP provides a simple message transfer service, and any acknowledgments or notifications it generates are handled purely internally (and applied within the scope of a single NTLP peer relationship). However, we expect that some signalling applications will require acknowledgments regarding the failure/success of state installation along the data path, and this will be an NSLP function. Acknowledgments can be sent along the sequence of NTLP peer relationships toward the signalling initiator, which relieves the requirements on the security associations that need to be maintained by NEs and can allow NAT traversal in both directions (if this direction is toward the sender, it implies maintaining reverse routing state in the NTLP). In certain circumstances, e.g. trusted domains, an optimisation could be to send acknowledgments directly to the signalling initiator outside the NTLP, although any such approach would have to take into account the necessity of handling denial of service attacks launched from outside the network. The semantics of the acknowledgment messages are of particular importance. A NE sending a message could assume responsibility for the entire downstream chain of NEs, indicating for instance the availability of reserved resources for the entire downstream path. Alternatively, the message could have a more local meaning, indicating for instance that a certain failure or degradation occurred at a particular point in the network. Notifications differ from acknowledgments, because they are not (necessarily) generated in response to other signalling messages. This means that it may not be obvious to determine where the notification should be sent.

### ***4.3.4 The NSIS Layer Transport Protocol***

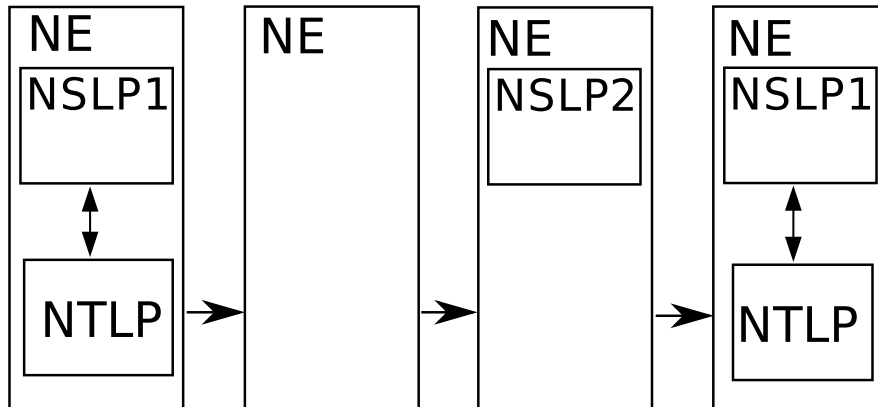
#### **4.3.4.1 GIST Description**

The GIST layer is responsible for the transport of signalling messages. When a signalling message is ready to be sent, it is given to the GIST layer along with information about the flow it is for; it is then up to the GIST layer to get the message to the next network element (NE) along the path (downstream, in the flow direction

from the source to the destination; or upstream, in the opposite direction of the flow from the destination to the source), where it is received and the local GIST responsibility ends.

In the receiving NE, the GIST either forwards the message directly to the next hop or, if there is an appropriate signalling application, passes it upwards for further processing; the signalling application can then generate another message to be sent via GIST.

Figure 4.6. is an example of the NSIS two-layer architecture, showing two different signalling applications and how NEs handle the signalling messages accordingly.



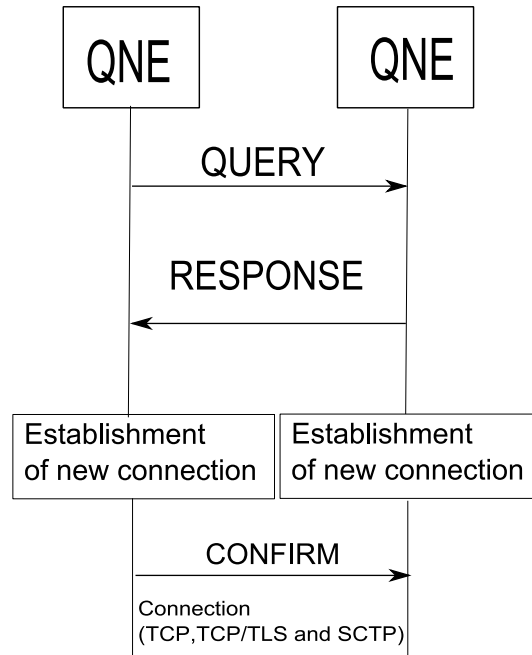
**Fig. 4.6** Signalling with Heterogeneous NSLPs.

In this example, the first NE in the path has the NTLP and NSLP 1. NSLP 1 generates a signalling message and sends it to the local NTLP so that it can be forwarded to the flow destination. In the next NE, there are no NSLPs, so the message is forwarded without being processed by the NTLP. In the next hop the NTLP and NSLP 2 are present. Since the message NSLP ID does not correspond to the connected NSLP 2, the message is forwarded to the flow destination again. In the next hop there is again NTLP and NSLP 1. Since the correct NSLP is available, NTLP sends the message to NSLP 1. When NSLP receives the message from NTLP, it processes it accordingly.

GIST allows two modes of operation, the Datagram mode (D-mode) and the Connection mode (C-mode). D-mode uses UDP to encapsulate the messages and is used for small and infrequent messages. All Query messages must be sent in D-mode. The C-mode uses TCP or any other stream or message oriented transport protocol (currently only Stream Control Transmission Protocol, SCTP [83], is being researched in addition to TCP) which allows GIST to support reliability and security (for example using Transport Layer Security, TLS, [84] over TCP) in the message transport.

To meet the routing requirement, GIST defines a 3-way handshake to set up the necessary connection with the adjacent peers. This 3-way handshake contains

a QUERY, a RESPONSE and an optional CONFIRM message. Figure 4.7 describes the 3-way handshake process between two entities that support GIST. In this handshake, the QUERY message is the first message to be sent. This message is always sent in D-mode and with the IP Router Alert Option (RAO) [85] flag active. With this flag, the message sent by GIST travels along the network and every router that checks this flag analyses the packet content. GIST entities in the network analyse all packets flagged with the IP RAO and process all QUERY messages.



**Fig. 4.7** GIST 3-way handshake.

When a QUERY message is intercepted, the NSLP ID is checked and if the corresponding NSLP is present, the message is processed by GIST. Otherwise, the message is forwarded to the flow destination so that other GIST entities can intercept the message, or the destination is reached.

The purpose of the QUERY message is the discovery of the next NSIS hop in the path and the transport of a proposal for the establishment of a connection between the two entities. These messages contain a QUERY Cookie object and when a connection is requested, the association characteristics (for instance the protocol and port to use in the association) are also present. The QUERY Cookie is a security payload that is carried by the QUERY message, which allows the detection and prevention of several security problems in the handshake.

GIST entities that receive a QUERY message need to reply with a RESPONSE message. This message is sent to the previous GIST entity by getting its identity

from the QUERY message. This message contains a RESPONSE Cookie, which is a cryptographic key based on the received QUERY Cookie. This RESPONSE Cookie increases the security of the protocol, allowing the upstream hop to check if the Response message is not sent by a fake NSIS entity. If the received QUERY message requested an association, the RESPONSE message also includes the association response.

If the association between the two GIST entities is requested (by the NSLP or by a local GIST decision/configuration) when the RESPONSE message is received in the upstream GIST, the association is created and a CONFIRM message is sent to the downstream GIST using the association. This association can be supported through TCP, secure TCP with Transport Layer Security (TCP/TLS) or the Stream Control Transmission Protocol (SCTP). Only after the CONFIRM message is sent, the NSLPs payloads can start flowing between the two GISTs.

The associations created via the 3-way handshake can be re-used for different sessions and NSLPs when the downstream peer and the association characteristics are the same. Even though the 3-way handshake is needed for each new session, the RESPONSE and CONFIRM messages are sent using the already established association.

GIST was designed as a soft-state protocol to manage all the messages and associations. GIST uses states for each action occurred in the system and associates a timer to each state. Each time the state is updated, the timer is restarted. If the state is not updated, the timer expires and the state is removed. GIST has two main state tables: Message Routing State (MRS) and Message Association State (MAS). The MRS is responsible for managing individual flows and the MAS is responsible for managing the associations between individual peers. When a timer expires (if no message is received for the corresponding flow or association), the state automatically is removed from the state tables. If a state is required again, a new handshake is needed and a new association must be created.

After the handshake is completed, data messages can be sent with the NSLP payload. GIST does not check the NSLP payload, the only processing done to the message is the decrement of the message hop-count, the corresponding states are refreshed (MRS and MAS) and finally the payload is sent to the corresponding NSLP.

QoS NSLP is described in the following subsection. This NSLP is one example of an NSLP that uses GIST as its transport protocol.

#### 4.3.4.2 Signalling for Quality of Service

In the case of signalling for QoS, we can apply all the basic NSIS concepts. In addition, there is an assumed directionality of the signalling process in that one end of the signalling flow takes responsibility for actually requesting the resources. This leads to the following definitions:

- The protocol employs a client/server model where the PEP requests, updates and deletes to the remote PDP and the PDP returns decisions back to the PEP.

- QoS NSIS Responder (QNR): the signalling entity that acts as the endpoint for the signalling and can optionally interact with applications as well.
- QoS NSIS Forwarder (QNF): a signalling entity between a QNI and QNR, which propagates NSIS signalling further through the network.
- COPS provides message level security for authentication, replay protection and message integrity. COPS can also reuse existing protocol for security (IPSEC) or to authenticate and secure the channel between the PEP and the PDP.

Each of these entities will interact with a resource management function (RMF), which actually allocates network resources (router buffers, interface bandwidth, etc.). Note that there is no constraint on which end of the signalling flow should take the QNI role: with respect to the data flow direction it could be at the sending or receiving end.

#### 4.3.4.3 Protocol Message Semantics

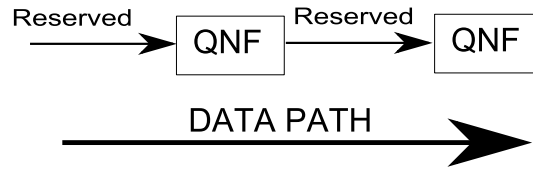
The QoS NSLP will include a set of messages to reserve resources along the signalling path. A possible set of message semantics for the QoS NSLP is shown below. Note that the "direction" column in the table below only indicates the "orientation" of the message. Messages can be originated and absorbed at QNF nodes as well as the QNI or QNR. An example might be QNFs at the edge of a domain exchanging messages to set up resources for a flow across it. Note that it is left open whether the responder can release or modify a reservation, during or after setup. This seems mainly a matter of assumptions about authorisation, and the possibilities might depend on resource type specifics. The table also explicitly includes a refresh operation. This does nothing to a reservation except extend its lifetime, and is one possible state management mechanism.

Operation	Direction	Description
Request	I to R	Create a new reservation for a flow
Modify	I to R (R to I)	Modify an existing reservation
Release	I to R (R to I)	Delete (tear down) an existing reservation
Accept/Reject	R to I	Confirm or reject a reservation request
Notify	I to R (R to I)	Report an event detected within the network
Refresh	I to R	State Management

**Table 4.6** NSIS Protocol Messages

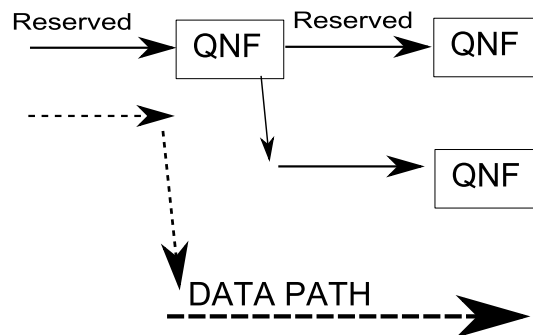
#### 4.3.4.4 Route Changes and QoS Reservations

The normal operation of the NSIS protocol will lead to the situation depicted in Figure 4.8. , where the reserved resources match the data path



**Fig. 4.8** Normal NSIS Protocol Operation.

A route change can occur while such a reservation is in place. The route change will be installed immediately and any data will be forwarded on the new path. This situation is depicted in Figure 4.9.



