

Advanced Internet Protocols, Services, and Applications

Eiji Oki • Roberto Rojas-Cessa
Mallikarjun Tatipamula • Christian Vogt

Contents

[Cover](#)

[Title Page](#)

[Copyright](#)

[Preface](#)

[Audience](#)

[Organization](#)

[Acknowledgments](#)

[About the Authors](#)

[Chapter 1: Transmission Control Protocol/Internet Protocol Overview](#)

[1.1 Fundamental Architecture](#)

[1.2 Internet Protocol Basics](#)

[1.2.1 Packet Header](#)

[1.2.2 Internet Protocol Address](#)

[1.2.3 Internet Protocol Classification](#)

[1.2.4 Subnet and its Masking](#)

[1.2.5 Subnet Calculation](#)

[1.3 Routing](#)

[1.3.1 Routing Across Providers](#)

[1.3.2 Routing within Edge Networks](#)

[1.3.3 Routing Scalability](#)

[References](#)

Chapter 2: Transport-Layer Protocols

2.1 Transmission Control Protocol

2.1.1 Transmission Control Protocol Header Structure

2.1.2 Three-Way Handshake

2.1.3 Transmission Control Protocol Flow Control and Congestion Control

2.1.4 Port Number

2.2 User Datagram Protocol

2.2.1 User Datagram Protocol Header Structure

2.3 Stream Control Transmission Protocol

2.3.1 Stream Control Transmission Protocol Packet Structure

2.3.2 Security: Prevention of SYN Attacks

2.4 Real-Time Transport Protocol

2.4.1 Real-Time Transport Protocol Header Structure

References

Chapter 3: Internet Architecture

3.1 Internet Exchange Point

3.2 History of Internet Exchange Points

3.3 Internet Service Provider Interconnection Relationships

3.4 Peering and Transit

References

Chapter 4: IP Routing Protocols

4.1 Overview of Routing Protocols

4.1.1 Interior Gateway Protocol

4.1.2 Exterior Gateway Protocol

4.2 Routing Information Protocol

4.2.1 Routing Information Protocol Header Format

4.2.2 Update of Routing Table in Routing Information Protocol

4.2.3 Maintenance of Routing Table in Routing Information Protocol

4.2.4 Split Horizon

4.2.5 Limitations of Routing Information Protocol

4.3 Open Shortest Path First

4.3.1 Shortest-Path Algorithm

4.3.2 Hierarchical Routing

4.3.3 Open Shortest Path First Packet Format

4.3.4 Comparison of Routing Information Protocol and Open Shortest Path First

4.4 Border Gateway Protocol

4.4.1 Border Gateway Protocol Message Flows

4.4.2 Border Gateway Protocol Policy Selection Attributes

References

Chapter 5: Multiprotocol Label Switching

5.1 Overview

5.2 Functions and Mechanisms

5.3 Applicabilities

References

Chapter 6: IP Quality Of Service

6.1 Introduction

6.2 Quality of Service in IP Version 4

6.3 Integrated Services

6.3.1 Packet Scheduler

6.3.2 Packet Classifier

6.3.3 Admission Control

6.3.4 Resource Reservation Protocol (RSVP)

6.4 Differentiated Services

6.5 Quality of Service with Nested Differentiated Services Levels

6.5.1 Drawbacks of Explicit Endpoint Admission Control with Path Selection

6.5.2 OSPF-Based Adaptive and Flexible Quality of Service Provisioning

[6.5.3 Combination of Security and Quality of Service](#)

[6.5.4 Path Selection Algorithm Analysis](#)

[References](#)

[Chapter 7: IP Multicast and Anycast](#)

[7.1 Addressing](#)

[7.1.1 Multicast Addressing](#)

[7.1.2 Differences between Multicasting and Multiple Unicasting](#)

[7.2 Multicast Routing](#)

[7.2.1 Optimal Routing: Shortest-Path Trees](#)

[7.2.2 Unicast Routing](#)

[7.2.3 Multicast Routing](#)

[7.3 Routing Protocols](#)

[7.3.1 Multicast Open Shortest Path First \(MOSPF\)](#)

[7.3.2 Distance Vector Multicast Routing Protocol](#)

[7.3.3 Core-Based Tree \(CBT\) Protocol](#)

[7.3.4 Protocol-Independent Multicast](#)

[7.3.5 Simple Multicast Routing Protocol](#)

[7.4 Anycasting](#)

[7.4.1 Architectural Issues](#)

[7.4.2 Anycast Addresses](#)

[7.4.3 Differences between the Services Offered by IP Multicasting and IP Anycasting](#)

[7.5 IPv6 Anycast Routing Protocol: Protocol-Independent Anycast—Sparse Mode](#)

[References](#)

[Chapter 8: Layer-2 Transport over Packet](#)

[8.1 Draft-Martini Signaling and Encapsulation](#)

[8.1.1 Functionality](#)

[8.1.2 Encapsulation](#)

[8.1.3 Protocol-Specific Encapsulation](#)

[8.2 Layer-2 Tunneling Protocol](#)

[8.2.1 Layer-2 Tunneling Protocol Version 3](#)

[8.2.2 Pseudowire Emulation Edge to Edge](#)

[References](#)

Chapter 9: Virtual Private Wired Service

[9.1 Types of Private Wire Services](#)

[9.1.1 Layer-2 Virtual Private Services: Wide Area Networks and Local Area Networks](#)

[9.1.2 Virtual Private Wire Service](#)

[9.1.3 Virtual Private Multicast Service](#)

[9.1.4 IP-Only Layer-2 Virtual Private Network](#)

[9.1.5 Internet Protocol Security](#)

[9.2 Generic Routing Encapsulation](#)

[9.3 Layer-2 Tunneling Protocol](#)

[9.4 Layer-3 Virtual Private Network 2547bis, Virtual Router](#)

[9.4.1 Virtual Router Redundancy Protocol](#)

[References](#)

Chapter 10: IP and Optical Networking

[10.1 IP/Optical Network Evolution](#)

[10.1.1 Where Networking is Today](#)

[10.1.2 Where Networking is Going](#)

[10.2 Challenges in Legacy Traditional IP/Optical Networks](#)

[10.2.1 Proprietary Network Management Systems](#)

[10.2.2 Complexity of Provisioning in Legacy IP/Optical Networks](#)

[10.3 Automated Provisioning in IP/Optical Networks](#)

[10.4 Control Plane Models for IP/Optical Networking](#)

[10.4.1 Optical Internetworking Forum's Optical User Network Interface: Overlay Model](#)

[10.4.2 Internet Engineering Task Force's Generalized Multiprotocol Label Switching: Peer Model](#)

[10.5 Next-Generation MultiLayer Network Design](#)

Requirements

10.6 Benefits and Challenges in IP/Optical Networking

References

Chapter 11: IP Version 6

11.1 Addresses in IP Version 6

11.1.1 Unicast IP Addresses

11.1.2 Multicast IP Addresses

11.2 IP Packet Headers

11.3 IP Address Resolution

11.4 IP Version 6 Deployment: Drivers and Impediments

11.4.1 Need for Backwards Compatibility

11.4.2 Initial Deployment Drivers

11.4.3 Reaching a Critical Mass

References

Chapter 12: IP Traffic Engineering

12.1 Models of Traffic Demands

12.2 Optimal Routing with Multiprotocol Label Switching

12.2.1 Overview

12.2.2 Applicability of Optimal Routing

12.2.3 Network Model

12.2.4 Optimal Routing Formulations with Three Models

12.3 Link-Weight Optimization with Open Shortest Path First

12.3.1 Overview

12.3.2 Examples of Routing Control with Link Weights

12.3.3 Link-Weight Setting Against Network Failure

12.4 Extended Shortest-Path-Based Routing Schemes

12.4.1 Smart-Open Shortest Path First

12.4.2 Two-Phase Routing

12.4.3 Fine Two-Phase Routing

12.4.4 Features of Routing Schemes

References

Chapter 13: IP Network Security

13.1 Introduction

13.2 Detection of Denial-of-Service Attack

13.2.1 Backscatter Analysis

13.2.2 Multilevel Tree or Online Packet Statistics

13.3 IP Traceback

13.3.1 IP Traceback Solutions

13.4 Edge Sampling Scheme

13.5 Advanced Marking Scheme

References

Chapter 14: Mobility Support for IP

14.1 Mobility Management Approaches

14.1.1 Host Routes

14.1.2 Tunneling

14.1.3 Route Optimization

14.2 Security Threats Related to IP Mobility

14.2.1 Impersonation

14.2.2 Redirection-Based Flooding

14.2.3 Possible Solutions

14.3 Mobility Support in IPv6

14.4 Reactive Versus Proactive Mobility Support

14.5 Relation to Multihoming

14.6 Protocols Supplementing Mobility

14.6.1 Router and Subnet Prefix Discovery

14.6.2 Movement Detection

14.6.3 IP Address Configuration

14.6.4 Neighbor Unreachability Detection

14.6.5 Internet Control Message Protocol for IP Version 6

14.6.6 Optimizations

14.6.7 Media-Independent Handover Services
References

Index

ADVANCED INTERNET PROTOCOLS, SERVICES, AND APPLICATIONS

Eiji Oki

University of Electro-Communications, Japan

Roberto Rojas-Cessa

New Jersey Institute of Technology, USA

Mallikarjun Tatipamula

Ericsson Silicon Valley, USA

Christian Vogt

Ericsson Silicon Valley, USA



A JOHN WILEY & SONS, INC., PUBLICATION

Copyright © 2012 by John Wiley & Sons, Inc. All rights reserved

Published by John Wiley & Sons, Inc., Hoboken, New Jersey Published simultaneously in Canada

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning, or otherwise, except as permitted under Section 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 750-4470, or on the web at www.copyright.com. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at <http://www.wiley.com/go/permission>.

Limit of Liability/Disclaimer of Warranty: While the publisher and author have used their best efforts in preparing this book, they make no representations or warranties with respect to the accuracy or completeness of the contents of this book and specifically disclaim any implied warranties of merchantability or fitness for a particular purpose. No warranty may be created or extended by sales representatives or written sales materials. The advice and strategies contained herein may not be suitable for your situation. You should consult with a professional where appropriate. Neither the publisher nor author shall be liable for any loss of profit or any other commercial damages, including but not limited to special, incidental, consequential, or other damages.

For general information on our other products and services or for technical support, please contact our Customer Care Department within the United States at (800) 762-2974, outside the United States at (317) 572-3993 or fax (317) 572-4002.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic formats. For more information about Wiley products, visit our web site at www.wiley.com.

Library of Congress Cataloging-in-Publication Data:

Oki, Eiji, 1969-

Advanced Internet protocols, services, and applications / Eiji Oki, Roberto Rojas-Cessa, Mallikarjun Tatipamula, and Christian Vogt.

pages cm

Includes bibliographical references and index.

ISBN 978-0-470-49903-0

1. Computer network protocols. I. Rojas-Cessa, Roberto. II. Tatipamula, Mallikarjun. III. Vogt, Christian (Marketing executive) IV. Title.

TK5105.55.O54 2012

004.6-dc23

2011048524

Preface

Today, the Internet and computer networking are essential to business, learning, personal communication, and entertainment. The infrastructure that carries virtually all messages and transactions sent over the Internet is based on advanced Internet protocols. These advanced Internet protocols ensure that both public and private networks operate with maximum performance, security, and flexibility.

This book is intended to provide a comprehensive technical overview and survey of advanced Internet protocols. First, a solid introduction and discussion of internetworking technologies, architectures, and protocols is provided. The book also presents the application of these concepts into next-generation networks and discusses protection and restoration as well as various tunneling protocols and applications. Finally, emerging topics are discussed.

This book covers basic concepts in Transmission Control Protocol (TCP)/Internet Protocol (IP), Internet architecture, IP routing protocols including transport-layer protocols, IP version 6, Multiprotocol Label Switching (MPLS), networking services such as IP Quality of Service (QoS), IP Multicast, anycast, Layer-2 and Layer-3 virtual private networks (L2VPN and L3VPN), and applications such as IP over Dense Wavelength Division Multiplexing (DWDM), IP traffic engineering, IP in mobility, and IP network security.

Although there are no strict prerequisites for reading this book, a data communication background would be helpful. All the concepts in this book are developed from basics on an intuitive basis, with further insight provided through examples of real-world networks, services, and applications. The authors are well-experienced researchers and engineers from both industry and academia.

Dr. Mallikarjun Tatipamula first offered a course on Advanced Internet Protocols to graduate students at a number of leading universities in addition to tutorials at various conferences. The positive response from students xi triggered the idea of writing this book with the objective of setting a strong foundation for students regarding Internet protocols.

The authors have developed some of the contents of this book in their

graduate courses and seminars in their universities and organizations. These contents have been improved, thanks to feedback from students and industry colleagues. The courses in which this material has been used have attracted both academic and industrial practitioners, as the Internet and computer networking are key topics in the information technology industry. These are people interested in the principles of the Internet and advanced networking technologies, including designing networks and network elements and being able to consult network designers in order to satisfy their customers' needs.

Audience

This book is intended for graduate students, R&D managers, software and hardware engineers, system engineers, and telecommunications and networking professionals. This book will interest those who are currently active or anticipate future involvement in internetworking and are seeking a broad understanding and comprehensive technical overview of Internet technologies, architectures, and protocols including current status and future direction.

Organization

The book is organized as follows.

- Chapter 1 describes Transmission Control Protocol (TCP)/Internet Protocol (IP), which is an Internet protocol stack that enables communications between two computers, or hosts, through the Internet. It is a collection of different protocols. A protocol is a set of rules that controls the way data is sent between hosts.
- Chapter 2 explains protocols in the transport layer, which is the fourth layer of the Open Systems Interconnection (OSI) reference model. Transparent transfer of data between end users using services of the network layer is provided. The well-known protocols in this layer are TCP, User Datagram Protocol (UDP), Stream Control Transmission Protocol (SCTP), and Real-time Transport Protocol (RTP).
- Chapter 3 describes Internet architecture, including basic Internet topology, Internet exchange points (IXPs), the history of IXPs, and

the principles of Internet relationships and Internet service providers (ISPs).