**Fig. 4.9** Route Change.

Resource reservation on the new path will only be started when the next control message is routed along the new path. This means that there is a certain time interval during which resources are not reserved on (part of) the data path, and certain delay or drop-sensitive applications will require this time interval to be minimised. Several techniques to achieve this could be considered. As an example, RSVP has the concept of local repair, where the router may be triggered by a route change. In that case the RSVP node can start sending PATH messages directly after the route has been changed. Another approach would be to pre-install back-up state, and it would be the responsibility of the QoS-NSLP to do this, but mechanisms for identifying back-up paths and routing the necessary signalling messages along them are not currently considered in the NSIS requirements and framework. It is not guaranteed that the new path will be able to provide the same guarantees that were available on the old path. Therefore, it might be desirable for the QNF to wait until resources have been reserved on the new path before allowing the route change to be installed

(unless of course the old path no longer exists). However, delaying the route change installation while waiting for reservation setup needs careful analysis of the interaction with the routing protocol being used, in order to avoid routing loops. This solution adapts to a route change when a route change creates congestion on the new routed path.

#### 4.3.4.5 Resource Management Interactions

The QoS NSLP itself is not involved in any specific resource allocation or management techniques. The definition of an NSLP for resource reservation with Quality of Service, however, implies the notion of admission control. For a QoS NSLP, the measure of signalling success will be the ability to reserve resources from the total resource pool that is provisioned in the network. Resource Management Function (RMF) is responsible for all resource provisioning, monitoring and assurance functions in the network. A QoS NSLP will rely on the RMF to do resource management and to provide input for admission control. In this model, the RMF acts as a server toward the client NSLP(s). It should be noted, however, that the RMF may in turn use another NSLP instance to do the actual resource provisioning in the network. In this case, the RMF acts as the initiator (client) of an NSLP. This essentially corresponds to a multi-level signalling paradigm with an "upper" level handling Internet working QoS signalling, possibly running end-to-end, and a "lower" level handling the more specialised intra-domain QoS signalling, running between just the edges of the network. Given that NSIS signalling is already supposed to be able to support multiple instances of NSLPs for a given flow, and limited scope (e.g., edge-to-edge) operation, it is not currently clear that supporting the multi-level model leads to any new protocol requirements for the QoS NSLP.

#### 4.3.4.6 NAT & Firewall NSLP

The NAT and Firewall (NATFW) NSIS Signalling Layer Protocol (NSLP) [86] is being defined in the NSIS IETF Working Group to provide dynamic configuration of NATs and firewalls along the data path of a specific flow. NATs and firewalls are devices in the network that may create obstacles to some applications. Applications such as IP telephony and peer-to-peer applications generate traffic that is unable to traverse these obstacles. This NSLP is designed to dynamically configure NATs and firewalls along the data path. It is required to load firewall rules with an action that allows data flow packets passing the firewall. In NAT, it is required to create NAT bindings to the data flow packets.

A simple scenario of the NATFW NSLP between a sender and a receiver with two middle-boxes (device in the network that intercepts the flow of packets between end hosts and performs control actions, like NATs and firewalls) is depicted in Figure 4.10. In this example, the NAT or firewall representation is integrated within the

NSIS entity, but they can be two separate entities where the NSIS entity requests the middle-box to change configuration.

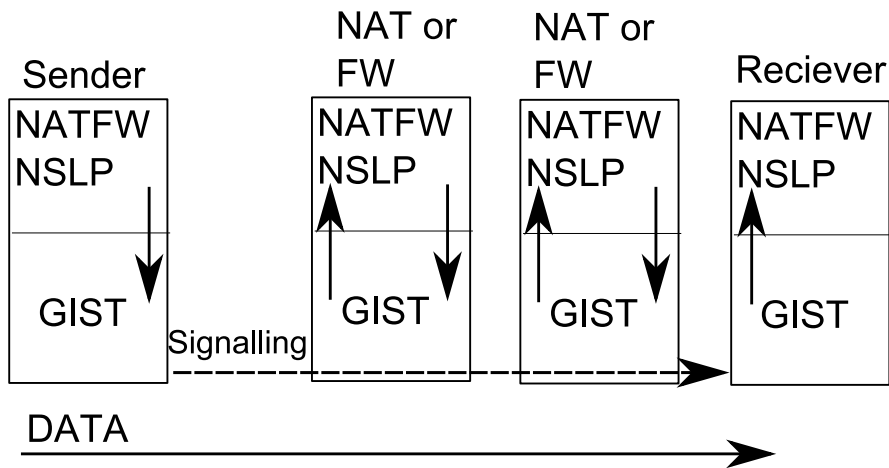


Fig. 4.10 Simple NATFW NSLP overview.

This example describes a source host (sender) that generates a NATFW NSLP signalling message and sends it to the destination host (receiver). This message follows the data path and every NSIS entity along the data path with the NATFW NSLP application processes the message. Based on the processing of the message, the NATFW NSLP changes the middle-boxes accordingly and forwards the message to the receiver. After all the middle-boxes are configured for the specific flow, the data can start to flow with no obstacles.

If the sender or the receiver is not NATFW NSLP aware, the NSLP specification allows the usage of a NATFW NSLP proxy. This is referred to as proxy mode operation.

The described scenario does not work if the receiver is behind NATs. The sender cannot address the receiver directly. To solve this issue, a mode of operation was defined as the RESERVE-EXTERNAL-ADDRESS (REA) mode. This mode allows a receiver to locate upstream NATs and pre-allocate a public address.

NATFW NSLP signalling messages contain general information (like IP address, ports, protocol) and policy rules. The policy rules are abstractions of the network equipment policy rules that need to be installed. The request initiator generates the abstract policy rules and in each NATFW NSLP in the path these rules must be mapped to the particular NAT or firewall rules. This mapping is vendor and model dependent.

To provide the described functionalities, five message types have been defined: CREATE, RESERVE-EXTERNAL-ADDRESS, TRACE, NOTIFY and RESPONSE. The CREATE message creates, changes, refreshes and deletes NATFW NSLP sessions on the data path from the sender to the destination. The REA mes-

sage is forwarded from the receiver to the edge NAT to allow inbound CREATE messages to be forwarded to the receiver. This message reserves an external address (and a port number if needed) in the edge router and requests the configuration of all intermediate middle-boxes (between the receiver and the edge NAT). The TRACE message gathers information from all NATFW NSLP in the data path. The NOTIFY message is an asynchronous message used by the NSLP to notify upstream peers about specific events. The RESPONSE message is a response to CREATE, REA and TRACE messages.

All messages received by a NATFW NSLP before being processed are checked as to whether the requested actions are authenticated and authorised. For this purpose all the NATFW NSLP messages carry the NATFW CREDENTIAL object. The information and structure of this object depends on the authentication and authorisation model used in each domain.

## 4.4 Common Open Policy Service (COPS)

### 4.4.1 COPS Overview

COPS (Common Open Policy Service specified in the RFC 2748 [87]) is a simple query and response protocol that can be used to exchange policy information between a policy server (Policy Decision Point or PDP) and its clients (Policy Enforcement Points or PEPs). COPS supports two models of policy control: outsourcing and configuration (also known as provisioning). In the outsourcing model the PEP contacts the PDP every time a policy decision has to be made. The PDP makes the decision and communicates it back to the PEP, which enforces it. In the configuration model the PDP configures the PEP with the policy to be used. The PEP stores the policy received from the PDP locally and uses it to make decisions instead of contacting the PDP every time a new event occurs. The main characteristics of COPS protocol include:

- The protocol employs a client/server model where the PEP requests, updates and deletes to the remote PDP and the PDP returns decisions back to the PEP.
- The protocol uses TCP (Transfer Control Protocol) as its transport protocol for reliable exchange of messages between policy clients and a server.
- The protocol is extensible in that it is designed to leverage off self-identifying objects and can support diverse client specific information without requiring modifications to the COPS protocol itself.
- COPS provides message level security for authentication, replay protection and message integrity. COPS can also reuse existing protocol for security (IPSEC) or to authenticate and secure the channel between the PEP and the PDP.
- The protocol allows the server to push configuration information to the client, and then allows the server to remove such state from the client when it is no longer applicable.

- The protocol is stateful in two main aspects:
  - The *Request/Decision* state is shared between client and server. The requests from the client PEP are installed or remembered by the remote PDP until they are explicitly deleted by the PEP. At the same time, decisions from the remote PDP can be generate asynchronously at any time for a currently installed request state.
  - In the *Inter-associated* state, the server may respond to new queries differently because of previously installed *Request/Decision* state(s) that are related.

#### 4.4.2 Basic Model

The basic model of COPS and the framework of policy-based admission control are shown in the Figure 4.11. The network nodes can be routers, switches or hubs. The resources are allocated or released inside a node by PEP (Policy Enforcement Point). A node can be a policy ignorant node, which does not support the COPS protocol. The network nodes are grouped into administrative domains. There is always at least one policy server in each administrative domain. Inside the policy server there is the PDP (Policy Decision Point), which makes the final decision about the handling of the resources. There can also be a local PDP in the network node (see Figure 4.11. ).

The PEP may communicate with a policy server to obtain policy decisions or directives. It is responsible for initiating a persistent TCP connection to a PDP, for notifying the PDP when a request state has changed on the PEP and, finally, for the deletion of any state that is no longer applicable. When the PEP sends a configuration request, it expects the PDP to continuously send the named units of configuration data until it is successfully installed, and then the PEP should send a report message to the PDP confirming the installation. The policy protocol is designed to communicate self-identifying objects that contain the data necessary for identifying request states, establishing the context for a request, identifying the type of the request, referencing previously installed request, relaying policy decisions, reporting errors and providing message integrity. When a failure is detected, the PEP must try to reconnect to the remote PDP or attempt to connect to a backup alternative PDP. While disconnected, the PEP should revert to making local decision with LPDP; once the connection is re-established, the PEP is expected to notify the PDP of any deleted state or new event that passed local admission control after the connection was lost. The PDP can also send unsolicited decisions to the PEP, e.g., the PDP can force the PEP to change previous decisions. Respectively, the PDP can send information for accounting or monitoring purposes to the PEP. The PEP outsources the decision making to the PDP. Although the PEP can make local decisions with the LPDP, the final decision is made by the remote PDP. There are three different types of outsourcing events that need the decision from the PDP and one not outsourcing type (Figure 4.12. ). These define the context of the each request and decision. The

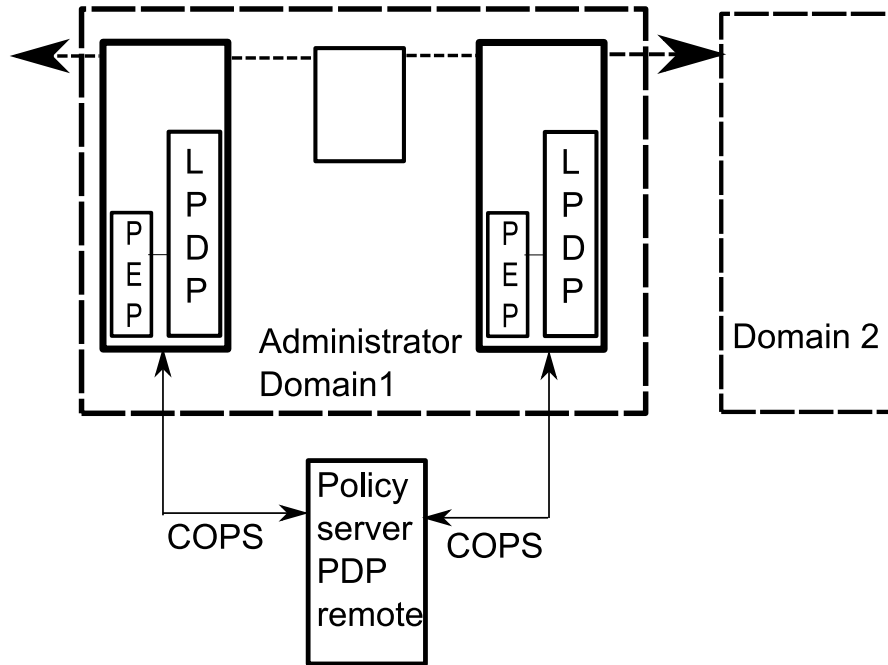


Fig. 4.11 COPS in the framework of policy-based admission.

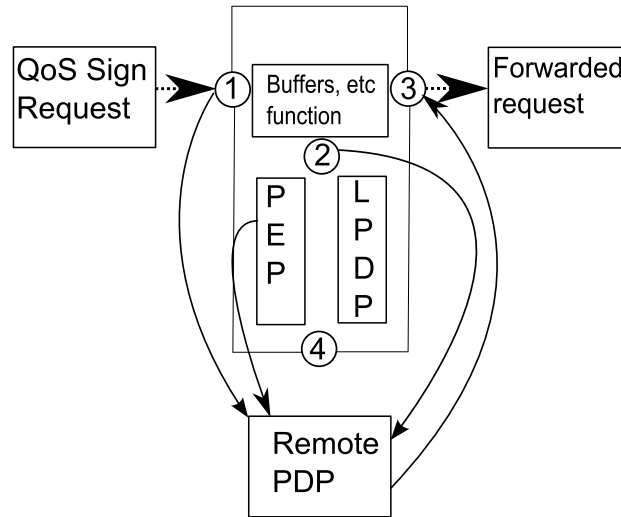
incoming message (e.g., RSVP message) causes the PEP to ask the PDP how to handle the incoming signal (1), e.g. to accept or reject it. The next step to be done is to allocate local resources (2) in order to process the message and finally the PEP enforces the forwarding decision made by PDP (3).

1. arrival of an incoming message
2. allocation of local resources
3. forwarding of an outgoing message
4. configuration information

The fourth type of event, which is not counted as an outsourcing event, is the request for configuration information from the remote PDP. The PEP can make the request, for example when it is booting up. The PEP can also ask for configuration information later, for instance when the PEP notices that it must install new instructions in order to handle a new module or interface.

#### 4.4.3 COPS Protocol

The COPS protocol is designed so that the messages are self-identifying policy objects, which contain policy elements. The policy elements are units of information



**Fig. 4.12** COPS processing of QoS requests.

necessary for the evaluation of policy rules. A single policy element may carry user or application identification, whereas another policy element may carry user credentials or credit card information. The policy elements themselves are expected to be independent of which QoS signalling protocol is used.

#### 4.4.3.1 COPS Header

COPS messages are binary-encoded and consist of the common header followed by a number of typed objects.

Version	Flags	Op Code	Client Type
Message Length			

**Table 4.7** The COPS common header.

The fields in the header are as follows:

- *Version*: 4 bits. COPS version number. Current version is 1.
- *Flags*: 4 bits. Defined the flag values (all other flags must be set to 0): 0x1 Solicited Message Flag
- *Bit*: This flag is set when the message is solicited by another COPS message.
- *Op Code*: 8 bits. Contains a message type, indicates the COPS operations, which are:

- 1 =Request (REQ)
  - 2 =Decision (DEC)
  - 3 =Report State (RPT)
  - 4 =Delete Request State (DRQ)
  - 5 =Synchronise State Req (SSQ)
  - 6 =Client-Open (OPN)
  - 7 =Client-Accept (CAT)
  - 8 =Client-Close (CC)
  - 9 =Keep-Alive (KA)
  - 10=Synchronise Complete (SSC)
- *Client-type*: 16 bits. It indicates that type of policy client. Interpretation of all encapsulated objects is relative to the client-type. Client-types that set the most significant bit in the client-type field are enterprise specific (these are client-types 0x8000 - 0xFFFF). For KA Messages, the client-type in the header must always be set to 0 as the KA is used for connection verification (not per client session verification).
  - *Message Length*: 32 bits. Size of message in octets including the standard COPS header and all encapsulated objects. Messages must be aligned on 4 octet intervals.

#### 4.4.3.2 COPS Specific Object Formats

The COPS header defines the message type. A number of policy elements follow after the header. These elements are used for transferring the information for decision-making and the actual decisions. The contents of the policy elements depend on the QoS signaling protocol, but the main structure of the COPS protocol remains intact. The policy elements, which are defined in the COPS protocol, conform to the following structure:

Length (octets)	C-Num	C-Type
Object Contents		

**Table 4.8** The COPS specific object format.

The length is a two-octet value that describes the number of octets (including the header) composing the object. If the length in octets does not fall on a 32-bit word boundary, padding must be added to the end of the object so that it is aligned to the next 32-bit boundary before the object can be sent on the wire. On the receiving side, a subsequent object boundary can be found by simply rounding up the previous stated object length to the next 32-bit boundary. Typically, C-Num (8 bits) identifies the class of information contained in the object, and the C-Type (8 bits) identifies the subtype or version of the information contained in the object. For instance, with In Interface (C-Num = 3) and Out Interface (C-Num = 4) objects C-Type = 1 means that interface is IPv4 and C-Type = 2 means that the interface is IPv6.

The possible values of the C-Num fields and their meanings are presented in Table 6.3.2.1. Every COPS specific object has respective contents of the object. This content is of variable length depending of the C-Num and C-Type. The messages between the PEP and the PDP are constructed from these objects. In the next subsection some examples of the messages are shown.

C Num	Name	Explanation
1	Handle	The Handle Object encapsulates a unique value that identifies an installed state. This identification is used by most COPS operations.
2	Context	Specifies the type of event(s) that triggered the query. Required for request messages.
3	In-Interface	This object is used to identify the incoming interface on which a particular request applies and the address where the received message originated.
4	Out-Interface	This object is used to identify the outgoing interface to which a specific request applies and the address for where the forwarded message is to be sent.
5	Reason Code	This object specifies the reason why a request state was deleted. It appears in the delete request (DRQ) message.
6	Decision	Appears in the reply as the decision taken by the PDP
7	LPDP Decision	May appear in the request as local point decision
8	Error	Identifies a particular COPS protocol error
9	Client Specific Info (SI)	This contains client-type specific information, e.g. the contents of RSVP Path-message, if the client is RSVP.
10	Keep-alive timer	Maximum time interval over which a COPS message has to be sent or received
11	PEP Identification	Allows client identification to the PDP, required for Client-Open messages
12	Report Type	Type of report in request state associated with handle
13	PDP Redirect Address	A PDP when closing a PEP session for a particular client-type may optionally use this object to redirect the PEP to the specified PDP server address and TCP port number.
14	Last PDP Address	Used from PEP to specify the PDP the last PDP that has accepted to open session since he last rebooted, when PDP sends Open message
15	Accounting timer	Optional timer value used to determine the minimum interval between accounting type reports
16	Message Integrity	Sequence of number used for authentication and validating COPS messages

**Table 4.9** COPS Specific objects.

#### 4.4.4 COPS Messages

A PEP and a PDP exchange COPS traffic over a TCP connection that is always initiated by the PEP. One PDP implementation per server must listen on a well-known TCP port number (COPS=3288 by IANA). The location of the remote PDP can either be configured or obtained via a service location mechanism. If a single PEP can support multiple client-types, it may send multiple Client-Open messages, each specifying a particular client-type to a PDP over one or more TCP connections.

*Request (REQ)* [PEP → PDP]: The PEP establishes a request state client handle for which the remote PDP may maintain state. The handle is the identification with which the PDP communicates. If there are local changes in the PEP, the PEP is responsible to inform the PDP about the changes. The Context object tells the context where all the other objects must be interpreted. ClientSI is the client specific information; it can be the contents of a message of some QoS signaling protocol. Also, the incoming and outgoing interfaces are depicted in the message. LPDPDecision contains the decisions that the local PDP has done and they must be verified, completed or overwritten by the remote PDP. There are 5 different kinds of remote and local PDP decisions. Flags indicate the normal request/decision, stateless data means local decision, which does not affect the state of the request, replacement data replaces the existing data in a signaled message, client specific decision data is to used to introduce additional decision types and, finally, named data contains configuration information. The instructions on the usage of the four last mentioned types should be specified in several documents on COPS extension for the given client-type.

*Decision (DEC)* [PDP → PEP]: The PDP responds to the REQ with a DEC message that includes the associated client handle and one or more decision objects grouped relative to a Context object and Decision Flags object type pair. It is required that the first decision message for a new/updated request will have the Solicited Message flag set (value = 1) in the COPS header. This avoids the issue of keeping track which updated request (that is, a request reissued for the same handle) a particular decision corresponds to. It is important that for a given handle there will be at most one outstanding solicited decision per request. This essentially means that the PEP should not issue more than one REQ (for a given handle) before it receives a corresponding DEC with the solicited message flag set. The PDP must always issue decisions for requests on a particular handle in the order they arrive, and all requests must have a corresponding decision. To avoid deadlocks, the PEP can always timeout after issuing a request that does not receive a decision. It must then delete the expired handle, and may try again using a new handle. The Decision message may include either an Error object or one or more context plus associated decision objects. COPS protocol problems are reported in the Error object (e.g., an error with the format of the original request including malformed request messages, unknown COPS objects in the Request, etc.). The applicable Decision objects depend on the context and the type of client. The only ordering requirement for decision objects is that the required Decision Flags object type must precede the other Decision object types per context binding.

*Report State (RPT)* [PEP → PDP]: The RPT message is used by the PEP to communicate to the PDP its success or failure in carrying out the PDP's decision, or to report an accounting related change in state. The Report-Type specifies the kind of report and the optional ClientSI can carry additional information per Client-Type. For every DEC message containing a configuration context that is received by a PEP, the PEP must generate a corresponding Report State message with the Solicited Message flag set, describing its success or failure in applying the configuration decision. In addition, outsourcing decisions from the PDP may result in a corresponding solicited Report State from the PEP depending on the context and the type of client. RPT messages solicited by decisions for a given Client Handle must set the Solicited Message flag and must be sent in the same order as their corresponding Decision messages were received. There must never be more than one Report State message generated with the Solicited Message flag set per Decision. The Report State may also be used to provide periodic updates of client specific information for accounting and state monitoring purposes depending on the type of the client. In such cases, the accounting report type should be specified, utilising the appropriate client specific information object.

*Delete Request State (DRQ)* [PEP → PDP]: This message indicates to the remote PDP that the state identified by the client handle is no longer available/relevant. This information will then be used by the remote PDP to initiate the appropriate house-keeping actions. The reason code object is interpreted with respect to the client-type and signifies the reason for the removal. The format of the Delete Request State message is as follows: It is important that when a request state is finally removed from the PEP, a DRQ message for this request state is sent to the PDP so the corresponding state may likewise be removed on the PDP. Request states not explicitly deleted by the PEP will be maintained by the PDP until either the client session is closed or the connection is terminated.

*Synchronise State Query (SSQ)*: This message indicates that the remote PDP requests the client (which appears in the common header) to re-send its state. If the optional Client Handle is present, only the state associated with this handle is synchronised. If the PEP does not recognise the requested handle, it must immediately send a DRQ message to the PDP for the handle that was specified in the SSQ message. If no handle is specified in the SSQ message, all of the active client state must be synchronised with the PDP. The client performs state synchronisation by re-issuing request queries of the specified client-type for the existing state in the PEP. When the synchronisation is complete, the PEP must issue a synchronise state complete message to the PDP.

*Client-Open (OPN)* [PEP → PDP]: The PEP uses the Client-Open message to tell the PDP what kind of client-types the PEP support. The PEP can also specify the last PDP to which the PEP connected. The PEPID is a symbolic name of the PEP, which identifies it inside the administrative domain. The identifier is an ASCII string that can be an IP address or DNS name of the PEP. The PEP can forward some additional client specific information to the PDP. The last PDP address describes the last PDP for which the PEP is still caching decisions. The integrity object is used if security is in use or when the security is wanted in the usage.