- Chapter 4 describes IP routing protocol including an overview of Interior Gateway Protocols (IGPs) and Exterior Gateway Protocols (EGPs). Routing Information Protocol (RIP) and Open Shortest Path First (OSPF) is covered as an IGP, and Boarder Gateway Protocol (BGP) is covered as an EGP.
- Chapter 5 describes Multiprotocol Label Switching (MPLS) technologies, which enable networks to perform traffic engineering, resident communications, virtual private networks, Ethernet emulation, and so on. First, an overview to MPLS is given. Next, the functions and mechanisms of MPLS are described. Finally, MPLS applicabilities are discussed.
- Chapter 6 presents a myriad of concepts and paradigms used to provision quality of service by the Internet. It includes discussions on the implementation of mechanisms of traffic differentiators and policy and traffic polices and regulators. The discussion includes Diffserv, IntServ, and a combination of both as developed by recent research.
- Chapter 7 describes IP multicast and anycast. The majority of the traffic on the Internet is *unicast*: one source device sending to one destination device. However, where the number of receivers is more than one, multicast traffic can be transmitted over the Internet in ways that avoid loading the network with excessive traffic. The multicast mechanisms present the possibility of having multiple receivers for routing and packet delivery.
- Chapter 8 presents several Layer-2 encapsulation protocols. These encapsulation protocols are used to allow Layer-2 connectivity through nonadjacent networks. These protocols allow remote networks to communicate as if they were in the same network. Several examples of these encapsulation protocols are presented including Martini, Layer-2 Transport, and Pseudowire Emulation Edge to Edge protocol.
- Chapter 9 introduces point-to-point virtual connectivity and virtual broadcast access connectivity, such as virtual local area networks (LANs). Layer-2 virtual private network over transport can provide several features such as redundancy against failures and controlled

connectivity through a packet-switched network or Generalized MPLS (GMPLS). Multiple Layer-2 access protocols can be virtually emulated.

- Chapter 10 introduces the multilayer network evolution, followed by discussion on limitations in current legacy networks, automated provisioning in IP/optical networks, the proposed control plane techniques in the industry, key design requirements for next-generation multilayer IP/optical networks, and comparisons of the proposed control plane models against these key design requirements.
- Chapter 11 describes the main differences between IPv4 and IPv6, investigates the reasons for the lagging uptake of IPv6 so far, and explains why, contrary to widespread belief, IPv6's large address space alone is insufficient to drive early IPv6 deployment.
- Chapter 12 describes several routing schemes to maximize the network utilization for various traffic demand models for MPLS and OSPF technologies. One useful approach to enhancing routing performance is to minimize the maximum link utilization rate, also called the network congestion ratio, of all network links. Minimizing the network congestion ratio leads to an increase in admissible traffic.
- Chapter 13 discusses some popular threats and how they exploit protocol vulnerabilities. Some counter measures, based on additional algorithms that either work as an application or as a protocol at the network or transport layers, are discussed including methods to trace back threatening hosts.
- Chapter 14 explains the different ways of augmenting the Internet for mobility support, as well as the security threats that need to be considered as part of this. The relationship between mobility and multihoming will be explored further. The chapter also gives an overview of auxiliary protocols that play a fundamental role in IP mobility management.

Eiji Oki
Roberto Rojas-Cessa
Mallikarjun Tatipamula
Christian Vogt

Acknowledgments

This book could not have been published without the help of many people. We thank them for their efforts in improving the quality of the book. We have done our best to accurately describe advanced Internet protocols, services, and applications as well as the basic concepts. We alone are responsible for any remaining errors. If any error is found, please send an e-mail to eiji.oki@uec.ac.jp and rojas@njit.edu. We will correct them in future editions.

Several chapters of the book are based on our research works. We would like to thank the people who have contributed materials to some chapters, especially Dr. Nattapong Kitsuwan (UEC Tokyo), Mohammad Kamrul Islam (UEC Tokyo), Khondaker M. Salehin (New Jersey Institute of Technology), and Tuhina Sarkar (New Jersey Institute of Technology).

The entire manuscript draft was reviewed by Katsuhiro Amako (UEC Tokyo), Dr. Neda Beheshti (Ericsson), Komlan Ego (New Jersey Institute of Technology), Prof. Ziqian Dong (New York Institute of Technology), Agostinho A. Jose (UEC Tokyo), Prof. Chuan-Bi Lin (Chaoyang University of Technology, Taiwan), Abu Hena Al Muktadir (UEC Tokyo), Ihsen Aziz Ouédraogo (UEC Tokyo), Dr. Kiran Yedavalli (Ericsson), and Dr. Ying Zhang (Ericsson). We are immensely grateful for their comments and suggestions.

Eiji wishes to thank his wife, Naoko, his daughter, Kanako, and his son, Shunji, for their love and support. Roberto would like to thank his wife, Vatcharapan, and his children, Marco and Nelli, for their unconditional understanding, love, and support. Mallikarjun is grateful to his wife, Latha, and his children, Sashank, Santosh, and Vaishnavi, for their love and support.

Eiji Oki
Roberto Rojas-Cessa
Mallikarjun Tatipamula
Christian Vogt

About the Authors

Eiji Oki is an Associate Professor at the University of Electro-Communications, Tokyo, Japan. He received his Bachelor and Master of Engineering degrees in Instrumentation Engineering and a Doctorate of Philosophy in Electrical Engineering from Keio University, Yokohama, Japan, in 1991, 1993, and 1999, respectively. In 1993, he joined Nippon Telegraph and Telephone Corporation (NTT) Communication Switching Laboratories, Tokyo, Japan. He has been researching network design and control, traffic-control methods, and high-speed switching systems. From 2000 to 2001, he was a Visiting Scholar at the Polytechnic Institute of New York University, Brooklyn, New York, where he was involved in designing terabit switch/router systems. He was engaged in researching and developing high-speed optical IP backbone networks with NTT Laboratories. He joined the University of Electro-Communications, Tokyo, Japan, in July 2008. He has been active in standardization of path computation element (PCE) and GMPLS in the Internet Engineering Task Force (IETF). He wrote 11 IETF RFCs. He served as a guest co-editor for the special issue on “Multi-Domain Optical Networks: Issues and Challenges,” June 2008, in IEEE Communications Magazine; a guest co-editor for the special issue on Routing, “Path Computation and Traffic Engineering in Future Internet,” December 2007, in the Journal of Communications and Networks; a guest co-editor for the special section on “Photonic Network Technologies in Terabit Network Era,” April 2011, in IEICE Transactions on Communications; a Technical Program Committee (TPC) Co-Chair for the Workshop on High-Performance Switching and Routing in 2006 and 2010; a Track Co-Chair on Optical Networking for ICCCN 2009; a TPC Co-Chair for the International Conference on IP+Optical Network (iPOP 2010); and a Co-Chair of Optical Networks and Systems Symposium for IEEE ICC 2011. Professor Oki was the recipient of the 1998 Switching System Research Award and the 1999 Excellent Paper Award presented by IEICE, the 2001 xvii Asia-Pacific Outstanding Young Researcher Award presented by IEEE Communications Society for his contribution to broadband network, ATM, and optical IP technologies, and the 2010 Telecom System Technology Prize by the Telecommunications Advanced Foundation. He has coauthored two books,

Broadband Packet Switching Technologies, published by John Wiley & Sons, Inc., New York, in 2001, and *GMPLS Technologies*, published by CRC Press, Boca Raton, Florida, in 2005. He is an IEEE Senior Member.

Roberto Rojas-Cessa received a Master of Computer Engineering degree and a Doctorate of Philosophy in Electrical Engineering from the Polytechnic Institute of New York University, Brooklyn, New York. He also received a Master of Science in Electrical Engineering from the Research and Advanced Studies Center (CIVESTAV) in Mexico. He received his Bachelor of Science in Electronic Instrumentation from Universidad Veracruzana, Mexico. Currently, he is an Associate Professor in the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, New Jersey. He was an Adjunct Professor and a Research Associate in the Department of Electrical and Computer Engineering of Polytechnic Institute of New York University. He has been involved in design and implementation of application-specific integrated-circuits (ASIC) for biomedical applications and high-speed computer communications, and in the development of high-performance and scalable packet switches and reliable switches. He was part of the team designing a 40 Tb/s core router at Coree, Inc. in Tinton Falls, NJ. His research interests include high-speed switching and routing, fault tolerance, quality-of-service networks, network measurements, and distributed systems. His research has been funded by the U.S. National Science Foundation and Industry. He was the recipient of the Advance in Research Excellence of the ECE Department in 2004. He has served on several technical committees for IEEE conferences and as a reviewer for several IEEE journals. He has been a reviewer and panelist for the U.S. National Science Foundation and the U.S. Department of Energy. He has more than 10 years of experience in teaching Internet protocols and computer communications. Currently, he is the Director of the Networking Research Laboratory at the ECE Department and the Coordinator of the Networking Research Focus Area Group of the same department.

Mallikarjun Tatipamula is Head of Packet Technologies Research, Ericsson Silicon Valley. He leads a research team and is responsible for innovation and

implementation of leading-edge technologies including Openflow, next-generation routing architectures, application-aware networking, and Cloud computing, networking, and services. He closely works with leaders from universities, research and education networks, and service providers around the world. Prior to his role at Ericsson, he was Vice President and Head of Service Provider Sector at Juniper Networks, Sunnyvale, California. His team responsibilities include new technologies, architectures, standards, solutions, and creation of business strategy for the implementation of next-generation products in content delivery network/video, Cloud, IP/optical integration, security, mobility, and convergence. Prior to Juniper, he was with Cisco systems for over eight years and made active contributions to the Cisco IP NGN strategy. These contributions include Service Exchange Framework, IMS/FMC, advanced technologies such as IPv6, GMPLS, multicast, along with his contributions in the early days of VoIP, mobile wireless and IP/optical integration that led to definition and implementation plans. Prior to Cisco, Mallikarjun was a principal engineer at Motorola in the Cellular Infrastructure Group. He was responsible for defining system architecture for advanced wireless and satellite systems. From 1993 to 1997 he was a senior member of the scientific staff at BNR (now Nortel), Ottawa, responsible for development of Nortel's optical (OC12/OC48) products and involved in the development of the Nortel CDMA Base Station. From 1990 to 1992, Mallikarjun was with Indian Telephone Industries as an Assistant Executive Engineer in Optical Transmission R&D Laboratories and Indian Institute of Technology, Chennai, as a Senior Project Officer in the Fiber Optics Labs, responsible for development of optical data links. He is a key note speaker at leading telecommunications and networking events. Mallikarjun obtained Doctorate of Philosophy in Information Science and Technology from the University of Tokyo, Japan, a Master of Science in Communication Systems and High Frequency Technologies from the Indian Institute of Technology, Chennai, and a Bachelor of Technology in Electronics and Communication Engineering from NIT, Warangal, India. Mallikarjun has authored or coauthored a number of patents and publications with leaders from NRENs and service providers. He is a trusted partner, advisor, and thought leader. Mallikarjun is a lead editor for "Multimedia Communication Networks: Technologies and Services," by Artech House Publishers. He is a senior member of IEEE and his biography appeared in Marquis Who is Who in the World, Who is Who in America, and Who is Who Science and Engineering.

Mallikarjun delivered distinguished lectures at leading universities including Stanford University, the University of Tokyo, the Tokyo Institute of Technology, Tsing Hua University, Beijing University, China, and IIT Delhi.

Christian Vogt is a Senior Marketing Manager at Ericsson Silicon Valley. He works on product and marketing strategies regarding the convergence of wireless and wireline networks, and their transition from IP version 4 to 6. His technical interests further include Internet routing and addressing, IP mobility and multihoming, related security aspects, as well as next-generation Internet architectures. As part of this work, Christian co-chairs the Internet Area and Source Address Validation Improvements working groups in the Internet Engineering Task Force. Christian received his doctoral degree from the University of Karlsruhe in Germany in 2007 for his dissertation on efficient and secure mobility support in IP version 6. He also holds a Master of Science in Computer Science from the University of Southern California, Los Angeles, and a German Diplom in Computer Science from the University of Bonn. Currently, Christian is pursuing an Executive Master of Business Administration degree at the Wharton School, University of Pennsylvania.

Chapter 1

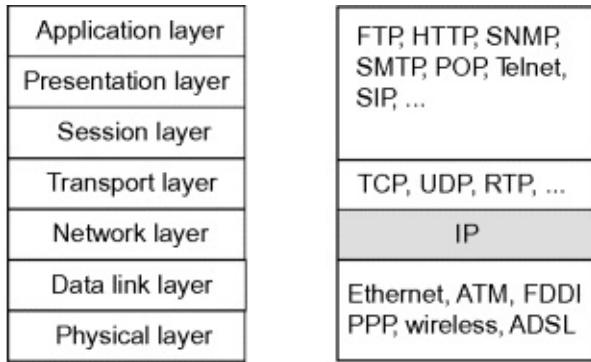
Transmission Control Protocol/Internet Protocol Overview

This first chapter provides an overview of Transmission Control Protocol (TCP)/Internet Protocol (IP), which is an Internet protocol stack to perform communications between two computers, or hosts, through the Internet. It is a collection of different protocols. A *protocol* is a set of rules that controls the way data is transmitted between hosts.

1.1 Fundamental Architecture

[Figure 1.1](#) shows the Open Systems Interconnection (OSI) reference model and the Internet protocol stack. The International Standardization Organization (ISO) specifies a guideline called the OSI reference model. It is an abstract description for layered communications and computer network protocol design. It consists of seven layers, which are, from the bottom, physical, data link, network, transport, session, presentation, and application, as shown in [Figure 1.1a](#). The application layer is the OSI layer closest to the end user, which means that both the OSI application layer and the user interact directly with the software application. This layer interacts with software applications that implement a communicating component. At the physical layer, data is recognized, or handled, as bits. At the data link layer, data is handled as frames. At the network layer, data is recognized as packets. In the transport layer, data is handled as segments or datagrams. In the remaining upper layers, users' information is recognized. 3

[Figure 1.1](#) Layer model.