*Client-Accept (CAT)* [PDP → PEP]: The Client-Accept message is used to positively respond to the Client-Open message. This message will return to the PEP a timer object, indicating the maximum time interval between keep-alive messages. Optionally, a timer specifying the minimum allowed interval between accounting report messages may be included when applicable. If the PDP refuses the client, it will instead issue a Client-Close message. The Keep-Alive (KA) timer corresponds to maximum acceptable intermediate time between the generation of messages by the PDP and PEP. The timer value is determined by the PDP and is specified in seconds. A timer value of 0 implies no secondary connection verification is necessary. The optional Accounting (ACCT) timer allows the PDP to indicate to the PEP that periodic accounting reports should not exceed the specified timer interval per client handle. This allows the PDP to control the rate at which accounting reports are sent by the PEP (when applicable). In general, accounting type Report messages are sent to the PDP when determined appropriate by the PEP. The accounting timer is merely used by the PDP to keep the rate of such updates in check (i.e., preventing the PEP from blasting the PDP with accounting reports). Not including this object implies that there are no PDP restrictions on the rate at which accounting updates are generated. If the PEP receives a malformed Client-Accept message, it must generate a Client-Close message specifying the appropriate error code.

*Client-Close (CC)* [PEP → PDP, PDP → PEP]: The Client-Close message can be issued by either the PDP or PEP to notify each other that a particular type of client is no longer being supported. The Error object is included to describe the reason for the closing (e.g., the requested client-type is not supported by the remote PDP or client failure).

*Keep-Alive (KA)* [PEP → PDP, PDP → PEP]: The keep-alive message must be transmitted by the PEP within the period defined by the minimum of all KA Timer values specified in all received CAT messages for the connection. A KA message must be generated randomly between  $\frac{1}{4}$  and  $\frac{3}{4}$  of this minimum KA timer interval. When the PDP receives a keep-alive message from a PEP, it must echo a keep-alive back to the PEP. This message provides validation for each side that the connection is still functioning even when there is no other message exchange. Note: The client-type in the header must always be set to 0, as the KA is used for connection verification (not per client session verification). Both client and server may assume the TCP connection is insufficient for the client-type with the minimum time value (specified in the CAT message), if no communication activity is detected for a period exceeding the timer period. For the PEP, such detection implies the remote PDP or connection is down and the PEP should now attempt to use an alternative backup PDP.

*Synchronise State Complete (SSC)* [PEP → PDP]: The Synchronise State Complete is sent by the PEP to the PDP after the PDP sends a synchronise state request to the PEP and the PEP has finished synchronisation. It is useful, since the PDP will know when all of the old client state has been successfully re-requested and, thus, the PEP and PDP are completely synchronised. The Client Handle object only needs to be included if the corresponding Synchronise State Message originally referenced a specific handle.

### **4.4.5 Common Operation**

COPS supports two models of policy control: outsourcing and configuration (also known as provisioning). In the outsourcing model the PEP contacts the PDP every time a policy decision has to be made. The PDP makes the decision and communicates it back to the PEP, which enforces it. In the configuration model the PDP configures the PEP with the policy to be used. The PEP stores the policy received from the PDP locally and uses it to make decisions instead of contacting the PDP every time a new event occurs.

#### **4.4.5.1 Outsourcing Operation**

In the outsourcing scenario the PEP contacts the PDP every time a policy decision needs to be made. Since the request is stateful, the request will be remembered, or installed, on the remote PDP. The unique handle (unique per TCP connection and client-type), specified in both the request and its corresponding decision identifies this request state. The PEP is responsible for deleting this request state once the request is no longer applicable. The PEP can update a previously installed request state by re-issuing a request for the previously installed handle. The remote PDP is then expected to make new decisions and send a decision message back to the PEP. Likewise, the server may change a previously issued decision on any currently installed request state at any time by issuing an unsolicited decision message. At all times the PEP module is expected to enforce the PDP's decisions and notify the PDP of any state changes. An example of this model is COPS usage for RSVP. RSVP-enabled routers, which act as PEPs and use COPS client-type 1, query the PDP when an RSVP message is received. The client specifies which RSVP message was received (e.g., PATH or RESV) and the expected behaviour of the router. The PDP decides whether or not the requested reservation is acceptable, according to the policy of the domain and communicates its decision back to the PEP (e.g., remove the resources assigned to a particular flow).

#### **4.4.5.2 Configuration Operations**

In the configuration scenario (known as COPS-PR), the PDP provides PEPs with policies and the PEP applies these policies. The PDP will then potentially send several decisions containing named units of configuration data to the PEP. The PEP is expected to install and use the configuration locally. When the PDP no longer wishes that the PEP uses a piece of configuration information, it will send a decision message specifying the named configuration and a decision flags object with the remove configuration command. The PEP should then proceed to remove the corresponding configuration and send a report message to the PDP that specifies it has been deleted. When the client downloads this policy, it does not need to contact the server to make individual decisions. This makes COPS-PR a highly scalable

protocol; moreover, the COPS-PR could potentially be used to transfer configuration parameters beyond policy information.

#### 4.4.5.3 Security

In the typical exchanges between remote PDP servers and PEP clients there are also the COPS security messages. If COPS level security is required, it must be negotiated during the initial Client-Open/Client-Accept message exchange specifying a Client-Type of zero (which is reserved for connection level security negotiation and connection verification). Security can be initiated by the PEP, if the PDP accepts the PEP's security key and algorithm by validating the message digest using the identified key, the PDP must send a Client-Accept message with a Client-Type of zero to the PEP carrying an Integrity object. This Integrity object will contain the initial sequence number that the PDP requires the PEP to increment during all subsequent communication with the PDP and the Key ID, identifying the key and algorithm used to compute the digest. The COPS protocol provides the Integrity object that can achieve authentication, message integrity, and replay prevention. Furthermore, it is good practice to use localised keys specific to a particular PEP such that a stolen PEP will not compromise the security of an entire administrative domain. The COPS Integrity object also provides sequence numbers to avoid replay attacks. The PDP chooses the initial sequence number for the PEP and the PEP chooses the initial sequence number for the PDP. These initial numbers are then incremented with each successive message sent over the connection in the corresponding direction. The initial sequence numbers should be chosen such that they are monotonically increasing and never repeat for a particular key. Security between the client (PEP) and server (PDP) may be provided by IP Security [IPSEC]. In this case, the IPSEC Authentication Header (AH) should be used for the validation of the connection. Additionally, IPSEC Encapsulation Security Payload (ESP) may be used to provide both validation and secrecy.

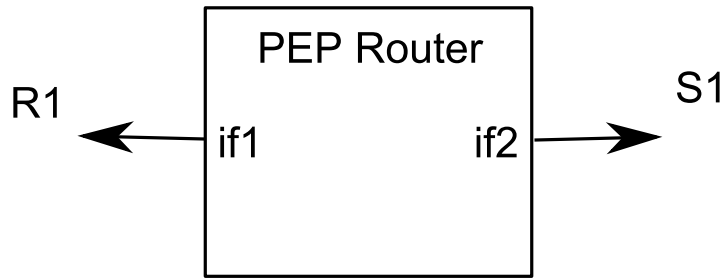
### 4.4.6 *Illustrative Examples, Using COPS for RSVP*

#### 4.4.6.1 Unicast Flow Example

This subsection details the steps in using COPS for controlling a Unicast RSVP flow. It details the contents of the COPS messages with respect to Figure 4.13.

The PEP router has two interfaces (if1, if2). Sender S1 sends to receiver R1. A Path message arrives from S1:

```
PEP --> PDP   REQ := <Handle A>
                    <Context: in & out, Path>
                    <In-Interface if2> <Out-Interface if1>
                    <ClientSI: all objects in Path message>
```



**Fig. 4.13** Unicast Example: a single PEP view.

```

PDP --> PEP   DEC := <Handle A> <Context: in & out, Path>
                    <Decision: Command, Install>
  
```

A Resv message arrives from R1:

```

PEP --> PDP   REQ := <Handle B>
                    <Context: in & allocation & out, Resv>
                    <In-Interface if1> <Out-Interface if2>
                    <ClientSI: all objects in Resv message>
  
```

```

PDP --> PEP   DEC := <Handle B>
                    <Context: in, Resv>
                    <Decision: command, Install>
                    <Context: allocation, Resv>
                    <Decision: command, Install>
                    <Decision: Stateless, Priority=7>
                    <Context: out, Resv>
                    <Decision: command, Install>
                    <Decision: replacement, POLICY-DATA1>
  
```

```

PEP --> PDP   RPT := <Handle B>
                    <Report type: commit>
  
```

Notice that the Decision was split because of the need to specify different decision objects for different context flags. Time passes, the PDP changes its decision:

```

PDP --> PEP   DEC := <Handle B>
                    <Context: allocation, Resv>
                    <Decision: command, Install>
                    <Decision: Stateless, Priority=3>
  
```

Because the priority is too low, the PEP preempts the flow:

```
PEP --> PDP    DRQ := <Handle B>
                    <Reason Code: Preempted>
```

Time Passes, the sender S1 ceases to send Path messages:

```
PEP --> PDP    DRQ := <Handle A>
                    <Reason: Timeout>
```

## 4.5 Conclusions

This chapter provided an overview on the signalling protocols and analysed in detail three of the most popular and promising signalling protocols/frameworks that are used in the current Internet: the Session Initiation Protocol (SIP), the Next Steps In Signalling (NSIS) framework, and the Common Open Policy Service (COPS) architecture and protocol.

For each signalling protocol/framework, background information was given and the main characteristics were described, presenting the overall architecture and overview of the protocol structure. The descriptions included architectural and protocol overviews, message formats, common operation and some examples.

## References

1. R. Steinmetz and K. Nahrstedt, *Multimedia Fundamentals: Media Coding and Content Processing*, 2nd ed. Prentice-Hall PTR, 2002, vol. 1.
2. —, *Multimedia Systems*, 1st ed. Berlin, Germany: Springer Verlag, 2004.
3. S. Floyd, "Connections with multiple congested gateways in packet-switched networks part 1: One-way traffic," *Computer Communications Review*, vol. 21, no. 5, pp. 30–47, October 1991.
4. M. Mathis, J. Semke, and J. Mahdavi, "The macroscopic behavior of the tcp congestion avoidance algorithm," *SIGCOMM Comput. Commun. Rev.*, vol. 27, no. 3, pp. 67–82, 1997.
5. ITU-T, "ITU-T recommendation G.114: One-way transmission time." 2003.
6. —, "ITU-T recommendation G.1010: End-user multimedia QoS categories." 2001.
7. T. Szigeti and C. Hattingh, *End-to-End QoS Network Design: Quality of Service in LANs, WANs, and VPNs (Networking Technology)*. Cisco Press, 2005.
8. S. Keshav, *An Engineering Approach to Computer Networking*. Addison Wesley, 1997.
9. R. Braden, D. Clark, and S. Shenker, "Integrated services in the internet architecture: an overview," RFC 1633, June 1994.
10. S. Shenker and J. Wroclawski, "General characterization parameters for integrated service network elements," RFC 2215, September 1997.
11. J. Wroclawski, "Specification of the Controlled-Load Network Element Service," RFC 2211 (Proposed Standard), September 1997. [Online]. Available: <http://www.ietf.org/rfc/rfc2211.txt>
12. S. Shenker, C. Partridge, and R. Guerin, "Specification of Guaranteed Quality of Service," RFC 2212 (Proposed Standard), September 1997. [Online]. Available: <http://www.ietf.org/rfc/rfc2212.txt>
13. J. Solomon, "Applicability Statement for IP Mobility Support," RFC 2005 (Proposed Standard), October 1996. [Online]. Available: <http://www.ietf.org/rfc/rfc2005.txt>
14. S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Service," RFC 2475 (Informational), December 1998, updated by RFC 3260. [Online]. Available: <http://www.ietf.org/rfc/rfc2475.txt>
15. K. Nichols, S. Blake, F. Baker, and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers," RFC 2474 (Proposed Standard), December 1998, updated by RFCs 3168, 3260. [Online]. Available: <http://www.ietf.org/rfc/rfc2474.txt>
16. K. Nichols and B. Carpenter, "Definition of Differentiated Services Per Domain Behaviors and Rules for their Specification," RFC 3086 (Informational), April 2001. [Online]. Available: <http://www.ietf.org/rfc/rfc3086.txt>
17. K. Nichols, V. Jacobson, and K. Poduri, "A per-domain behavior for circuit emulation in ip networks," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 2, pp. 71–83, 2004.
18. B. Davie, A. Charny, J. Bennet, K. Benson, J. L. Boudec, W. Courtney, S. Davari, V. Firoiu, and D. Stiliadis, "An Expedited Forwarding PHB (Per-Hop Behavior)," RFC 3246 (Proposed Standard), März 2002. [Online]. Available: <http://www.ietf.org/rfc/rfc3246.txt>
19. J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured Forwarding PHB Group," RFC 2597 (Proposed Standard), June 1999, updated by RFC 3260. [Online]. Available: <http://www.ietf.org/rfc/rfc2597.txt>
20. J. Babiarz, K. Chan, and F. Baker, "Configuration Guidelines for DiffServ Service Classes," RFC 4594 (Informational), August 2006. [Online]. Available: <http://www.ietf.org/rfc/rfc4594.txt>
21. L. Breslau, E. W. Knightly, S. Shenker, I. Stoica, and H. Zhang, "Endpoint admission control: architectural issues and performance," in *SIGCOMM '00: Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*. New York, NY, USA: ACM, 2000, pp. 57–69.
22. K. Ramakrishnan and S. Floyd, "A Proposal to add Explicit Congestion Notification (ECN) to IP," RFC 2481 (Experimental), Januar 1999, obsoleted by RFC 3168. [Online]. Available: <http://www.ietf.org/rfc/rfc2481.txt>