(a) OSI reference model

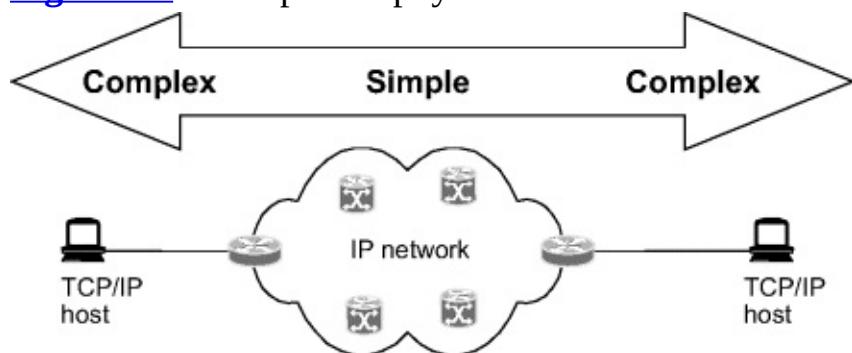
(b) Internet protocol stack

The Internet protocol stack is shown in [Figure 1.1b](#), where we can see to which layer in the OSI reference model each protocol corresponds. For example, the Internet protocol corresponds to the network layer. TCP corresponds to the transport layer. The Internet protocol stack that includes several protocols, as shown in [Figure 1.1b](#), is referred to as *TCP/IP*.

TCP/IP is being standardized by the Internet Engineering Task Force (IETF). It is widely used as the *de facto* standard protocol for building network equipment and internetworking. If the TCP/IP standard is used, computers can communicate with each other regardless of hardware and operating system.

[Figure 1.2](#) shows a basic philosophy of TCP/IP. In the TCP/IP philosophy, the IP core network functions simply and quickly, while the functionality of the edges surrounding the core is complex. Edges can be hosts, edge routers, network boundaries, etc. The IP network based on this philosophy is scalable and flexible. This enables different complexities at the edge.

[Figure 1.2](#) Basic philosophy of TCP/IP.



Application data and customer traffic can be transmitted on the IP layer because IP is the least common denominator, as shown in [Figure 1.3](#). IP supports transport-layer protocols such as TCP, User Datagram Protocol

(UDP), Real Time Protocol (RTP)/UDP, and Stream Control Transmission Protocol (SCTP), as shown in [Figure 1.4](#). IP works on link-layer protocols such as Ethernet, Point-to-Point Protocol (PPP) over Asynchronous Transfer Mode (ATM), satellite, wireless, optical, and IP/Multiprotocol Label Switching (MPLS), as shown in [Figure 1.5](#).

Figure 1.3 IP is the least common denominator.

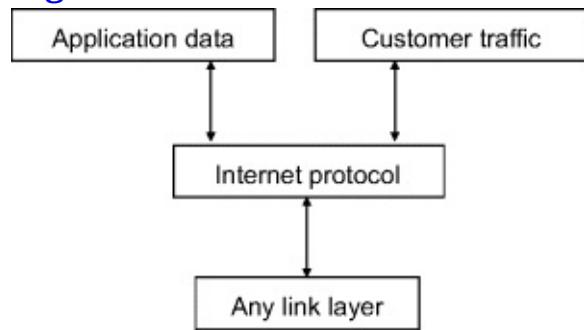


Figure 1.4 Any transport-layer protocols on IP.

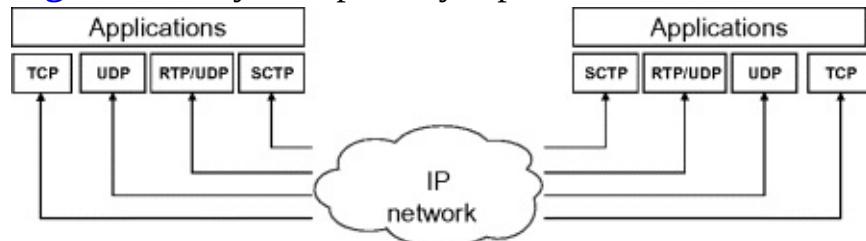
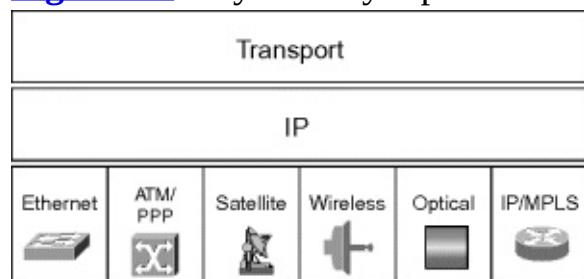
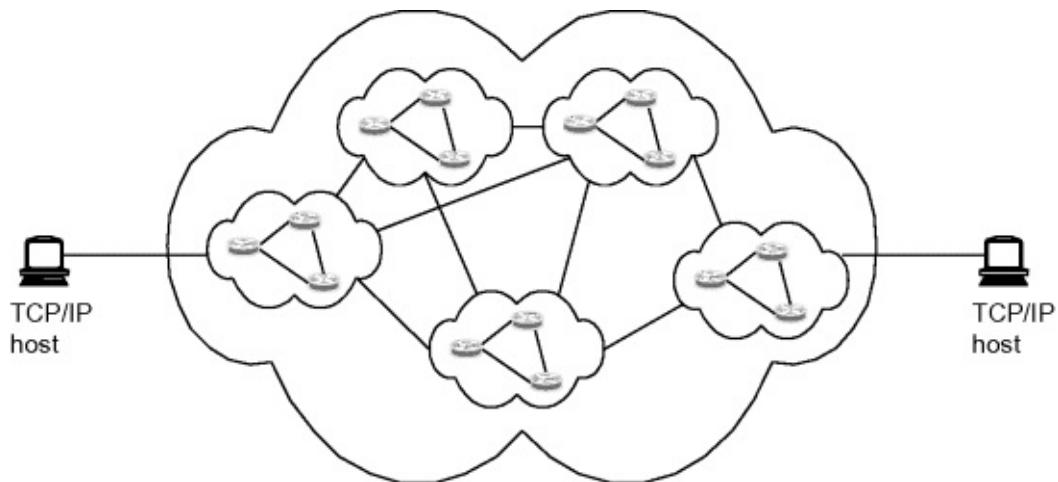


Figure 1.5 Any link-layer protocols under IP.



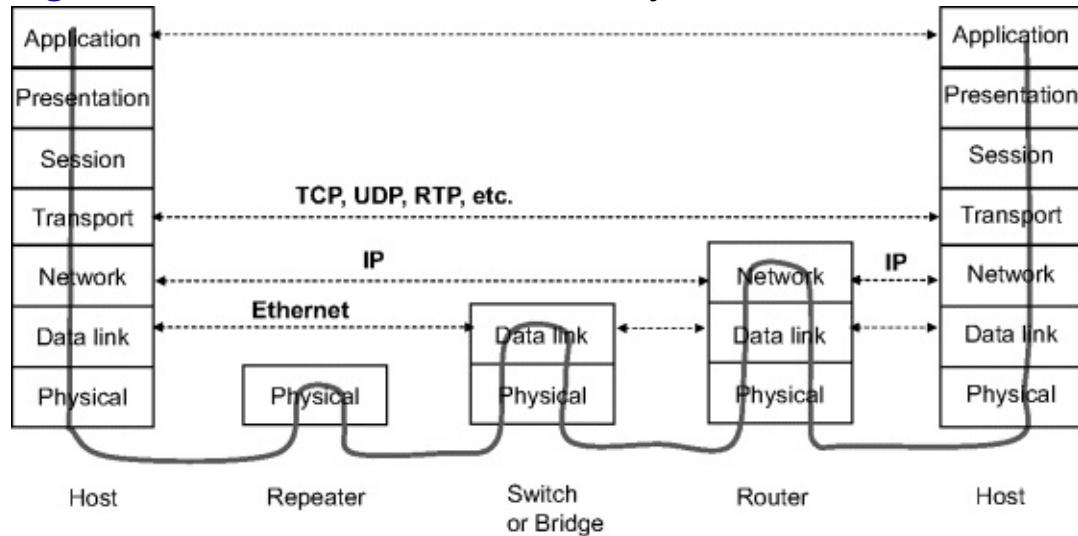
More than one network can be connected via IP, as shown in [Figure 1.6](#). IP enables building a large-scale network where smaller networks, or subnetworks, are concatenated to build one large network. Inside each network, a node does not need to be connected with other nodes in the same network via IP. This is the way the Internet is built.

Figure 1.6 Interworking via IP.



[Figure 1.7](#) shows the protocol stack based on the OSI reference model and network equipment and protocols connecting layers. A repeater is a functional device of layer 1. It enhances signal level. A bridge, or switch, processes frames, for example Ethernet frames, corresponding to layer 2. Two bridges are connected via Ethernet protocol. A router processes IP packets corresponding to layer 3. Two routers are connected via IP. Two hosts are connected via protocols for layers 4, 5, 6, and 7. For example, TCP, UDP, and RTP are used to connect two hosts on layer 4.

[Figure 1.7](#) Protocol stack and connectivity.

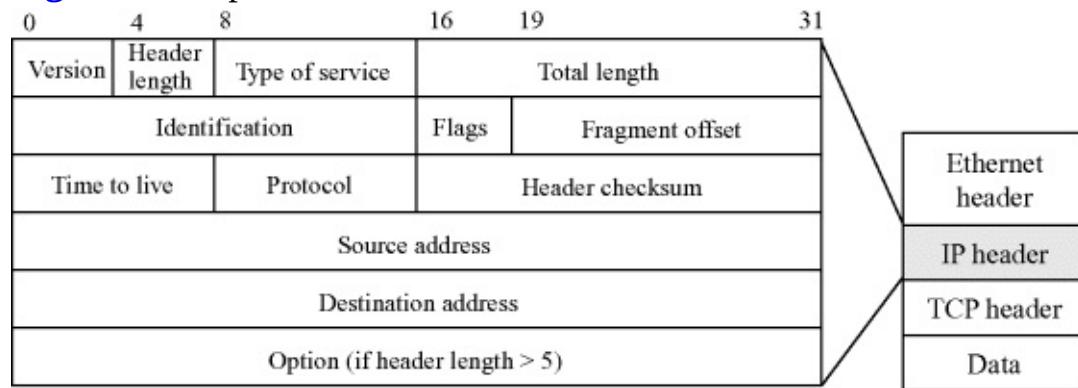


1.2 Internet Protocol Basics

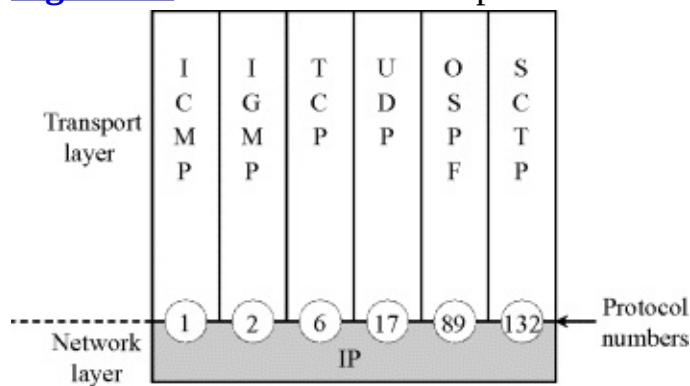
Internet Protocol is a protocol in the network layer. It is used for sending data from a source host to its destination host where each host has a unique

number. The IP header format is shown in [Figure 1.8](#). Version specifies the IP version. Header length indicates the size of the header. Type of service is used to guide the selection of the actual service parameters. Total length is the size of the IP packet including header and data. Identification is used for a particular purpose such as experimental work. Flag is used to control or identify fragments. Time-to-live (TTL) is a countdown field; every station must decrement this number by one or by the number of seconds it holds onto the packet. When the counter reaches zero, the TTL expires and the packet is dropped. Protocol specifies the protocol that is used to operate the data. There are several protocols dependent on applications. Example of well-known protocols are shown in [Figure 1.9](#). Header checksum is used for error checking. Source address and destination address specify the IP address of the source and the destination, respectively.

[Figure 1.8](#) IP packet header format.



[Figure 1.9](#) Some well-known protocols.

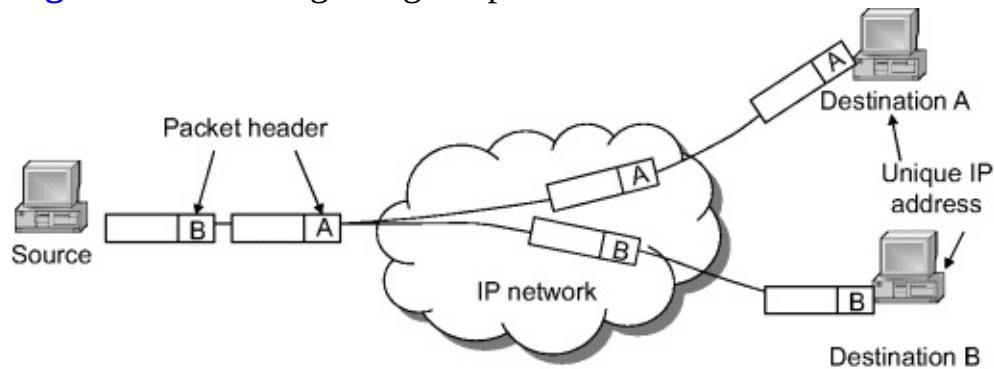


1.2.1 Packet Header

When data is transmitted from a source host to a destination host in a network, it is divided into small pieces. Each piece is called a packet in an IP

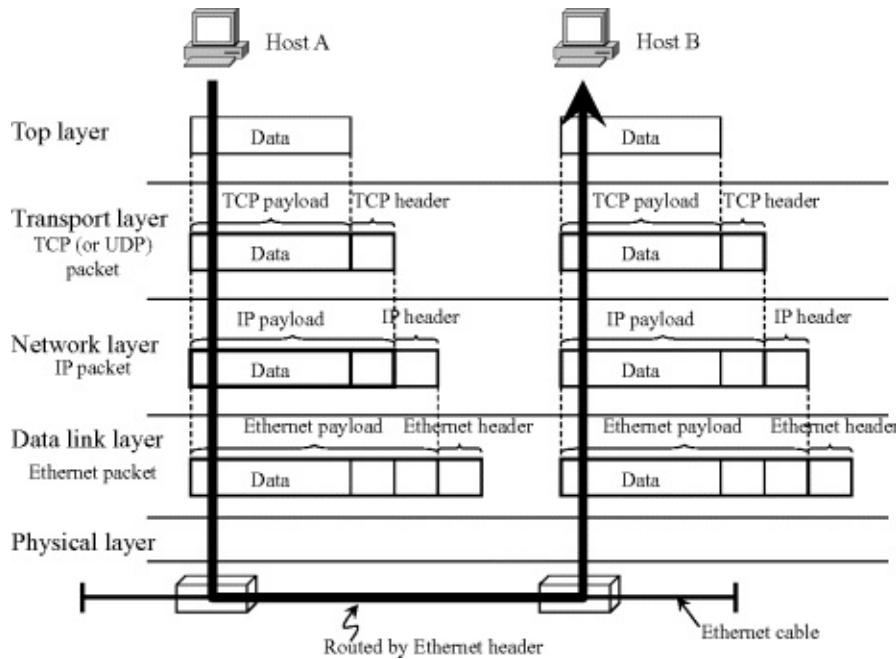
network. Sometimes it is called a frame, block, cell, or segment. The packet consists of a header and a payload. The header contains information such as the destination identification and length of the packet. The payload contains part of the body of the message. The packet is sent to the destination through an IP network along an appropriate available route. Each destination is assigned a unique identification number called an IP address. [Figure 1.10](#) shows an example of routing using the information in the packet header. Packets are sent to destinations A and B, referring to the IP addresses.