23. C. Cetinkaya, V. Kanodia, and E. W. Knightly, "Scalable services via egress admission control," *IEEE Transactions on Multimedia*, vol. 3, no. 1, pp. 69–81, 2001. [Online]. Available: [citeseer.ist.psu.edu/cetinkaya01scalable.html](http://citeseer.ist.psu.edu/cetinkaya01scalable.html)
24. S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," in *ACM SIGCOMM*, vol. 26, no. 4. New York: ACM Press, Aug. 1996, pp. 117–130.
25. V. Paxson, "Strategies for Sound Internet Measurement," in *4th Internet Measurements Conference*, 2004.
26. V. Paxson, G. Almes, J. Mahdavi, and M. Mathis, "Framework for IP Performance Metrics," RFC 2330, May 1998.
27. J. Mahdavi and V. Paxson, "IPPM Metrics for Measuring Connectivity," RFC 2678, Sept. 1999.
28. G. Almes, S. Kalidindi, and M. Zekauskas, "A One-way Delay Metric for IPPM," RFC 2679, Sept. 1999.
29. I.-T. R. Y.1541, "Network Performance Objectives for IP-based Services," Review Jan 2005.
30. G. Almes, S. Kalidindi, and M. Zekauskas, "A Round-trip Delay Metric for IPPM," RFC 2681, Sept. 1999.
31. —, "A One-way Packet Loss Metric for IPPM," RFC 2680, Sept. 1999.
32. C. Demichelis and P. Chimento, "IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)," RFC 3393, Nov. 2002.
33. M. Mathis and M. Allman, "A Framework for Defining Empirical Bulk Transfer Capacity Metrics," RFC 3148, July 2001.
34. I.-T. R. P.800.1, "Mean Opinion Score (MOS) terminology," 2003.
35. A. Takahashi, H. Yoshino, and N. Kitawaki, "Perceptual qos assessment technologies for voip," *IEEE/Comm. Mag.*, vol. 42, no. 7, pp. 28–34, 2004.
36. I.-T. R. G.107, "The E-model, a computational model for use in transmission planning," 2003.
37. I.-T. R. P.910, "Subjective video quality assessment methods for multimedia applications," 1999.
38. I.-T. R. P.862, "Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs," 2001.
39. D. L. Mills, "Network Time Protocol (Version 3) Specification, Implementation and Analysis," RFC 1305, Mar. 1992.
40. D. Veitch, S. Babu, and A. Pásztor, "Robust synchronization of software clocks across the internet," in *Internet Measurement Conference (IMC '04)*, 2004.
41. N. Duffield, "Sampling for passive internet measurement: A review," *Statistical Science*, vol. 19, no. 3, pp. 472–498, 2004.
42. J. Quittek, T. Zseby, B. Claise, and S. Zander, "Requirements for IP Flow Information Export (IPFIX)," RFC 3917, Oct. 2004.
43. M. Crovella and B. Krishnamurthy, *Internet Measurement. Infrastructure, Traffic, and Applications*. John Wiley & Sons, Ltd, 2006.
44. D. Harrington, R. Presuhn, and B. Wijnen, "An architecture for describing simple network management protocol (snmp) management frameworks," RFC 3411, Dec. 2002.
45. "Lobster - Large-scale Monitoring of Broadband Internet Infrastructures," <http://www.ist-lobster.org>.
46. N. G. Duffield and M. Grossglauser, "Trajectory sampling for direct traffic observation," *IEEE/ACM Trans. Netw.*, vol. 9, no. 3, pp. 280–292, 2001.
47. T. Zseby, S. Zander, and G. Carle, "Evaluation of Building Blocks for Passive One-Way-Delay Measurements," in *Passive and Active Measurements Conference*, 2001.
48. S. B. Moon, "Measurement and Analysis of End-to-end Delay and Loss in the Internet," Ph.D. dissertation, University of Massachusetts, 2000.
49. J. Strauss, D. Katabi, and F. Kaashoek, "A measurement study of available bandwidth estimation tools," in *Internet Measurement Conference (IMC)*, 2003.

50. L. Lao, C. Dovrolis, and M. Y. Sanadidi, "The probe gap model can underestimate the available bandwidth of multihop paths," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 5, pp. 29–34, 2006.
51. B. Melander, M. Bjorkman, and P. Gunningberg, "A New End-to-End Probing and Analysis Method for Estimating Bandwidth Bottlenecks," in *IEEE Globecom Global Internet Symposium*, 2000.
52. Q. Liu and J.-N. Hwang, "End-to-end available bandwidth estimation and time measurement adjustment for multimedia qos," in *ICME '03: Proceedings of the 2003 International Conference on Multimedia and Expo - Volume 3 (ICME '03)*. Washington, DC, USA: IEEE Computer Society, 2003, pp. 373–376.
53. A. Pásztor and D. Veitch, "Active Probing using Packet Quartets," in *Internet Measurement Workshop*, 2002.
54. R. Jain and S. A. Routhier, "Packet Trains-Measurements and a New Model for Computer Network Traffic," *IEEE Journal of Selected Areas in Communications*, vol. SAC-4, no. 6, pp. 986–995, 1986.
55. C. Dovrolis, P. Ramanathan, and D. Moore, "Packet-dispersion techniques and a capacity-estimation methodology," *IEEE/ACM Trans. Netw.*, vol. 12, no. 6, pp. 963–977, 2004.
56. "PlanetLab: An open platform for developing, deploying, and accessing planetary-scale services," <http://www.planet-lab.org/>.
57. "Cooperative Association for Internet Data Analysis (CAIDA)," <http://www.caida.org/>.
58. "Active Measurement Project (AMP)," <http://amp.nlanr.net/>.
59. F. Georgatos, F. Gruber, D. Karrenberg, M. Santcroos, A. sanj, H. Uijterwaal, and R. Wilhem, "Providing Active Measurements as a Regular Service for ISPs," in *Proceedings of Passive and Active Measurement*, 2001.
60. "Evergrow Traffic Observatory Measurement InfrastruCTure," <http://www.etomic.org/>.
61. "DIMES (Distributed Internet MEasurements & Simulations)," <http://www.netdimes.org/>.
62. "Ever-growing global scale-free networks, their provisioning, repair and unique functions," <http://www.evergrow.org/>.
63. S. Shalunov, B. Teitelbaum, A. Karp, J. W. Boote, and M. J. Zekauskas, "One-Way Active Measurements Protocol," RFC 4656, Sept. 2006.
64. S. McCanne and V. Jacobson, "The BSD Packet Filter: A New Architecture for User-level Packet Capture," in *USENIX Winter*, 1993.
65. "perfSONAR - PERFORMANCE Service-Oriented Network monitoring ARchitecture," <http://www.perfsonar.net>.
66. I. Miloucheva, P. Gutierrez, D. Hetzer, A. Nassri, and M. Beoni, "Intermon architecture for complex QoS analysis in inter-domain environment based on discovery of topology and traffic impact," in *Inter-domain Performance and Simulation Workshop, Budapest*, March 2004.
67. "Passive Measurement and Analysis (PMA)," <http://pma.nlanr.net/>.
68. D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, "Requirements for Traffic Engineering Over MPLS," RFC 2702 (Informational), Sept. 1999. [Online]. Available: <http://www.ietf.org/rfc/rfc2702.txt>
69. E. Rosen, D. Tappan, G. Fedorkow, Y. Rekhter, D. Farinacci, T. Li, and A. Conta, "MPLS Label Stack Encoding," RFC 3032 (Proposed Standard), Jan. 2001, updated by RFCs 3443, 4182. [Online]. Available: <http://www.ietf.org/rfc/rfc3032.txt>
70. L. Andersson, P. Doolan, N. Feldman, A. Fredette, and B. Thomas, "LDP Specification," RFC 3036 (Proposed Standard), Jan. 2001. [Online]. Available: <http://www.ietf.org/rfc/rfc3036.txt>
71. H. Smit and T. Li, "Intermediate System to Intermediate System (IS-IS) Extensions for Traffic Engineering (TE)," RFC 3784 (Informational), June 2004, updated by RFC 4205. [Online]. Available: <http://www.ietf.org/rfc/rfc3784.txt>
72. D. Katz, K. Kompella, and D. Yeung, "Traffic Engineering (TE) Extensions to OSPF Version 2," RFC 3630 (Proposed Standard), Sept. 2003, updated by RFC 4203. [Online]. Available: <http://www.ietf.org/rfc/rfc3630.txt>