Figure 1.10 Routing using the packet header information.



[Figure 1.11](#) shows how the data is wrapped when it is transferred to other layers. On the top layer, host A sends the raw data to host B. First, the raw data is wrapped as a TCP structure called the TCP packet. This raw data becomes the TCP payload, and the TCP header, which includes basic information about the TCP packet, is added. Next, the TCP packet is transferred from the transport layer to the network layer. In the network layer, the packet in the network layer is called an IP packet. Here, the TCP packet becomes the IP payload, and the IP header, which includes basic information about the IP packet, is added. At this point, the IP packet is transferred to the data link layer, where it is wrapped into an Ethernet packet. The IP packet becomes the Ethernet payload, and the ethernet header, which includes basic information about the Ethernet packet, is added. When the Ethernet packet is transferred back to the network layer, the Ethernet header is unwrapped and remains that way until the data reaches host B.

Figure 1.11 Wrapping and unwrapping the data as layers change.



1.2.2 Internet Protocol Address

An IP address is a unique number assigned to each device in the computer network utilized by the Internet protocol.

In IP version 4 (IPv4), an IP address is defined by using 32 bits of binary numbers, which yields $4,294,967,296 (2^{32})$ numbers. To enhance readability, an IP address is often represented as a decimal using four 8-bit sections, or octets. The number in each octet ranges between 0 and 255.

An IP address consists of the network address, which is represented by the higher order bits, and the host address, which is represented by the lower order bits, as shown in [Figure 1.12](#). Based on control purpose, the number of bits in the network and host addresses are changeable. [Figure 1.13](#) is an example of an IP address, 10.28.1.222. The number in each octet is converted to an 8-bit binary number.

[**Figure 1.12**](#) IP address structure.

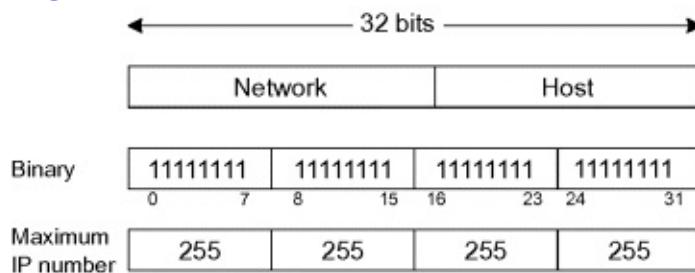


Figure 1.13 Example of IP address.

Decimal	10	28	1	222
Binary	00001010	00011100	00000001	11011110

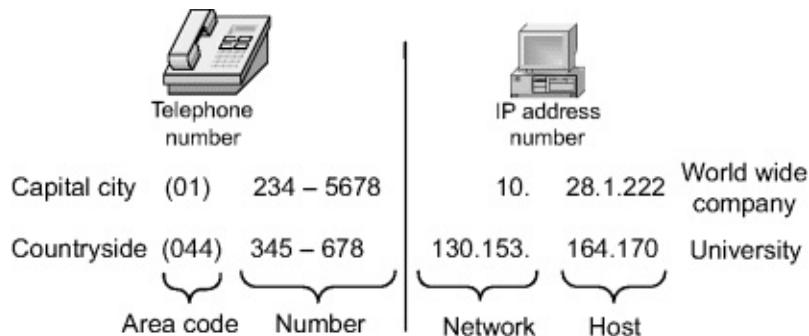
1.2.3 Internet Protocol Classification

IP addresses are classified to enable regularity of management. They are limited to only 32 bits and are comprised of a network and host address. The number of hosts is lower if more bits are allocated for the network, and conversely, the number of networks is lower if more bits are allocated for the hosts.

The network and host addresses in the IP address system are similar to the area code and the client number, respectively, in the telephone system of countries such as the United Kingdom, Japan, India, and Thailand. There are many users in cities, while there are fewer users in small towns. Since the number of capital cities is smaller than the number of towns, fewer digits of area code are enough to support the cities. More digits are needed in the area codes of towns.

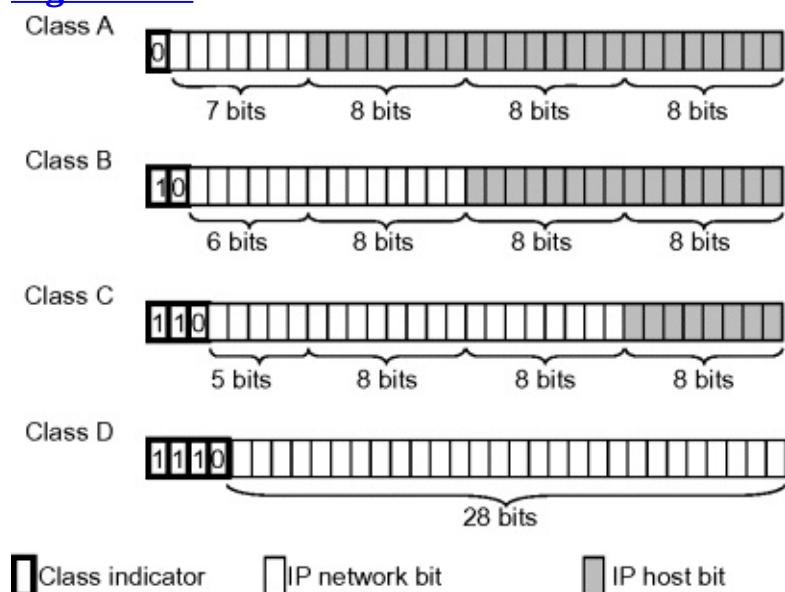
In [Figure 1.14](#), an example of the telephone number system and IP address system are compared. The telephone system contains nine digits. In the capital city, the first two digits specify the area code and the last seven digits specify the client number. This allows for more than 10 million numerical combinations. In the countryside, the first three digits are used for the area code and the last six digits are for the client number. This allows for approximately 1 million numerical combinations. In the IP address system, a company with branches throughout the world would have several hosts. A university located in a single country would require fewer hosts than the global company. The company would use the first octet to specify a network address and the last three octets for the host address. The network address of the university would be indicated by the first two octets, and the last two octets would indicate the host address.

Figure 1.14 Comparison of the structure of the telephone number and IP address.



The IP address can be classified into four classes: A, B, C, and D. [Figure 1.15](#) shows IP addresses of each class.

Figure 1.15 IP classification.



- *Class A* is indicated when the starting bit of “0.” The first eight bits specify a network address. The last 24 bits specify a host address. The range of the number in the first octet is between 0 and 127. However, 126 numbers are usable since IP addresses starting with 0 and 127 are reserved. 16,777,214 hosts can be obtained for one network.
- *Class B* is indicated when the first two bits are “10.” The first 16 bits specify a network address and the other 16 bits specify a host address. The range of the number in the first octet is between 128 and 191. 16,384 networks address are available. 65,532 hosts can be obtained for each network.
- *Class C* is indicated when the first three bits are “110.” The first 24 bits specify a network address and the other eight bits specify host

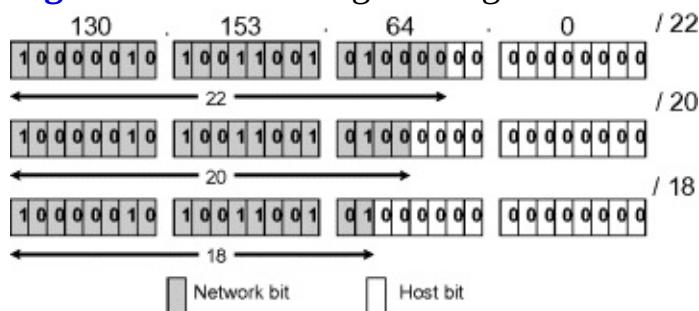
address. The range of the number in the first octet is between 192 and 223. 2,097,152 networks address are available. 254 hosts can be obtained for each network.

- *Class D* is indicated when the first four bits are “1110.” This class is used for multicast purpose. The range of the number in the first octet is between 224 and 239.

1.2.4 Subnet and its Masking

Subnet is a way to group, or subnetwork, IP addresses in the same network. Subnet mask is a parameter to indicate network address, by bit “1,” and host address, by bit “0.” Usually, the network address and the host address are separated based on IP class as explained in the previous subsection. However, the mask can vary due to IP address management. The number of the subnet mask can be written by “/xx” after the IP address, where xx is the number of network bits. [Figure 1.16](#) shows an example of the IP address 130.153.64.0. If the first 22 bits are masked as a network address, it is notated as 130.153.64.0/22. The network address is also called a prefix.

[Figure 1.16](#) Subnetting to categorize hosts.

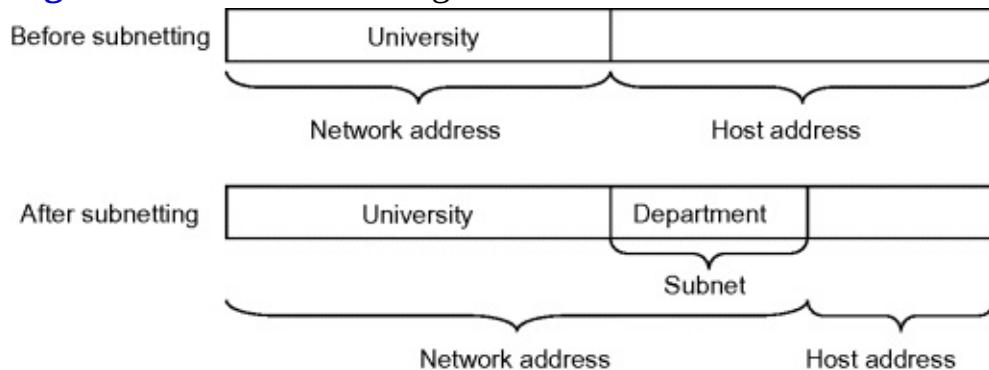


There are two main functions of subnetting:

- *To categorize hosts into a group.* [Figure 1.17](#) illustrates subnetting for a university network. Before subnetting, the IP address has a network address, which indicates the IP number of the university, and a host address, which is the address of the hosts in the university. If the IP address is randomly distributed to hosts, it is difficult to manage IP addresses since IP addresses are not classified. For example, if the computer engineering department allows File Transfer Protocol (FTP) service while it is blocked in the electrical engineering department, it is necessary to search the IP addresses that belong to electrical engineering and block each and

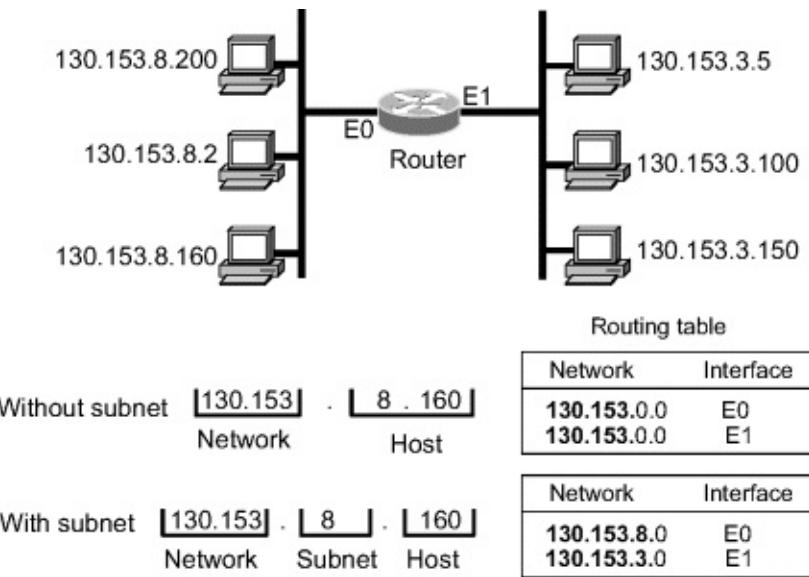
every IP. By subnetting, it is possible to block the group of IP addresses that belong to electrical engineering. In [Figure 1.17](#), subnetting is used to divide departments. After subnetting by using bits to indicate the department, the new network address includes the department subnet number.

Figure 1.17 Subnet masking.



- *To support a routing table.* A switch uses this table to forward data to a specific port. [Figure 1.18](#) shows routing tables with and without subnetting. Hosts with IP 130.153.8.XXX are connected to the port *E0* of the router. Hosts with IP 130.153.3.XXX are connected to the router at the *E1* port. Based on IP classification, the network address of these IP addresses is 130.153.0.0 because these IPs are class B. In the routing table without subnetting, the data that requests a host with IP address 130.153.8.160, for which the network address is 130.153.0.0, is forwarded to both ports *E0* and *E1*. This is incorrect because the host with IP address 130.153.8.160 is connected at the *E0* port of the router. With subnetting, the first 24 bits are masked as a network address and the last 8 bits are a host address. In the routing table, the data that requests a host with network address 130.153.8.0 is forwarded to port *E0* and 130.153.3.0 to port *E1*. The router can distinguish the IP addresses and forward to the correct port.

Figure 1.18 Subnetting to support a routing table.



1.2.5 Subnet Calculation

This section explains how to calculate the subnet from a specific IP address. If we know a host IP address and subnet mask, what can we obtain from the subnet calculation?

- *Subnet IP* is an address number that indicates a subnetwork within a larger network. Subnet IP = (Host IP) AND (Subnet mask). *Trick:* If the bit of a subnet mask is “1,” the bit of the subnet IP is the same as the bit of the host IP. Otherwise, the bit of the subnet IP is “0.”
- *Broadcast IP* is used to send information to all hosts that belong to the same network. Broadcast IP = (Inverted Subnet Mask) OR (Subnet IP). *Trick:* It is obtained by changing all of the “0” bits in the host address of the subnet IP to “1.”
- *First* and *last IP* indicate the range of IP addresses that hosts on the same network can use. The first IP = (Subnet IP) + 1, and the last IP = (Broadcast IP) – 1.
- *Number of hosts* indicates the number of hosts on the subnetwork. It is counted from the first IP to the last IP, which is (Last IP) – (First IP) + 1.

[Figure 1.19](#) is an example of subnet calculation for 130.153.2.160/26. “/26” represents the number of bits in the mask. It indicates that 26 bits are required for the network address. The subnet mask becomes 11111111.11111111.11111111.11000000 (255.255.255.192 in decimal). The subnet IP is 130.153.2.160 AND 255.255.255.192 = 130.153.2.128. To

calculate broadcast IP, the inverted subnet mask is needed. The inverted subnet mask is 00000000.00000000.00000000.00111111 (0.0.0.63 in decimal). Broadcast IP is 0.0.0.63 OR 130.153.2.128 = 130.153.2.191. The first IP is 130.153.2.128 + 1 = 130.153.2.129 and the last IP is 130.153.2.191 - 1 = 130.153.2.190. The number of hosts available on this subnetwork is 130.153.2.190 - 130.153.2.129 + 1 = 62 hosts.

Figure 1.19 An example of subnet calculation.

	Network address				Host address
Host IP	10000010	10011001	00000010	10 100000	
	130	. 153	. 2	. 160	
Subnet mask	11111111	11111111	11111111	11 0000000	
	255	. 255	. 255	. 192	
Subnet IP	10000010	10011001	00000010	10 0000000	
	130	. 153	. 2	. 128	
Inverted subnet mask	00000000	00000000	00000000	00 1111111	
	0	. 0	. 0	. 63	
Broadcast IP	10000010	10011001	00000010	10 1111111	
	130	. 153	. 2	. 191	
First IP	10000010	10011001	00000010	10 0000001	
	130	. 153	. 2	. 129	
Last IP	10000010	10011001	00000010	10 1111111	
	130	. 153	. 2	. 190	

The subnet calculation is also used to design IP addresses for the hosts in the network. The following example provides insight into how to design the IP addresses.

How do you design the IP address to support these requirements for XYZ branch if you are a network administrator?

- ABC company has several branches throughout the world.
- XYZ branch uses IP 10.28.1.xxx.
- At XYZ branch, each department requested the following number of computers:

Information System (IS) 18 computers

Account (ACC) 16 computers

Human resource (HR) 12 computers

Production 25 computers

General affair (GA) 10 computers

The IP 10.28.1.xxx is a class A address. The network IP is 10.0.0.0.

However, the IP of XYZ branch starts with 10.28.1.xxx. Production requests 25 computers, which is the highest number of any department. Using five bits as the host is enough to support 2^5 , or 32, IP addresses, including subnet IP and broadcast IP, and enough for five departments. Therefore, we use five bits for the host address and the remaining 27 bits for network address. The IP address in this network is 10.28.1.xxx/27. The subnet mask becomes 11111111.11111111.11111111.11100000 (255.255.255.224 in decimal). Information about subnet IP, broadcast IP, first IP, and last IP is shown in [Figure 1.20](#).

Figure 1.20 IP address information for each department.

	Subnet IP				Broadcast IP	First IP	Last IP
Subnet mask	11111111 . 11111111 . 11111111 . 11100000						
	255	255	255	224			
IS	00001010	. 00011100	. 00000001	. 00000000			
	10	. 28	. 1	. 0	10.28.1.31	10.28.1.1	10.28.1.30
ACC	00001010	. 00011100	. 00000001	. 00100000			
	10	. 28	. 1	. 32	10.28.1.63	10.28.1.33	10.28.1.62
HR	00001010	. 00011100	. 00000001	. 01000000			
	10	. 28	. 1	. 64	10.28.1.95	10.28.1.65	10.28.1.94
Production	00001010	. 00011100	. 00000001	. 01100000			
	10	. 28	. 1	. 96	10.28.1.127	10.28.1.97	10.28.1.126
GA	00001010	. 00011100	. 00000001	. 10000000			
	10	. 28	. 1	. 128	10.28.1.159	10.28.1.129	10.28.1.158

1.3 Routing

The process of forwarding a packet across the Internet from the source node to the destination node is called *routing*. Routing proceeds in a hop-by-hop manner, where each node on the packet's path, from the source node via routers to the destination node, independently determines a neighbor closer to the destination node and hands the packet on to that neighbor. The routing process completes when that neighbor happens to be the destination node itself.

Routers maintain a *routing table* that maps a packet's IP destination address onto the IP address of a neighboring router closer to the destination node, or

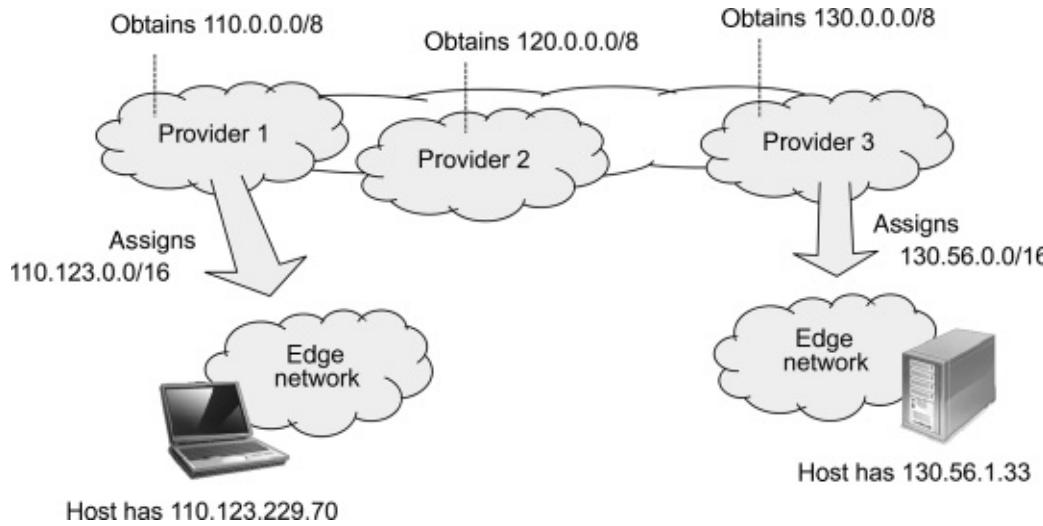
to determine that the destination node is itself a neighbor. A routing table may be manually preconfigured into a router, but in general, routers participate in a routing protocol to exchange the topological information they need to build a routing table automatically. Given that the prefix of an IP destination address already locates the destination node, it is typically sufficient for a router to perform a routing table look-up based on only the IP address prefix. The interface identifier of the IP destination address is needed only when the packet is delivered to the destination node in the final forwarding step.

To understand the Internet routing system, it is useful to differentiate between two types of networks: networks that carry transit traffic for other networks, or so-called *providers*, and networks that do not, or so-called *edge networks*. Whereas edge networks can simply forward packets to their respective provider for further transportation toward the destination node, providers need to know about the interconnectivity of other providers in order to make forwarding decisions. The routing across providers and within edge networks is explained in the proceeding section.

1.3.1 Routing Across Providers

Providers need to explore Internet topology at large in order to make routing decisions. They use the Border Gateway Protocol (BGP) to tell each other where they are located and which providers they are able to reach [1]. Providers identify themselves in BGP through their IP address prefixes. This is possible because the IP address prefixes of different providers are mutually disjointed. [Figure 1.21](#) illustrates three providers with IP address prefixes 110.0.0.0/8, 120.0.0.0/8, and 130.0.0.0/8, respectively.

[Figure 1.21](#) Provider-assigned addressing.



Each provider gathers the information received through BGP in a global routing table. For example, in [Figure 1.21](#), the global routing table of provider 1 shows that provider 2 is neighboring to the right and that provider 3 can be reached via provider 2. Therefore, provider 1 sends rightward all packets destined to IP address prefixes 120.0.0.0/8 and 130.0.0.0/8. The global routing table of provider 2 points to the right for provider 3 and to the left for provider 1. Provider 2 thus forwards packets destined to IP address prefixes 110.0.0.0/8 and 130.0.0.0/8 to the left and to the right, respectively. Finally, the global routing table of provider 3 shows that providers 1 and 2 can both be reached to the left. So provider 3 forwards leftward all packets destined to IP address prefixes 110.0.0.0/8 and 120.0.0.0/8.

In addition to BGP, internal routing protocols are used by routers within the same provider's network to jointly explore their local interconnectivity. The most common internal routing protocol is known as *Open Shortest Path First* (OSPF) [2, 3]. There are three important differences between BGP and OSPF:

- *Scope of operation*: BGP is used to explore interconnectivity between provider networks, and hence to explore Internet topology at large, whereas OSPF is used to explore the interconnectivity of routers within the same network. BGP's scope of operation is therefore global, while OSPF's scope of operation is local.
- *Routing policy*: Since BGP operates across the borders of administratively separate domains, it becomes a tool to express routing policy. OSPF, on the other hand, is not used to express routing policy because it operates within the borders of a single

administrative domain.

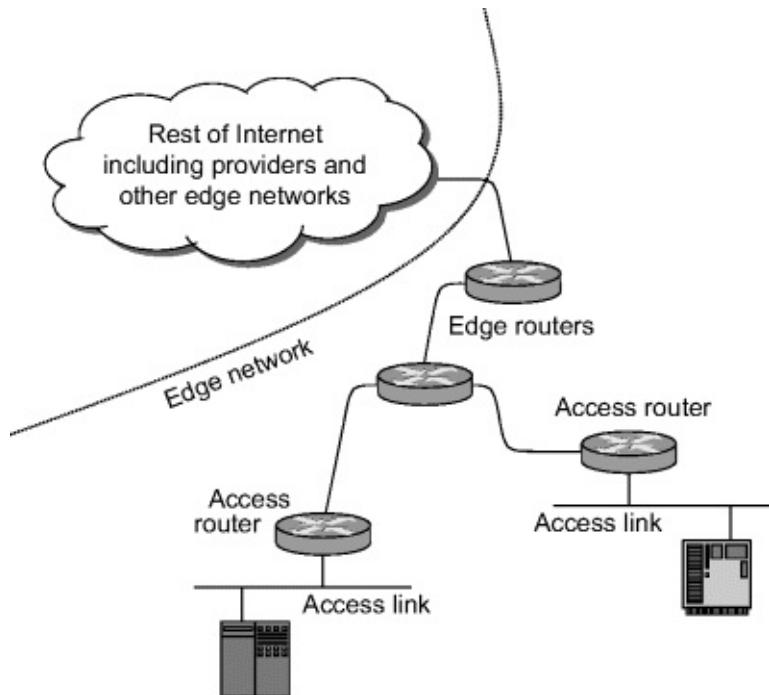
- *Topology resolution:* The global scope of an external routing protocol implies higher scalability requirements for external routing protocols compared with internal routing protocols. The extra scalability often comes at the cost of a coarser resolution of the explored Internet topology. BGP, for example, supplies only the direction in which a certain destination can be reached. It supplies neither the sequence of provider networks through which the destination is reached, nor the sequence of routers traversed within any given provider network. OSPF, on the other hand, can afford to supply the sequence of traversed routers for a given destination, because it operates within the smaller scope of a single network.

It is these three differences that suggest the use of different protocols within and across provider networks.

1.3.2 Routing within Edge Networks

[Figure 1.22](#) illustrates the typical topology of an edge network. Some of the links in an edge network are *access links* to which nodes can directly attach. The access link for a stationary node may be a single physical wire, such as an Ethernet cable or a Digital Subscriber Line (DSL), or an *access point* that connects a wireless access technology to the fixed network. An access link may include more than one access point for extended geographic coverage. The first edge router for packets that the node sends to destinations off-link is an *access router*. Edge networks communicate with each other through providers.

[Figure 1.22](#) Edge network topology.



The preferred addressing and routing in edge networks differs from the addressing and routing across providers. Edge networks are called upon to use a portion of the IP address prefix allocated to their respective providers. The left edge network in [Figure 1.21](#) thus gets a portion of provider 1's IP address prefix, say, 110.123.0.0/16. And the right edge network gets a portion of provider 3's IP address prefix, say, 130.56.0.0/16. Accordingly, a host in the left edge network will configure an IP address that belongs to provider 1, and a host in the right edge network will configure an IP address that belongs to provider 3. As shown in [Figure 1.21](#), these IP addresses are 110.123.229.70 and 130.56.1.33, respectively. Similar to routers within a provider, routers within an edge network use an internal routing protocol, typically OSPF, to jointly explore their interconnectivity.

The reason for reusing providers' IP address prefixes in edge networks, instead of giving edge networks provider-independent IP address prefixes, is scalability. Edge networks become implicitly reachable via their respective provider's entry in the global routing table. They do not need their own routing table entry. In specialist terms, IP addresses used inside an edge network are called *aggregatable* with the address space of the edge network's provider. Thus, in the example of [Figure 1.21](#), packets destined to the host shown on the lower left find the destination edge network by following the routing table entries for provider 1, and provider 1 forwards the packets rightward.