73. R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification," RFC 2205 (Proposed Standard), Sept. 1997, updated by RFCs 2750, 3936, 4495. [Online]. Available: <http://www.ietf.org/rfc/rfc2205.txt>
74. D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels," RFC 3209 (Proposed Standard), Dec. 2001, updated by RFCs 3936, 4420, 4874. [Online]. Available: <http://www.ietf.org/rfc/rfc3209.txt>
75. B. Jamoussi, L. Andersson, R. Callon, R. Dantu, L. Wu, P. Doolan, T. Worster, N. Feldman, A. Fredette, M. Girish, E. Gray, J. Heinanen, T. Kilty, and A. Malis, "Constraint-Based LSP Setup using LDP," RFC 3212 (Proposed Standard), Jan. 2002, updated by RFC 3468. [Online]. Available: <http://www.ietf.org/rfc/rfc3212.txt>
76. D. Grossman, "New Terminology and Clarifications for Diffserv," RFC 3260 (Informational), Apr. 2002. [Online]. Available: <http://www.ietf.org/rfc/rfc3260.txt>
77. F. Le Faucheur, L. Wu, B. Davie, S. Davari, P. Vaananen, R. Krishnan, P. Cheval, and J. Heinanen, "Multi-Protocol Label Switching (MPLS) Support of Differentiated Services," RFC 3270 (Proposed Standard), May 2002. [Online]. Available: <http://www.ietf.org/rfc/rfc3270.txt>
78. F. L. Faucheur and W. Lai, "Requirements for Support of Differentiated Services-aware MPLS Traffic Engineering," RFC 3564 (Informational), July 2003. [Online]. Available: <http://www.ietf.org/rfc/rfc3564.txt>
79. F. Le Faucheur and W. Lai, "Maximum Allocation Bandwidth Constraints Model for Diffserv-aware MPLS Traffic Engineering," RFC 4125 (Experimental), June 2005. [Online]. Available: <http://www.ietf.org/rfc/rfc4125.txt>
80. F. Le Faucheur, "Russian Dolls Bandwidth Constraints Model for Diffserv-aware MPLS Traffic Engineering," RFC 4127 (Experimental), June 2005. [Online]. Available: <http://www.ietf.org/rfc/rfc4127.txt>
81. W. Lai, "Bandwidth Constraints Models for Differentiated Services (Diffserv)-aware MPLS Traffic Engineering: Performance Evaluation," RFC 4128 (Informational), June 2005. [Online]. Available: <http://www.ietf.org/rfc/rfc4128.txt>
82. F. Le Faucheur, "Protocol Extensions for Support of Diffserv-aware MPLS Traffic Engineering," RFC 4124 (Proposed Standard), June 2005. [Online]. Available: <http://www.ietf.org/rfc/rfc4124.txt>
83. R. Stewart, Q. Xie, K. Morneault, and H. Schwarzbauer, "Stream Control Transmission Protocol," IETF, Tech. Rep. 2960, 2000.
84. T. Dierks and C. Allen, "The TLS Protocol," IETF, Tech. Rep. 2246, 1999.
85. D. Katz, "Router Alert Option," IETF, Tech. Rep. 2113, 1997.
86. M. Stiernerling, H. Tschofenig, C. Aoun, and E. Davies, "Nat/firewall nsis signaling layer protocol (nslp)," IETF, internet draft, 2006.
87. D. Durham, J. Boyle, R. Cohen, S. Herzog, R. Rajan, and A. Sastry, "The COPS (Common Open Policy Service) Protocol," IETF, Tech. Rep. 2748, 2000.
88. isox214, "ITU-T recommendation X.214 (11/95) information technology," Nov. 1995.
89. J. Postel, "Transmission control protocol: DARPA internet program protocol specification," IETF, Request For Comments 793, 1981.
90. —, "User datagram protocol (UDP)," IETF, Request For Comments 768, Aug. 1980.
91. H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "An extension to the Selective Acknowledgment (SACK) Options for TCP," IETF, Task Force 3550, 2003.
92. V. Jacobson and R. Braden, "TCP extensions for long-delay paths," IETF, Request For Comments 1072, Oct. 1988.
93. V. Jacobson, R. Braden, and D. Borman, "Tcp extensions for high performance," IETF, Request For Comments 1323, 1992.
94. L. S. Brakno and L. L. Peterson, "TCP Vegas: End to End congestion Avoidance on a Global Internet," *IEEE Journal on Selected Areas in Communications*, no. 13, pp. 1465–1480, 1995.
95. S. Floyd, T. Henderson, and A. Gurtov, "The newreno modification to tcp's fast recovery algorithm," IETF, Request For Comments 3782, Apr. 2004.
96. S. Floyd, J. Mahdavi, M. Mathis, and M. Podolsky, "An extension to the selective acknowledgement (SACK) option for TCP," IETF, Request For Comments 2883, July 2000.

97. O. Ait-Hellal and E. Altman, "Analysis of tcp-vegas and tcp-reno," in *Proc. of the IEEE International Conference on Communications - ICC*, 1997, pp. 495–499.
98. V. Jacobson, L. Peterson, L. Brakmo, and S. Floyd, "Problems with arizona's vegas," mailing list, end2end-tf, Tech. Rep., 1994, <ftp://ftp.ee.lbl.gov/email/vanj.94mar14.txt>.
99. L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control for fast long-distance networks," in *Proc. of IEEE INFOCOM*, Mar. 2004, pp. 2514–2524.
100. S. Floyd, "Highspeed tcp for large congestion windows," IETF, Request For Comments 3649, Dec. 2003.
101. D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," in *Proc. of ACM SIGCOMM*, 2002.
102. Y. Xia, L. Subramanian, I. Stoica, and S. Kalyanaraman, "One more bit is enough," in *Proc. of ACM SIGCOMM*, 2005.
103. K. Ramakrishnan, S. Floyd, and D. Black, "The addition of explicit congestion notification (ECN) to ip," IETF, Request For Comments 3168, Sept. 2001.
104. P. Sinha, T. Nandagopal, N. Venkitaraman, R. Sivakumar, and V. Bharghavan, "WTCP: A reliable transport protocol for wireless wide-area networks," *Wireless Networks*, vol. 8, no. 2-3, pp. 301–316, 2002.
105. M. Gerla, M. Y. Sanadidi, R. Wang, A. Zanella, C. Casetti, and S. Mascolo, "Tcp westwood: congestion window control using bandwidth estimation," in *Proc. of IEEE GLOBECOM*, 2001, pp. 1698–1702.
106. C. P. Fu and S. C. Liew, "Tcp veno: Tcp enhancement for transmission over wireless access networks," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 2, pp. 216–229, Feb. 2003.
107. R. S. et Al., "Sockets API extensions for stream control transmission protocol (SCTP)," IETF, Internet Draft draft-ietf-tsvwg-sctpsocket-06.txt, Mar. 2003.
108. E. Kohler, M. Handley, and S. Floyd, "Datagram congestion control protocol (DCCP)," IETF, Request For Comments 4340, Mar. 2006.
109. S. Floyd and E. Kohler, "Profile for DCCP Congestion Control ID 2: TCP-like Congestion Control," IETF, Request For Comments 4341, Mar. 2006.
110. S. Floyd, E. Kohler, and J. Padhye, "Profile for DCCP congestion control ID 3: TRFC congestion control," IETF, Request For Comments 4342, Mar. 2006.
111. M. Handley, S. Floyd, J. Padhye, and J. Widmer, "Tcp-friendly rate control (TFRC): Protocol specification," IETF, Request For Comments 3448, Jan. 2003.
112. S. Iren, P. D. Amer, and P. T. Conrad, "The transport layer: tutorial and survey," *ACM Computer Survey*, vol. 31, no. 4, pp. 360–404, 1999.
113. S. Floyd, "Congestion control principles," IETF, Request For Comments 2914, Sept. 2000.
114. Y.-Q. Z. D. Wu, T. Hou, "Transporting real-time video over the internet: Challenges and approaches," in *Proc. of the IEEE*, vol. 88, no. 12, Dec. 2000, pp. 1855–1875.
115. V. Jacobson, "Congestion avoidance and control," in *Proc. of ACM SIGCOMM*, Stanford, CA, Aug. 1988, pp. 314–329.
116. R. Talluri, "Error-resilience video coding in the ISO MPEG-4 standard," *IEEE Communications Magazine*, pp. 112–119, June 1998.
117. Y. Wang and Q.-F. Zhu, "Error control and concealment for video communication: A review," in *Proc. of the IEEE*, vol. 86, no. 5, May 1998, pp. 974–997.
118. M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP selective acknowledgment options," IETF, Request For Comments 2018, Oct. 1996.
119. J. Widmer, "Equation-based congestion control," Diploma Thesis, University of Mannheim, Germany, Feb. 2000.
120. S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the internet," *IEEE/ACM Transactions on Networking*, vol. 7, no. 4, pp. 458–472, 1999.
121. P. Amer, C. Chassot, C. Connolly, P. Conrad, and M. Diaz, "Partial order transport service for MM and other application," *IEEE/ACM Transactions on Networking*, vol. 2, no. 5, 1994.
122. L. Rojas-Cardenas, E. Chaput, L. Dairaine, P. Snac, and M. Diaz, "Video transport over partial order connections," *Computer Networks*, vol. 31, no. 7, pp. 709–725, Apr. 1999.

123. J.-P. V. A. Farrel and J. Ash, "A Path Computation Element (PCE)-Based Architecture," The Internet Society, Request For Comments 4655, Aug. 2006.
124. J. Enríquez, M. A. Callejo, and et al., "EuQoS Architecture Deliverable D122," 2007.
125. A. Beben, "EQ-BGP: an efficient inter-domain QoS routing protocol," in *AINA '06: Proceedings of the 20th IEEE International Conference on Advanced Information Networking and Applications*. Los Alamitos, CA, United States: IEEE Computer Society, Apr. 2006, pp. 560–564.
126. X. Masip-Bruin, M. Yannuzzi, R. Serral-Gracia, J. Domingo-Pascual, J. Enriquez-Gabeiras, M. Callejo, M. Diaz, F. Racaru, G. Stea, E. Mingozzi, A. Beben, W. Burakowski, E. Monteiro, and L. Cordeiro, "The EuQoS System: A solution for QoS Routing in Heterogeneous Networks," *IEEE Communications Magazine*, vol. 45, no. 2, Feb. 2007.
127. Y. Rekhter, T. Li, and S. Hares, "A Border Gateway Protocol 4 (BGP-4)," The Internet Society, Request For Comments 4271, Jan. 2006.
128. L. Baresse and et al., "Integrated EuQoS system Software architecture for application use cases and API for application driving Deliverable D421."
129. P. Krawiec and W. Burakowski, "Resource allocation strategies for new connections in qos multi-domain networks with signaling capabilities," in *Proceedings of Australian Telecommunication Networks and Applications Conference 2007 (ATNAC 2007)*, Christchurch, New Zealand, Dec. 2007, pp. 337–342.
130. A. Beben and et al., "Definition of measurement and monitoring system for Phase 2 Deliverable D222."
131. A. Bak, W. Burakowski, F. Ricciato, S. Salsano, and H. Tarasiuk, "A framework for providing differentiated QoS guarantees in IP-based network," *Computer Communications*, vol. 26, no. 4, pp. 327–337, 2003.
132. C. Brandauer, W. Burakowski, M. Dabrowski, B. Koch, and H. Tarasiu, "AC algorithms in AQUILA QoS IP network," *European Transactions on Telecommunications*, vol. 16, no. 3, pp. 225–232, 2005.
133. M. Dabrowski, G. Eichler, M. Fudala, D. Katzengruber, T. Kilkanen, N. Miettinen, H. Tarasiuk, and M. Titze, "Evaluation of the AQUILA Architecture: Trial Results for Signalling Performance, Network Services and User Acceptance," in *Art-QoS*, 2003, pp. 218–233.
134. K. Chan, J. Babiarz, and F. Baker, "Aggregation of DiffServ Service Classes," Transport Area Working Group," Internet-Draft, Nov. 2007.
135. "Nexuiz project," [www.alientrap.org/nexuiz/](http://www.alientrap.org/nexuiz/).
136. J. M. Batalla and R. Janowski, "Provisioning dedicated class of service for reliable transfer of signaling traffic," in *International Teletraffic Congress*, ser. Lecture Notes in Computer Science, L. Mason, T. Drwiega, and J. Yan, Eds., vol. 4516. Springer, 2007, pp. 853–864.
137. "Medigraf," <http://www.medigraf.pt/>.
138. J. W. Roberts, U. Mocci, and J. T. Virtamo, Eds., *Broadband Network Teletraffic - Performance Evaluation and Design of Broadband Multiservice Networks: Final Report of Action COST 242*, ser. Lecture Notes in Computer Science. Springer, 1996, vol. 1155.
139. L. Kleinrock, *Queueing Systems, Volume 2, Computer Applications*. John Wiley & Sons Inc, 1976.
140. H. Tarasiuk, R. Janowski, and W. Burakowski, "Admissible traffic load of real time class of service for inter-domain peers," in *Proceedings of Joint International Conference on Autonomous and Autonomous Systems and International Conference on Networking and Services (ICAS/ICNS 2005)*. Papeete, Tahiti, French Polynesia: IEEE Computer Society, Oct. 2005.
141. —, "Application of Admission Control and Traffic Shaping for providing TCP Throughput Guarantees," in *Proceedings of International Workshop To-QoS'2006 in conjunction with IFIP 2006 Networking Conference (ed. W. Burakowski)*, Coimbra, Portugal, May 2006, pp. 163–172.
142. "IEEE standards for local and metropolitan area networks. Virtual bridged local area networks," *IEEE Std 802.1Q, 2003 Edition (Incorporates IEEE Std 802.1Q-1998, IEEE Std 802.1u-2001, IEEE Std 802.1v-2001, and IEEE Std 802.1s-2002)*, 2003.
143. "IEEE Standard for Local and metropolitan area networks Media Access Control (MAC) Bridges," *IEEE Std 802.1D-2004 (Revision of IEEE Std 802.1D-1998)*, 2004.

144. M. Carmo, J. S. Silva, E. Monteiro, P. Simões, and F. Boavida, "Ethernet QoS Modeling in Emerging Scenarios," in *Proceedings of 3rd International Workshop on Internet Performance, Simulation, Monitoring and Measurement (IPS-MoMe 2005)*. Warsaw, Poland: IST MoMe Cluster, Mar. 2005, pp. 90–96.
145. M. Carmo, B. Carvalho, J. S. Silva, E. Monteiro, P. Simões, M. Curado, and F. Boavida, "NSIS-based Quality of Service and Resource Allocation in Ethernet Networks," in *Proceedings of 4th International Conference on Wired/Wireless Internet Communications 2006*, 2006.
146. M. Carmo, J. S. Silva, and E. Monteiro, "EuQoS approach for Resource Allocation in Ethernet Networks," *International Journal of Network Management*, vol. 17, no. 5, pp. 373–388, sep / oct 2007.
147. "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements," *IEEE Std 802.11e-2005*, 2005.
148. E. Lochin and L. D. A. Jourjon, "gtfrc, a tcp friendly qos-aware rate control for diffserv assured service," *Springer Telecommunication Systems*, vol. 10.1007/s11235-006-9004-2, ISSN: 1018-4864 (Print) 1572-9451 (Online), September 2006 2006.
149. E. Exposito, P. Sénac, and M. Diaz, "Compositional architecture pattern for qos-oriented communication mechanisms," in *MMM*, Y.-P. P. Chen, Ed. IEEE Computer Society, 2005, pp. 413–420.
150. B. Quinn and K. Almeroth, "Ip multicast applications: Challenges and solutions," The Internet Society, Request For Comments 3170, Sept. 2001.
151. Z. Albanna, K. Almeroth, D. Meyer, and M. Schipper, "Iana guidelines for ipv4 multicast address assignments," The Internet Society, Request For Comments 3171, Aug. 2001.
152. R. Hinden and S. Deering, "Internet protocol version 6 (ipv6) addressing architecture," The Internet Society, Request For Comments 3513, Apr. 2003.
153. B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan, "Internet group management protocol, version 3," The Internet Society, Request For Comments 3376, Oct. 2002.
154. A. Adams, J. Nicholas, and W. Siadak, "Protocol independent multicast - dense mode (pim-dm): Protocol specification (revised)," The Internet Society, Request For Comments 3973, Jan. 2005.
155. B. Fenner, M. Handley, H. Holbrook, and I. Kouvelas, "Protocol independent multicast - sparse mode (pim-sm): Protocol specification (revised)," The Internet Society, Request For Comments 4601, Aug. 2006.
156. D. Waitzman, C. Partridge, and S. Deering, "Distance vector multicast routing protocol," Network Working Group, Request For Comments 1075, Nov. 1988.
157. J. Moy, "Multicast extensions to ospf," Network Working Group, Request For Comments 1584, Mar. 1994.
158. K. Savetz, N. Randall, and Y. Lepage, *MBONE: Multicasting Tomorrow's Internet*. John Wiley & Sons Inc, 1996.
159. R. Zhang and Y. C. Hu, "Borg: A hybrid protocol for scalable application-level multicast in peer-to-peer networks," in *NOSSDAV '03: Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video*, ACM. New York, NY, USA: ACM Press, June 2003, pp. 172–179.
160. A. Sobeih, W. Yurcik, and J. C. Hou, "Vring: A case for building application-layer multicast rings (rather than trees)," in *MASCOTS '04: Proceedings of the The IEEE Computer Society's 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'04)*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 437–446.
161. S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. H. Katz, and J. D. Kubiawicz, "Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination," in *NOSSDAV '01: Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video*. Port Jefferson, New York, United States: ACM Press, 2001, pp. 11–20.

162. M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: High-bandwidth multicast in a cooperative environment," in *19th ACM Symposium on Operating Systems Principles (SOSP'03)*, Bolton Landing, New York, USA, oct 2003.
163. A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," in *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, Heidelberg, Germany, nov 2001, pp. 329–350.
164. M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, "Scribe: A large-scale and decentralized application-level multicast infrastructure," *IEEE Journal on Selected Areas in Communication (JSAC)*, vol. 20, no. 8, oct 2002.
165. M. Brogle, D. Milic, and T. Braun, "Qos enabled multicast for structured p2p networks," in *4th IEEE Consumer Communications and Networking Conference*, Las Vegas, NV, USA, Jan. 2007.
166. D. Milic, M. Brogle, and T. Braun, "Video broadcasting using overlay multicast," in *Seventh IEEE International Symposium on Multimedia (ISM 2005)*, Irvine, CA, USA, Dec. 2005, pp. 515–522.
167. A. Sulistio, C. S. Yeo, and R. Buyya, "A taxonomy of computer-based simulations and its mapping to parallel and distributed systems simulation tools," *Softw. Pract. Exper.*, vol. 34, no. 7, pp. 653–673, 2004.
168. N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, pp. 1087–1092, 1953.
169. T. Yung, J. Martin, M. Takai, and R. Bagrodia, "Integration of fluidbased analytical model with packet-level simulation for analysis of computer networks," 2001.
170. C. Kiddle, R. Simmonds, C. Williamson, and B. Unger, "Hybrid packet/fluid flow network simulation," in *PADS '03: Proceedings of the seventeenth workshop on Parallel and distributed simulation*. Washington, DC, USA: IEEE Computer Society, 2003, p. 143.
171. J. Incera, R. Marie, D. Ros, and G. Rubino, "Fluidsim: a tool to simulate fluid models of high-speed networks," *Perform. Eval.*, vol. 44, no. 1-4, pp. 25–49, 2001.
172. B. Liu, D. R. Figueiredo, Y. Guo, J. F. Kurose, and D. F. Towsley, "A study of networks simulation efficiency: Fluid simulation vs. packet-level simulation," in *INFOCOM*, 2001, pp. 1244–1253.
173. "Glomosim," <http://pcl.cs.ucla.edu/projects/glomosim/>.
174. "Qualnet," <http://www.scalable-networks.com/>.
175. "Swans," <http://jist.ece.cornell.edu/>.
176. "Ssfnet," <http://www.ssfnet.org/homePage.html>.
177. "Omnet++," <http://www.omnetpp.org/>.
178. "ns-2," <http://www.isi.edu/nsnam/ns/index.html>.
179. "Vint - virtual internet network testbed project," <http://www.isi.edu/nsnam/vint/index.html>.
180. K. Fall and K. Varadhan, "The ns manual (formerly ns notes and documentation)," 2002.
181. F. J. Ros and P. M. Ruiz, "Implementing a new manet unicast routing protocol in ns2," Dept. of Information and Communications Engineering University of Murcia, Tech. Rep., 2004.
182. "Vmware software (virtual machine)," <http://www.vmware.com>, 2005.
183. V. J. Easton and J. H. McColl, "Statistics glossary v1.1," <http://www.cas.lancs.ac.uk/glossaryv1.1/main.html>, vol. 2005, 2005.
184. M. Allman, A. Caldwell, and S. Ostermann, "One: The ohio network emulator," *TR-19972*, 1997.
185. L. Rizzo, "Dumynet: a simple approach to the evaluation of protocols," *ACM Computer Communication Review*, 1997.
186. M. Carson and D. Santay, "Nist net: A linux-based network emulation tool," *ACM Computer Communication Review*, 2003.
187. S. Hemminger, "Netem, network emulator," <http://developer.osdl.org/shemminger/netem/>, 2005.
188. P. Zheng and L. M. Nil, "Empower: A network emulator for wireline and wireless networks," in *IEEE Infocom*, San Francisco, California, USA, 2003.

189. R. Buyya, D. Abramson, and J. Giddy, "A case for economy grid architecture for service oriented grid computing," in *10th IEEE International Heterogeneous Computing Workshop*, San Francisco, California, USA, 2001.
190. B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, "An integrated experimental environment for distributed systems and networks," in *5th symposium on Operating systems design and implementation*, Boston, USA, 2002.
191. CNRS, "Grid 5000," <http://www.grid5000.org>, 2003.
192. L. Peterson, T. Anderson, D. Culler, and T. Roscoe, "A blueprint for introducing disruptive technology into the internet," in *First Workshop on Hot Topics in Networking (HotNets-1)*, Princeton, New Jersey, USA, 2002.
193. D. Herrscher and K. Rothermel, "A dynamic network scenario emulation tool," in *11th International Conference on computer communications and networks*, Florida, USA, 2002.
194. "Network node emulation and method of node emulation," *European patent*, 2002.
195. S. Dawson and F. Jahanian, "Probing and fault injection of distributed protocols implementations," in *International Conference on Distributed Computer Systems*, 1995.
196. M. Zec and M. Mikuc, "Operating system support for integrated network emulation in imunes," in *First Workshop on Operating System and Architectural Support for the on demand IT InfraStructure*, Boston, USA, 2004.
197. X. W. Huang, R. Sharma, and S. Keshav, "The entrapid protocol development environment," in *IEEE Infocom*, Boston, USA, 1999.
198. E. Savage and D. Wetherell, "Alpine, a user level infrastructure for network development environment," in *IEEE Infocom*, Boston, USA, 1999.
199. F. Baumgartner, T. Braun, and B. Bhargava, "Virtual routers: a tool for emulating ip routers," in *27th Annual IEEE Conference on Local Computer Networks*, 2002.
200. B. Noble, M. Satyanarayanan, D. Narayanan, J. E. Tilton, J. Flinn, and K. R. Walker, "Trace-based mobile network emulation," in *ACM SIGCOMM*, Cannes, France, 1997.
201. NS-Group, "The network simulator - ns-2," <http://www.isi.edu/nsnam/ns>, 1989.
202. K. Fall, "Network emulation in the vint/ns simulator," in *Fourth IEEE Symposium on Computers and Communications*, 1999.
203. C. Kiddle, R. Simmonds, and B. Unger, "Improving scalability of network emulation through parallelism and abstraction," in *38th Annual Simulation Symposium (ANSS'05)*, 2005.
204. M. Gineste, H. Thalmensy, L. Dairaine, P. Senac, and M. Diaz, "Active emulation of a dvb-rcs satellite link in an end-to-end qos-oriented heterogeneous network," in *23rd AIAA International Communication Satellite Systems (ICSSC)*, Rome, Italy, 2005.
205. ETSI, "Dvb-rcs: Digital video broadcasting (dvb); interaction channel for satellite distribution systems," vol. EN 301 790, 2003.
206. —, "Digital video broadcasting (dvb); dvb specification for data broadcasting," vol. EN 301 192, 1999.
207. ISO/IEC, "Information technology – generic coding of moving pictures and associated audio: Systems," vol. 13818-1, 1994.
208. —, "Information technology - generic coding of moving pictures and associated audio information - part 6: Extensions for dsm-cc," vol. 13818-6, 1998.
209. S.Kota and M. Marchese, "Quality of service for satellite ip networks: a survey," *International Journal of Satellite Communications and Networking*, vol. 21, pp. 303–349, 2003.
210. ETSI, "Satellite earth stations and systems (ses); broadband satellite multimedia (bsm) services and architectures: Qos functional architecture," vol. TS 102 462, 2006.
211. S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," vol. RFC 2475, 1998.