1.3.3 Routing Scalability

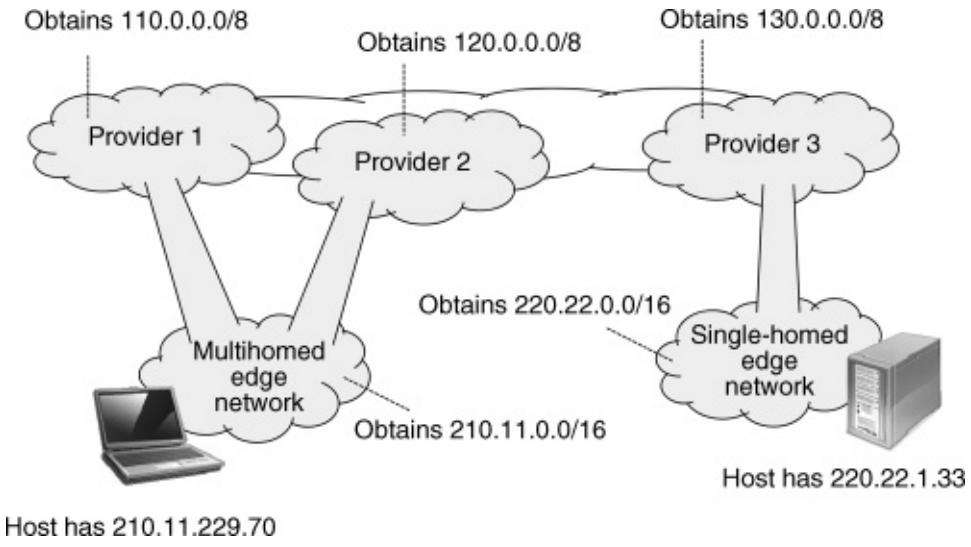
Although the reuse of providers' IP address prefixes in edge networks aids the scalability of the global routing system, edge network operators frequently demand their own provider-independent IP address prefix. The reason is that the reuse of providers' IP address prefixes limits the flexibility of edge networks in two ways:

- Reuse of providers' IP address prefixes implies that, when an edge network becomes multihomed, it will obtain IP address prefixes from each of its providers, and hosts will configure IP addresses from each such prefix. Packets that are sent from or destined to either of these IP addresses will then be routed via the provider to which the IP address belongs. Rerouting for the purpose of failover or load balancing is then impossible.
- Reuse of providers' IP address prefixes causes a slight form of provider lock-in. It requires edge networks to change their IP address prefix whenever they change providers, and this requires them to undergo renumbering. *Renumbering* is an expensive and time-consuming procedure that requires substantial manual efforts. It implies IP address changes to hosts and network equipment edge-network-wide. It affects routers, host, Domain Name System (DNS) and Dynamic Host Configuration Protocol (DHCP) servers, firewalls, intrusion detection systems, remote-monitoring systems, load balancers, as well as scripts and configuration files.

Consequently, it is no surprise that edge network operators are reluctant to reuse providers' IP address prefixes. They prefer to have individual IP address prefixes like providers.

To enable multihoming and eliminate renumbering, the current practice is to assign edge networks provider-independent IP address prefixes. This is illustrated in [Figure 1.23](#) for the multihomed edge network at the lower left. Accordingly, the IP addresses of hosts in the edge network are no longer bound to provider 1 or to provider 2. Packets sent from such an IP address can be routed to either provider.

[**Figure 1.23**](#) Provider-independent addressing.



Host has 210.11.229.70

On the other hand, provider-independent addressing has an adverse impact on the scalability of the Internet routing system. In order for packets to find their way toward a provider-independent IP address, the IP address prefix of the edge network needs to be advertised among providers via BGP. This implies higher router load and, thus, less efficient packet forwarding.

- *Increased routing table size:* Global routing tables must list the provider-independent IP address prefixes of edge networks separately because these IP address prefixes can no longer be aggregated with an IP address prefix of the provider. The result is an increase in the size of global routing tables. This effect can be substantial because the number of potentially multihomed edge networks is orders of magnitude larger than the number of providers.
- *More frequent routing table updates:* For an edge network to change the provider based on which packets will reach it, IP address prefix announcements in BGP must be updated. The result is an increase in the update frequency of the global routing table.

Even though a scalable Internet routing system is in the interest of edge network operators and providers, the selfish yet legitimate interest of each individual network operator is putting the scalability of the Internet routing system under pressure. This problem is not new. In fact, it has been a concern for almost the entire life of the Internet. Nonetheless, there has never been a major change in the Internet routing architecture. Addressing has always been done according to the Internet protocol, where version 6 of the protocol does not differ conceptually from version 4. And routing has not changed

since the introduction of BGP in the 1980s. There was only one evolutionary step, Classless Inter-Domain Routing (CIDR) which is a means to improve the efficiency of IP address allocation and aggregation.

Mitigating the scalability problem of the Internet routing architecture is important to enable continued efficient functioning of the Internet and reasonable upgrade intervals for routers. A scalability problem always becomes more and more significant as the affected system grows. The Internet routing scalability problem may have been acceptable in the early days of the Internet, when the Internet was still small. It may even still be acceptable today. But the problem is becoming more and more noticeable, and it will continue in this direction.

For a few years now, therefore, people are more seriously considering improvements to the Internet routing system—not only in the engineering community, but also and foremost in the research community. In October 2006, the Internet Architecture Board (IAB) had a meeting about the routing scalability problem. The problem was brought up foremost by router vendors, who had decided to request that the engineering community do something about the problem. The IAB decided to further investigate the severity of the problem and possible solutions. It chartered a routing research group inside the Internet Research Task Force (IRTF) for these efforts. In 2010, the routing research group released a recommendation for the Internet Engineering Task Force (IETF) on possible solutions to the routing scalability problem [4]. The recommendation considers a diverse set of proposed solutions, ranging from backwards-compatible evolutionary techniques to revolutionary clean-slate approaches. Some solutions tackle the problem with more scalable router architectures, others advocate changes to IP addressing schemes and routing protocols.

References

1. Rekhter, Y. and Li, T., “A Border Gateway Protocol 4 (BGP-4),” IETF RFC 1771, Mar. 1995.
2. Moy, J., “OSPF version 2,” IETF RFC 2328, Apr. 1998.
3. Coltun, R., Ferguson, D., Moy, J. and Lindem, A., “Open Shortest Path First for IPv6,” IETF RFC 5340, Jul. 2008.
4. Li, T. (Editor), “Recommendation for a Routing Architecture,” IRTF RFC

6115, Feb. 2011.

Chapter 2

Transport-Layer Protocols

This chapter explains protocols in the transport layer, which is the fourth layer of the Open Systems Interconnection (OSI) reference model. Transparent transfer of data between end users using services of the network layer is provided. The well-known protocols in this layer are Transmission Control Protocol (TCP), User Datagram Protocol (UDP), Stream Control Transmission Protocol (SCTP), and Realtime Transport Protocol (RTP).

2.1 Transmission Control Protocol

Transmission Control Protocol is a connection-oriented, end-to-end, reliable protocol in the transport layer [1]. Creation of connection is needed before sending the data. TCP has the advantage of transporting data reliably. Lost packets are resent by a TCP retransmission mechanism. There are many protocols that rely on TCP, such as FTP, HTTP, POP3, and SMTP. However, TCP is not suitable when transmission speed is more important than reliability, such as with telephone or video conferencing over the Internet.

2.1.1 Transmission Control Protocol Header Structure

[Figure 2.1](#) shows the TCP header structure.

- *Source port* and *destination port* identify the application service of both sender and receiver.
- *Sequence number* specifies a number from the first data byte in a particular segment. If the Synchronize Sequence Numbers (SYN) flag is set, the sequence number is the initial sequence number and the first data byte is the initial sequence number + 1.
- *The acknowledgment number* has the same function as the sequence number. However, it is used to respond to a sender. If the ACK bit is set, this field contains the value of the next sequence number that

the sender of the segment is expecting to receive. Once a connection is established this is always sent.

- *Data offset* indicates the starting bit of the data. Usually, the header length is 20 bytes. If there is some data in the option field, the header length might be extended up to 60 bytes.
- *Reserve* indicates reserved for future use.
- *Control bits* distinguish session management messages from data.
 - URG: Urgent pointer valid flag.
 - ACK: Acknowledgment number valid flag.
 - PSH: Push flag.
 - RST: Reset connection flag.
 - SYN: Synchronize sequence numbers flag.
 - FIN: End of data flag.
- *Window size* indicates the number of data bytes, beginning with the one indicated in the acknowledgment field, that the sender of this segment is willing to accept.
- *Checksum* is used to verify that the end-to-end data has not been corrupted.
- *Urgent pointer* indicates the sequence number of the last byte in a sequence of urgent data. This is applicable when the URG bit is set.

Figure 2.1 TCP header structure.

0	4	10	16	31
Source port		Destination port		
Sequence number				
Acknowledgment number				
Data offset	Reserve	Control bits	Window size	
Checksum		Urgent pointer		
Option				

2.1.2 Three-Way Handshake

Three-way handshaking is a process to establish a connection between two hosts—client and server. The process is as follows:

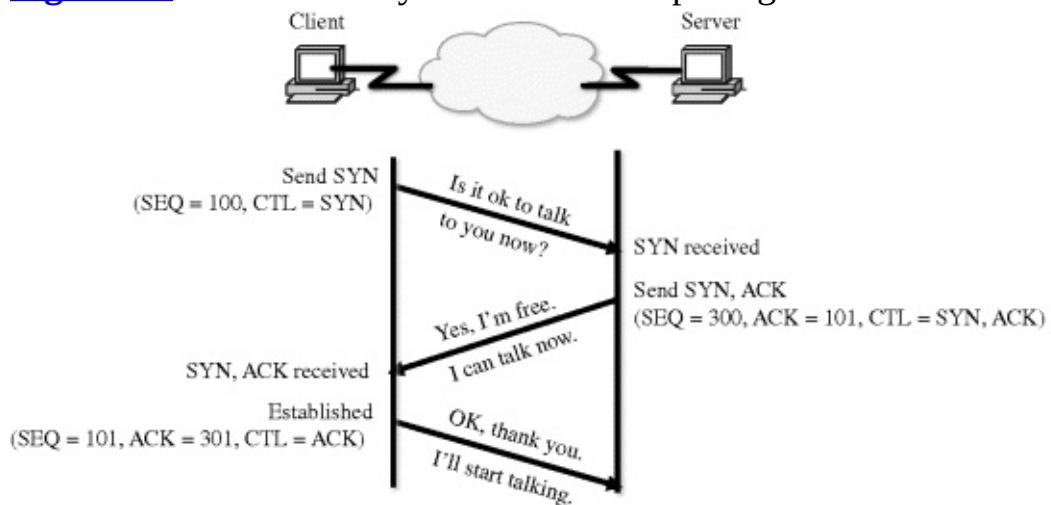
- The client sends a packet with sequence number X to the server. Only the SYN flag is set.

- The server receives a packet with sequence number X . It replies acknowledgement number $X + 1$ and a new sequence number Y . The SYN and ACK flags are set.
- The client receives a packet with sequence number Y and acknowledgement number $X + 1$. The client sends sequence number $X + 1$ and acknowledgement number $Y + 1$ back to the server. Only the ACK flag is set.

After the above three steps have taken place, the connection is established and the client starts sending data.

[Figure 2.2](#) shows an example of a three-way handshake. First, the client asks the server, “Is it ok to talk to you now?” by setting the SYN flag and sending the sequence number (SEQ) = 100 to the server. After the server receives the request message, it replies “Yes, I’m free. I can talk now.” to the client by sending SEQ = 300 and acknowledgement number (ACK) = 101 with the SYN and ACK flags set. The client receives the message and says “OK, thank you. I’ll start talking.” to the server by sending SEQ = 101 and ACK = 301 with the ACK flag set. The connection is ready.

[Figure 2.2](#) TCP three-way handshake for opening connection.



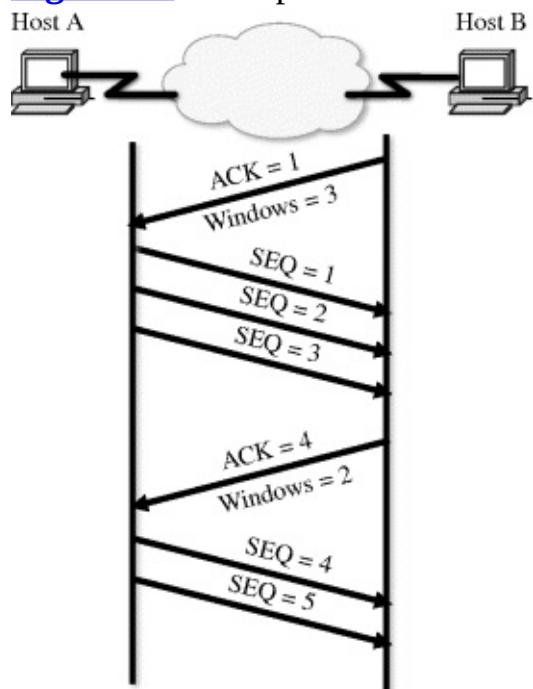
2.1.3 Transmission Control Protocol Flow Control and Congestion Control

TCP has flow and congestion control mechanisms. Flow control enables the source host to control data that is sent to

the destination host so that the destination host can receive the data correctly. Congestion control helps the source host control the data rate to avoid network congestion.

TCP uses a sliding window mechanism to control the flow of data. After the connection is established, each end host holds the incoming data in a buffer. The end host sends acknowledgment indicating the available size of buffer space, a window advertisement, to the opposite end host. This allows the communicating hosts to negotiate how much data is transmitted each time. [Figure 2.3](#) shows an example of flow control using a sliding window. Host B sends ACK and window size = 3 to host A. This means that host B can receive three SEQ from host A. Host A send SEQ = 1, 2, and 3 to host B. Next, host B sends window size = 2 to host A. Host A sends SEQ = 4 and 5 to host B.

[Figure 2.3](#) Example of flow control using sliding window.



The TCP congestion control mechanism controls the window size to avoid network congestion. If a network is not congested, a host is allowed to increase the transmission rate. Otherwise, it decreases the rate. [Figure 2.4](#) shows performance comparisons of short and long round-trip delays with

different window sizes. For example, consider short and long delays with a window size of 3. With a window size of 3, a maximum of three outstanding packets whose acknowledgements are returned back to the source host are allowed. In the case of a long delay, it takes more time to send packets because the source host has to wait longer to receive the acknowledgements. As a result, throughput is smaller than in the case of a short delay. Now consider cases with the same delay but with the different window sizes of 3 and 8. Increasing the window size allows more packets to be transmitted without receiving acknowledgements. If the network is not congested, throughput increases with the window size. However, in case the network is congested, once one packet is lost, the lost packet and all the packets behind the lost packet need to be retransmitted, as shown in [Figure 2.5](#). An inappropriate large window size may cause more network congestion in a recursive manner, while a small window size limits throughput. Therefore, it is crucial to appropriately control window size.