# Index

- active emulation, 236, 249
- Admission Control, 11
- Aggregation, 40
- Application-aware transport mechanisms, 130
- Application layer multicast, 172
- Application Quality Service Signalling Negotiation, 141
- AQ-SSN, 141
- Assured Forwarding, 72
- Audio Application, 7
- Automatic Repeat Request, 127
  
- Bandwidth, 4
- Bandwidth Constraint Model, 76
- Behavior Aggregates, 71
- Best-effort, 1
  
- CAC, 144, 162
- CAT, 112
- CC, 112
- Class-Type, 75
- Class of Service, 16
- Codec, 5
- Compression, 20
- Computer simulation, 187
- congestion control, 124, 126
- Connection Admission Control, 144
- Conservation Law, 9
- Constrained Shortest Path First, 64
- Constraint-Based Routing, 62
- Control Component, 59
- Controlled Load Service, 13
- COPS, 79, 104
  - Protocol, 106
- CoS, 138, 144
- CoSs in LAN/Ethernet, 163
- CoSs in MPLS, 166
  
- CoSs in satellite, 167
- CoSs in UMTS, 165
- CoSs in WiFi, 164
- CoSs in xDSL, 163
  
- Data analysis, 37
- Data collection, 37
- Datagram Congestion Control Protocol, 123
- Data storage, 37
- Data Traffic, 8
- DEC, 110
- Delay Variation, 2, 3, 7
- Differentiated Services, 15
  - Codepoint, 15
- DiffServ, 15, 71
- DiffServ-aware Traffic Engineering, 75
- drift, 37
- Dropping, 18
- DRQ, 111
- DSCP, 15
- Dummynet, 229, 238
- DVB-RCS, 239
- DVB-S, 239
  
- E-LSP, 72
- e2e CoSs, 157
- Egress Admission Control, 18
- Elastic Application, 5
- emulation models, 231
- End-to-end Classes of Service, 157
- End-to-end QoS, 19
- End Point Admission Control, 17
- Endpoint Admission Control, 18
- Enhanced Transport Protocol Mechanisms, 129
- EQ-BGP, 147, 149
- EQ-ETP, 168

- EQ-SAP, 143, 152, 154
- EQ-Service Access Point, 152
- EQ link, 147
- EQ path, 141
- EQ paths, 145
- error control mechanisms, 124
- ETP, 168
- EuQoS, 137
  - Application Quality Service Signalling
    - Negotiation, 141
  - AQ-SSN, 141
  - CAC, 144, 162
  - Connection Admission Control, 144
  - CoSs in LAN/Ethernet, 163
  - CoSs in MPLS, 166
  - CoSs in satellite, 167
  - CoSs in UMTS, 165
  - CoSs in WiFi, 164
  - CoSs in xDSL, 163
  - e2e CoSs, 157
  - End-to-end Classes of Service, 157
  - EQ-BGP, 147, 149
  - EQ-ETP, 168
  - EQ-SAP, 152
  - EQ-Service Access Point, 152
  - EQ link, 147
  - EQ paths, 145
  - ETP, 168
  - EuQoS-awareness, 182
  - EuQoS Enhanced QoS-oriented Transport Protocol, 168
  - EuQoS Enhanced Transport Protocol, 168
  - hard model, 147
  - inter-domain CAC, 162
  - invocation process, 151
  - loose model, 146
  - Medigraf, 180
  - Network Technology Dependent level, 139, 141
  - Network Technology Independent level, 139, 141
  - NTD, 139, 141
  - NTI, 139, 141
  - OAM, 146, 155
  - Operation, Administration and Maintenance, 146, 155
  - Path Computation Element, 144
  - PCE, 144
  - provisioning process, 146
  - QCM, 140
  - QoS on-demand service, 139, 152
  - Quality Control Module, 140
  - RA, 144
  - Resource Allocator, 144
  - Resource Manager, 143
  - RM, 143
  - Telemedicine applications, 180
- EuQoS-awareness, 182
- EuQoS Enhanced QoS-oriented Transport Protocol, 168
- EXP-Inferred-PSC LSP, 72
- Expedited Forwarding, 16
- experimentation, 220
- experiments, 221
  - applications, 223
  - protocols, 223
- Explicit Route Object, 67
- Explicit Route Signalling, 66
- Fairness, 10
- Fast Rerouting, 69
- FCFS, 9
- Field
  - Header, 85
- FIFO, 9
- firewall, 82
- First Come First Serve, 9
- First In First Out, 9
- Flow, 26
- Flow control, 125, 128
- Forking Proxy, 83
- Forwarding Component, 57
- Forwarding Equivalence Classes, 56
- Generalised Processor Sharing, 11
- GIST, 95, 96, 98
- GloMoSim, 193
- Go-Back-N, 128
- GPS, 11
- Guaranteed Service, 14
- hard model, 147
- Home Gateway, 142
- HTTP, 80
- IANA, 110
- IETF, 91
- IGMP, 171
- Inelastic Application, 5
- Integrated Services, 13, 66
- inter-domain CAC, 162
- Interactive Application, 6
- Internet group management protocol, 171
- IntServ, 13
- invocation process, 151
- IPSEC, 100, 104
- IPv4, 108

- KA, 112
- L-LSP, 72
- Label-Only-Inferred-PSC LSP, 72
- Label Distribution Protocol, 59, 66
- Label Forwarding Information Base, 58
- Label Stacking, 61
- Label Switched Path, 58
- Label Switching Routers, 56
- loose model, 146
- LPDP, 105
- Marking, 18
- MAS, 99
- Max-Min Fair Share, 10
- Maximum Allocation Model, 76
- Medigraf, 138, 180
- Metric, 26
  - $P_{BLOCK}$ , 31
  - $T_{release}$ , 32
  - $T_{set-up}$ , 31
  - BTC, 30
  - Bulk Transport Capacity, 30
  - Call Blocking, 31
  - Call Level, 30
  - Call release latency, 32
  - Call set-up latency, 31
  - Connectivity, 27
  - IP Delay Variation, 29
  - IPDV, 29
  - Jitter, 29
  - MOS, 33
  - Network Level, 26
  - One-Way Delay, 27
  - One-Way Packet Loss, 29
  - OWD, 27
  - OWPL, 29
  - Perceived QoS, 32
  - Round-Trip Delay, 28
  - RTD, 28
  - Subjective, 26
  - Type-P Packet, 27
  - User Level, 32
  - Wire-time, 27
- MIME, 86
- MPLS label, 57
- MRS, 99
- Multi-Protocol Label Switching, 56
- Multicast, 171
  - addressing, 171
  - protocols, 171
  - routing, 171
- Multicast Middleware, 175
- NAT, 95
- NAT and Firewall, 102
- Netem, 229
- Netflow, 40
- network emulation, 221
- Network Layer Reachability Information, 150
- Network Simulation, 189, 191
- Network Simulator ns-2, 195
- Network Simulators, 193
  - GloMoSim, 193
  - ns-2, *see* ns-2
  - OMNEST, 194
  - OMNeT++, 194
  - Qualnet, 193
  - SSFNet, 194
  - SWANS, 194
- Network Technology Dependent level, 139, 141
- Network Technology Independent level, 139, 141
- NLRI, 150
- Non-interactive Applications, 6
- Non-transparent Scaling, 20
- ns-2, 195
  - Analysing Methods, 218
  - Language Concept, 195
  - Links, 197
  - MobileNode, 200
  - Nodes, 197
  - Routing Protocol Implementation, 202
  - Structure, 196
  - Traffic, 197
  - Wireless Networks, 200
- nse, 235
- NSIS, 79, 91, 116, 143
- NSLP, 95, 96
- NTD, 139, 141
- NTI, 139, 141
- NTLP, 96
- NTP, 36
- OAM, 146, 155
- OMNEST, 194
- OMNeT++, 194
- One Way Delay, 2
- Open System Interconnection, 118
- Operation, Administration and Maintenance, 146, 155
- OPN, 111
- Ordered Aggregates, 72
- P2P, 138
- Packet Loss Rate, 4
- Packet Scheduling, 9

- Pastry, 173
- Path Computation Element, 144
- PCE, 143, 144
- PDB, 16
- PDP, 99, 104, 113
- Penultimate Hop Popping, 60
- PEP, 99, 104
- Per-domain Behaviour, 16
- Per Hop Behaviour, 15
- PHB, 15
- PHB Scheduling Class, 72
- Play-out Buffer, 7
- Play-out Buffering, 3
- PMPLS, 144
- Priority Scheduler, 11
- Probing Traffic, 17
- provisioning process, 143, 146
  
- QCM, 140
- QNF, 100
- QNI, 100
- QNR, 100
- QoS, 1, 79
  - Architectures, 13
  - End-to-end, 19
  - Parameters, 1
  - Requirements, 5
- QoS on-demand service, 139, 152
- QoS path, 141
- Quality Control Module, 140
- Quality of Service, 1
- Qualnet, 193
  
- RA, 144
- Random Dropping, 12
- Random Early Detection, 12
- RA Resource Allocator, 143
- rate-based congestion control, 126
- Rate controlled Scheduling, 12
- REA, 103
- Receiver-Driven Layered Multicast, 19
- RED, 12
- Registrar, 81
- Reliability mechanisms, 127
- Reservation Protocol, 66
- Resource Allocator, 144
- Resource Management
  - Function, 102
  - Interactions, 102
- Resource Manager, 143
- Resource Reservation Setup Protocol, 14
- RLM, 19
- RM, 143
- RMF, 100
- RM Ressource Manager, 143
- Round Robin, 11
- RPT, 111
- RSVP, 14, 91, 101
- RSVP-TE, 66
- Russian Doll Model, 76
  
- Sampling, 41
- satellite emulation, 236
- Scheduling, 19
- Scheduling Disciplines, 9
- Scribe, 174
- SCTP, 99
- SDP, 87
- SE, 92
- Selective Acknowledgement, 128
- Selective Repeat, 128
- Server
  - Proxy, 81, 82
  - Redirect Server, 81
  - Registrar, 82
- Service Class, 16
- Service Classes, 16
- Service Level Agreement, 15
- Service Level Specification, 15
- Shortest Path First, 64
- Shortest Path Tree, 64
- Signalling, 19, 79
  - Application Properties, 95
  - Centralised Architecture, 94
  - Entity, 92
  - QoS, 99
- Simulatin
  - Execution, 190
- Simulation
  - Continuous, 188
  - Discrete, 188
  - Discrete-event, 190
  - Distributed, 190
  - Experimentation and analysis, 188
  - Fluid Model, 190
  - Hybrid, 189
  - Hybrid Model, 191
  - Modelling, 188
  - Monte Carlo, 188
  - Network, 189, 191
  - Packet-Level, 190
  - Parallel, 190
  - Speedup, 191
  - Types, 188
  - Verification, 188
- simulation, 220
- SIP, 79, 80
  - Extension, 91

- Methods, 86
- Session, 88
- skew, 36
- SLA, 15
- SLS, 15
- SMS, 94
- SMTP, 80
- SNMP, 40
- SSC, 112
- SSFNet, 194
- SSQ, 111
- Start Line, 84
- Statistical Multiplexing, 18
- Stop-and-Wait, 127
- Stream Control Transmission Protocol, 122
- SWANS, 194
- Synchronisation, 36
  
- TAP device, 176
- TCP, 80, 104
- Telemedicine applications, 180
- Traffic
  - Data, 8
  - Strongly Regular, 8
  - Weakly Regular, 8
- Traffic Engineering, 53
- Traffic Engineering Database, 63
- traffic shapers, 228
- Transmission Control Protocol, 119
- Transparent Scaling, 19
- transport layer, 118
- transport protocols, 118
  
- UAC, 81
- UAS, 81, 82
- UDP, 80
- Unicast, 170
- URI, 81, 88
- User Datagram Protocol, 119
  
- Video Application, 7
- Virtual Wire, 16
  
- Weighted Fair Queuing, 11
- Weighted Round Robin, 11
- windows-based congestion control, 126
- Work-Conserving, 9