Figure 2.4 Comparison of short and long round-trip delays with different window sizes.

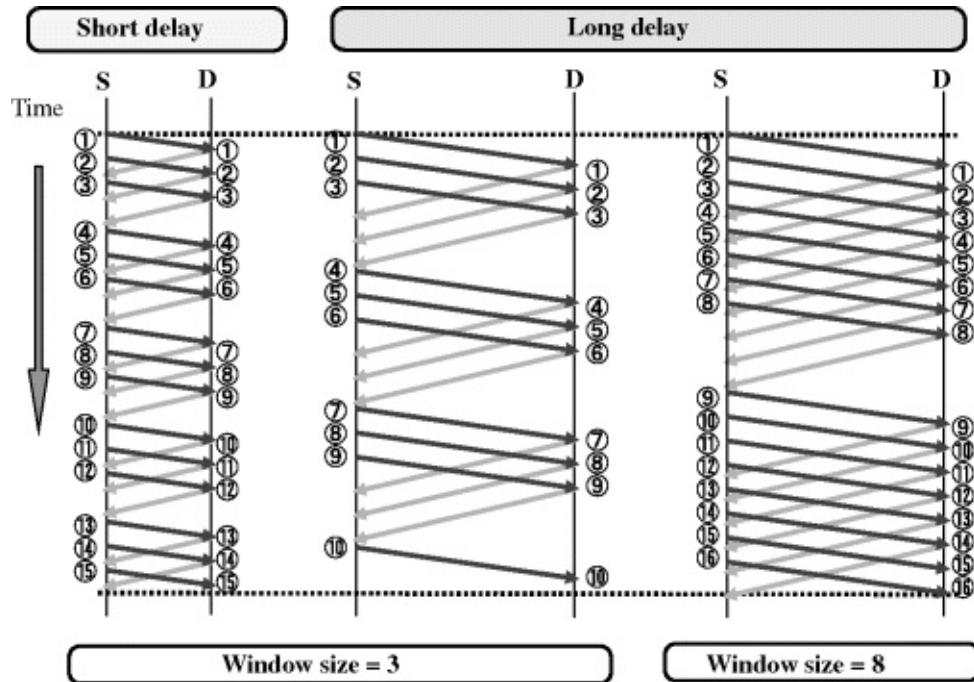
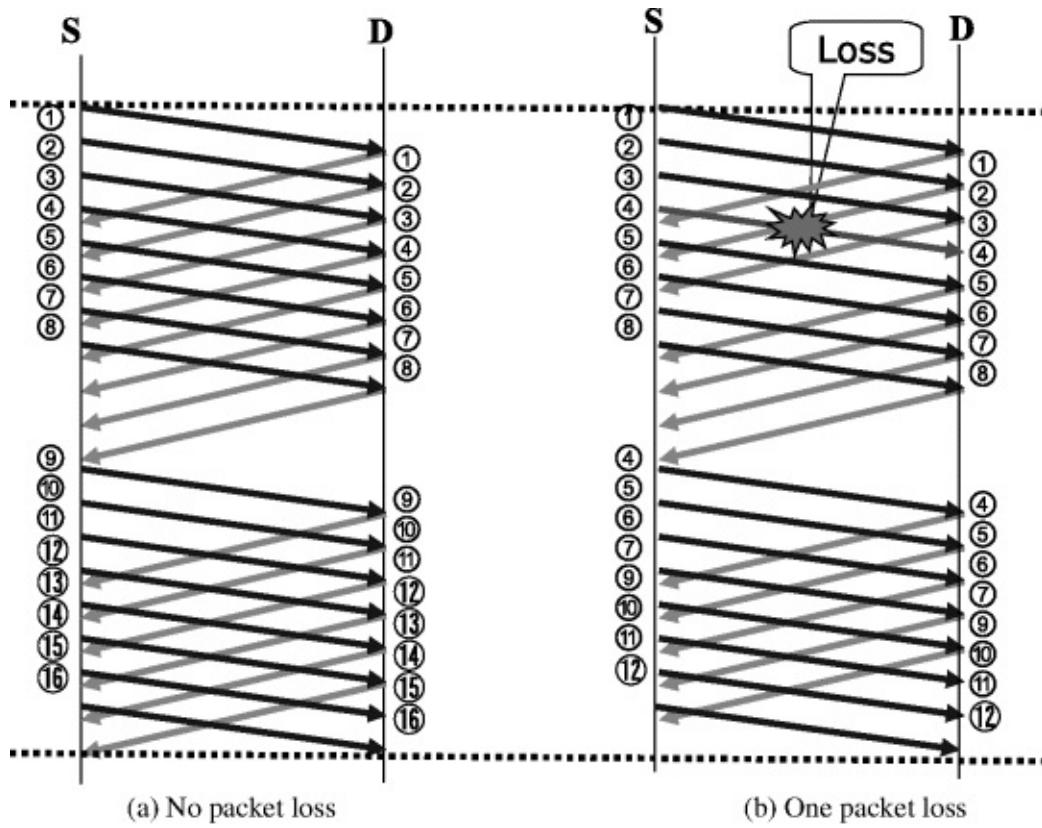


Figure 2.5 Flow control with and without packet loss.

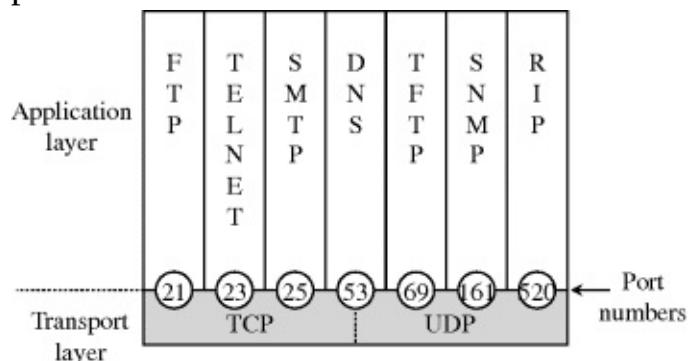


To control the window size in a distributed manner, a slow-start congestion control algorithm is a widely used algorithm to avoid network congestion [2]. The slow-start congestion control algorithm consists of two phases: the exponential growth phase and the linear growth phase. At the beginning, the window size is set to 1. Whenever an acknowledgment is received without any packet loss, the window size doubles until it reaches a specified threshold. After the window size exceeds the specified threshold, it increases linearly. If any packet loss is detected while the window size is increased or a corresponding acknowledgement to the sent segment is not received, the source host recognizes that the network is congested. Then, the source host decreases the window size to 1 and the slow-start congestion control is restarted. To recover the lost packet quickly, retransmissions are performed right after network congestion is detected. This algorithm is called TCP Tahoe. Another algorithm used to decrease window size when network congestion is detected is TCP Reno. In TCP Reno, if the retransmissions succeed, the source host recognizes that the network is not congested too heavily. The source host decreases the window size by half when network congestion is detected, and the slow-start congestion control is restarted. Otherwise, the window size is decreased to 1 as in TCP Tahoe.

2.1.4 Port Number

A port number is part of the addressing information used to identify the senders and receivers of messages. The port number allows different applications on the same computer to share network resources simultaneously. It is used in both TCP and UDP [3], which are explained in the following sections. Port numbers work like telephone extension numbers. A computer makes a request to an IP address (the phone number) and a specific port (the extension). The port number tells the server on the receiving end which application this request belongs to. Some examples of applications include Web sites, chat, e-mail, and telnet. The port numbers are between 0 and 65535. [Figure 2.6](#) shows the connection between application layer and transport layer using port numbers to specify an application. Well-known ports are shown in [Table 2.1](#).

[Figure 2.6](#) Connection between application layer and transport layer with port numbers.



[Table 2.1](#) Lists of Well-Known Ports.

Port No.	Protocol	Application Name
20 and 21	TCP	File Transfer Protocol (FTP)
22	TCP	Secure Shell (SSH)
23	TCP	Telnet
25	TCP	Simple Mail Transfer Protocol (SMTP)
53	TCP and UDP	Domain Name System (DNS)
69	UDP	Trivial File Transfer Protocol (tftp)
80	TCP	Hypertext Transfer Protocol (HTTP)
110	TCP	Post Office Protocol v3 (POP3)
119	TCP	Network News Protocol (NNTP)
161 and 162	UDP	Simple Network Management Protocol (SNMP)
443	TCP	Secure Sockets Layer over HTTP (https)

2.2 User Datagram Protocol

User Datagram Protocol (UDP) is a communications protocol that offers a limited amount of service when messages are exchanged between computers in a network that uses the IP [3]. With UDP there is no handshaking between sending and receiving transport-layer entities before sending a segment. For this reason, the UDP is said to be connectionless. UDP is used in connectionless-based applications such as Simple Network Management Protocol (SNMP), Routing Information Protocol (RIP), and Trivial File Transfer Protocol (TFTP).

2.2.1 User Datagram Protocol Header Structure

[Figure 2.7](#) shows the header format of the UDP packet from a client, or source, to a server, or destination. The header format is more simple than that of TCP.

[Figure 2.7](#) UDP header structure.

0	16	31
Source port	Destination port	
Length	Checksum	
Data		

- *Source port* is an optional field to indicate the port of sending process.
- *Destination port* indicates the service required at the remote server.
- *Length* indicates the number of bytes comprising the UDP header information and payload data.
- *Checksum* is used to verify that the end-to-end data has not been corrupted.

Since there is no acknowledgment to confirm that the data has been received, the client does not know that the data arrived at the server. However, the transmission is faster than TCP, which requires a three-way handshake to setup a connection. Furthermore, since the connection setup is not required in UDP, there is less data flow in the network. The disadvantage of UDP is that the arrival of data at the destination is not guaranteed.

2.3 Stream Control Transmission Protocol

Stream Control Transmission Protocol (SCTP) is a protocol in the transport layer [4]. It was originally designed for telephony signaling over IP networks to solve limitations of TCP. The other applications may benefit, such as multimedia Web browsing. In SCTP, all of the features in TCP including new features, multistreaming and multihoming, are provided. Multistreaming supports independent transport and delivery of multiple streams between hosts. Multihoming supports more than one path between hosts for resilience. Moreover, protection against SYN flooding attacks and multistreaming is improved. If you have an opportunity to use Voice over Internet Protocol (VoIP) telephone or other VoIP devices, you may be asked to set Session Initiation Protocol (SIP). The SIP is an application that uses SCTP.

2.3.1 Stream Control Transmission Protocol Packet Structure

[Figure 2.8](#) shows the SCTP packet structure, which consists of two parts: common header and chunks. The common header is associated with the following functions.

- *Source port* and *destination port* indicate the sending port at the source and receiving port at the destination.
- The *verification tag* is used to verify that the packet comes from the correct sender. The value of this field is always set to the same value as the value received by the other peer in the initiation tag during the initialization state.
- *Checksum* is used to confirm that the data has not been corrupted.

[Figure 2.8](#) SCTP packet structure.

Source port		Destination port
Verification flag		
Checksum		
Chunk 1 type	Chunk 1 flag	Chunk 1 length
Chunk 1 data		
...		
Chunk N type	Chunk N flag	Chunk N length
Chunk N data		

The following functions are associated with chunks.

- *Chunk type* indicates the content of the chunk value field. Some values of this field are defined in [Table 2.2](#) [4].

Table 2.2 Examples of Chunk Types.

Value	Type
0	Payload Data (DATA)
1	Initiation (INIT)
2	Initiation Acknowledgment (INIT-ACK)
7	Shutdown (SHUTDOWN)
8	Shutdown Acknowledgment (SHUTDOWN-ACK)
10	State Cookie (COOKIE)
11	Cookie Acknowledgment (COOKIE-ACK)

- *Chunk flag* depends on the chunk type. The default value is zero.
- *Chunk length* represents the size of the current chunk in bytes including chunk type, chunk flag, chunk length, and chunk data.
- *Chunk value* contains the actual information. The usage format depends on the chunk type [4].

2.3.2 Security: Prevention of SYN Attacks

SCTP was designed with a security advantages. A malicious user can flood several SYNs to the server. In this case, the memory resources of the server might be filled with fake SYNs. The server then denies service to the actual SYNs. [Figure 2.9](#) shows an example of the SYN flooding attack in a three-way handshake. A malicious user sends several SYN messages to the server.

After the server receives a SYN message, it replies with a SYN-ACK message that corresponds to the received SYN message to the user. The server waits for ACK messages from the user. We call this situation a half-open connection. Usually, the user would send ACK messages back to the server. However, the malicious user does not do it. The server keeps waiting for the ACK messages for a certain amount of time. The memory of the server might overflow, and other requests by normal users might be rejected. To overcome this problem, a four-way handshake, as shown in [Figure 2.10](#), is used to establish the connection to prevent SYN-flooding attacks. The processes of the four-way handshake are as follows:

- The client sends an INIT packet to the server.
- The server replies with an INIT-ACK packet to the client including a verification tag and a cookie. The cookie contains necessary details to identify and process the association including a signature for authenticity and a timestamp to prevent replay attacks using old cookies.
- When the client receives the INIT-ACK from the server, it sends a COOKIE-ECHO packet to the server that echoes the cookie received from the server. If the client uses a fake IP address, it never receives the INIT-ACK. This can prevent the client from sending the COOKIE-ECHO packet to the server.
- The server then acknowledges the COOKIE-ECHO received from the client with a COOKIE-ACK packet.

[Figure 2.9](#) SYN flooding attacks.

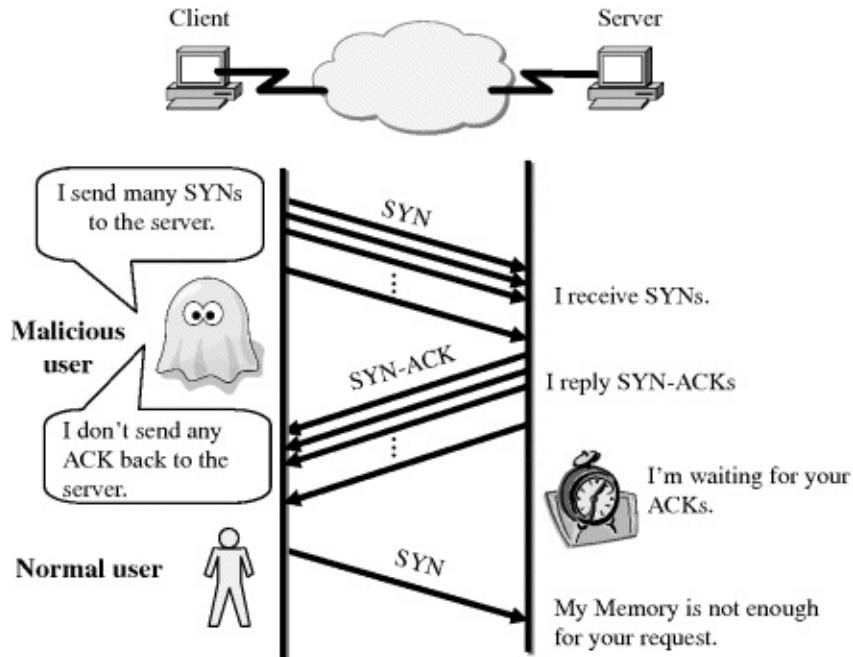
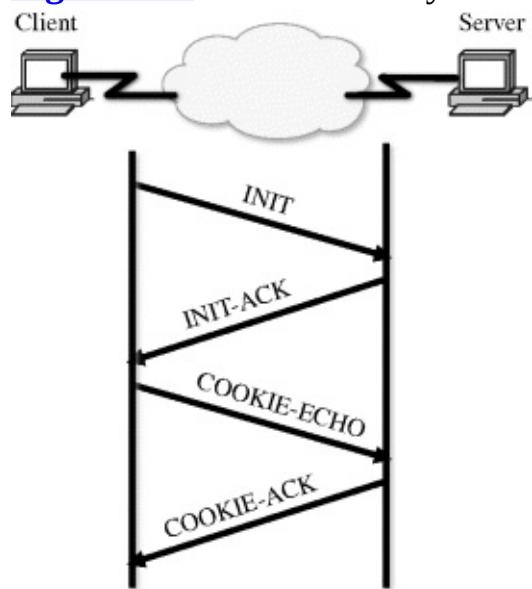


Figure 2.10 SCTP four-way handshake.



After the process of the four-way handshake is finished, the connection is established and the data is ready to be sent from the client to the server.

2.4 RealTime Transport Protocol

Realtime Transport Protocol (RTP) provides end-to-end network delivery services for the transmission of realtime data [5]. This protocol is used for applications with a realtime requirement, such as VoIP, streaming television,

and video conferencing. RTP does not establish the connection. Therefore, packets are not guaranteed to arrive in the order that they were sent. The order depends on the ability of the receiver to reconstruct the packet sequence and detect lost packets using the information provided in the packet header. The mechanism is used in conjunction with RTP control protocol (RTCP) to monitor the quality of the data distribution. It also provides control and identification mechanisms for RTP transmissions.

2.4.1 RealTime Transport Protocol Header Structure

[Figure 2.11](#) shows the structure of the RTP packet header. The details of each field are explained as follows:

- *Ver* is the RTP version number. The current version is 2.
- *P (Padding)* refers to the padding bit. If this bit is set, there is one or more bytes at the end of the packet that are not part of the payload. The last byte in the packet indicates the number of bytes of padding. The padding is used by some encryption algorithms.
- If the *X (extension)* bit is set, the fixed header is followed by one header extension. This extension mechanism enables adding information to the RTP header.
- *CC (CSRC count)* indicates the number of CSRC identifiers that follow the fixed header.
- *M (marker)* is defined by the particular media profile.
- *PT (payload type)* identifies the media profile of the RTP payload and determines its interpretation by the application.
- *Sequence number* is a unique packet number that identifies the packet's position in the sequence of packets. The packet number increased by one for each packet sent.
- *Timestamp* is used by the receiver for synchronization and jitter calculations.
- *SSRC (synchronization source)* is used to identify the source of realtime stream. The receiver uses this field to group packets with the same SSRC for playback.
- *CSRC (contributing source)* identifies the contributing sources for the payload. The number of contributing sources is indicated by the

CSRC count field.

Figure 2.11 RTP header structure.

0-1	2	3	4-7	8	9-15	16-31
Ver	P	X	CC	M	PT	Sequence number
Timestamp						
SSRC identifier						
CSRC identifier						

References

1. “Transmission Control Protocol Darpa Internet Program Protocol Specification,” IETF RFC 793, Sep. 1981.
2. Stevens, W., “TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms,” IETF RFC 2001, Jan. 1997.
3. Postel, J., “User Datagram Protocol,” IETF RFC 768, Aug. 1980.
4. Morneau, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L. and Paxson, V., “Stream Control Transmission Protocol”, IETF RFC 2960, Oct. 2000.
5. Schulzrinne, H., Casner, S., Frederick, R. and Jacobson, V., “RTP: A Transport Protocol for RealTime Applications,” IETF RFC 3550, Jul. 2003.

Chapter 3

Internet Architecture

One of the key factors contributing to the success of the Internet is an architecture that is well suited to rapid and substantial changes in usage. The Internet originally descended from the “Arpanet” in the late 1960s, a network connecting universities and government institutions for research purposes and enabling remote sites to quickly access information from each other and execute programs on distant computers. The World Wide Web commercialized the Internet in the 1990s, making it available beyond the academic and government communities. More recently, the higher speeds of third-generation cellular networks in conjunction with advances in smart phone user interfaces are enabling the widespread use of the Internet on mobile and portable devices. Today, the Internet is the most important communications medium other than the telephone network, serving nearly 2 billion users [1].

Despite this staggering evolution, the Internet still retains most of its early architectural properties. This chapter describes Internet architecture, including basic Internet topology, Internet exchange points (IXPs), the history of IXPs', and the principles of Internet relationships and Internet service providers (ISPs).

3.1 Internet Exchange Point

The Internet is a collection of *routers*, interconnected by *links*, that forward data from a source to a destination. Computers, or *nodes*, exchange data across the Internet by splitting the data into portions, or so-called *packets*, that are individually forwarded by routers. The process of forwarding a packet across the Internet from the source node to the destination node is called *routing*. Routing proceeds in a hop-by-hop manner, where each node on the packet's path, from the source node via routers to the destination node, independently determines a neighbor closer to the destination node and hands the packet on to that neighbor. The routing process completes when that

neighbor happens to be the destination node itself.

An Internet exchange point (IXP) is the location where two or more Internet service provider (ISP) networks meet [2]. It consists of ISP-owned and-managed routers interconnected through gigabit or optical Layer-2 switches and/or Layer-3 routers. Within an IXP, ISPs advertise their routes to other ISPs, and data packets flow from one ISP to another [3].

The service value that an ISP could offer to its customers by connecting to an IXP is increased by the number of additional networks to which customers gain access.

In addition, the IXP is also a logical point to offer value-added services such as Web hosting, caching, and content distribution. This makes the IXP as important to the function of the Internet as the ISPs' internal networks themselves.

An IXP could be private or public as shown in [Figures 3.1](#) and [3.2](#). A private IXP provides a direct peer-to-peer interconnection between two ISPs. A public IXP is managed by a third party and enables any ISP to connect with any other ISP. The IXP provides two types of connections between ISPs: bilateral peering arrangement (BLPA) and transit relationship. In a peering arrangement two ISPs agree to provide access to their customer routes for free. In a transit relationship, a smaller ISP pays a transit fee to a larger ISP for access to the wider Internet. Public IXPs ([Figure 3.2](#)) do not permit transit relationships between two ISPs, as they are meant to be neutral locations for peering. Transit relationships are supposed to be provider-to-consumer relationships and are against the spirit of a public IXP. However, public IXPs do provide a backdoor mechanism for transit relationships through a serial cable that completely bypasses the public IXP switch fabric as shown in [Figure 3.2](#).

Figure 3.1 Private Internet exchange points.

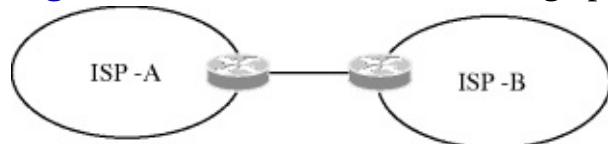
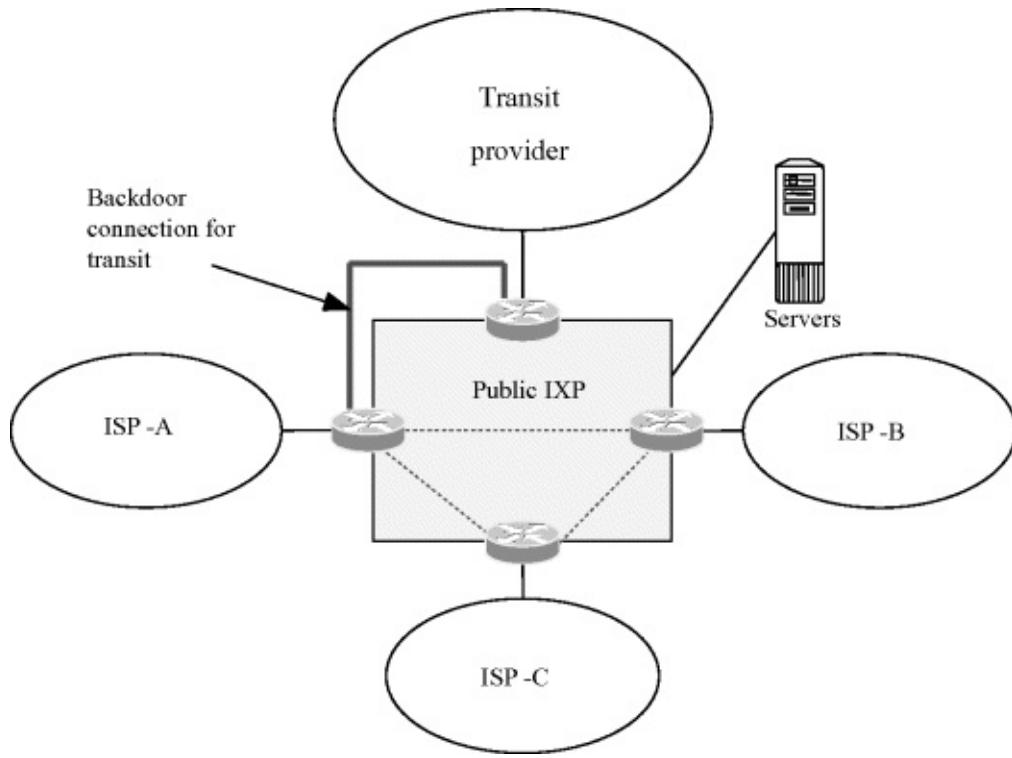


Figure 3.2 Public Internet exchange points.



The common mechanisms used to interconnect ISPs are based on Layer-2 and point-to-point data link technologies as shown in [Figures 3.1](#) and [3.2](#). The IXP can also provide value-added services such as Domain Name Systems (DNS), Network Time Protocol (NTP) servers, and mail servers that IXP members could use. In order to promote network stability and scalability the IXP provides a centralized database of routes and policies belonging to the ISPs connected to it. Routing information is distributed between ISP routers attached to the IXP through a route server attached to the IXP. The route server simplifies the IXPs configuration and improves its scaling capabilities. It reduces the number of control channels and the related configurations required for ISP routers to exchange routes from $O(n^2)$ to $O(n)$. This is achieved by allowing ISP routers to exchange routes through the route server instead of directly among themselves, which would require a full mesh. The data traffic flows directly between ISPs on a separate data channel that is different from the control channel.

3.2 History of Internet Exchange Points

In the early 1980s, the National Science Foundation established NSFnet, which took over the job of providing the first real Internet backbone from the

Arpanet. During that time, the individual government, research, and academic networks connected to each other through an early IXP setup by the government called the Federal Internet Exchange (FIX). Before this, these individual networks connected to each other directly. As an alternative to the government-sponsored FIX that excluded commercial participation, a Commercial Internet Exchange (CIX) was created in 1991 by three ISPs: Alternet, PSInet, and Cerfnet [4]. The CIX was the first full-fledged commercial IXP that provided connectivity among all its members and to the wider Internet through exchange of routes and traffic. The success of CIX proved that the value an ISP provided to its customers increased manyfold by connecting to a number of other networks through an IXP [5].

As the Internet buildup was gaining great momentum through the advent of e-commerce, it became evident that the government-sponsored Internet backbone would not scale and that commercial Internet providers would play a major role in providing the required backbone networks for the Internet. Thus, the NSFnet was gradually shutdown, and the NSF started awarding the task of building Internet exchange points to commercial operators. The initial NSF-awarded IXPs were, and continue to be, called network access points (NAPs). The first four NAPs awarded by the NSF are as follows:

- PacBell NAP in San Francisco, California
- Ameritech Advanced Data Services (AADS) NAP in Chicago, Illinois
- MFS Datanet (MAE-East) in Washington,
- Sprint NAP in Pennsauken, New Jersey

Later, the NSF originated the Routing Arbiter project designed to enhance the NAP architecture and scalability through the introduction of the NAP-attached route server.

3.3 Internet Service Provider Interconnection Relationships

The Internet consists of a large collection of hosts interconnected by networks of links and routers. The Internet is divided into thousands of distinct regions of administrative domain, each of which possesses one or several Autonomous Systems (ASs) [6].

The commercial agreements between ASs can be categorized into customer-provider, peering, mutual-transit, and mutual-backup. A provider provides Internet connectivity to its customers. A pair of peers exchange traffic between their respective customers. Mutual-transit agreements allow administrative domains (normally small ISPs close to each other) to connect to the Internet through each other. A customer pays its provider for connectivity to the rest of the Internet. Therefore, a provider does transit traffic for its customers. However, a customer does not transit traffic between two of its providers. A pair of peers agree to exchange traffic between their respective customers free of charge. A mutual-transit agreement allows a pair of administrative domains to provide connectivity to the rest of the Internet for each other. The mutual-transit relationship is usually between two administrative domains such as small ISPs who are located close to each other and who cannot afford additional Internet services for better connectivity. In case the connection to its provider fails, an AS may get connectivity to the Internet via another AS through mutual-backup agreement. A pair of administrative domains may also provide backup connectivity to the Internet for each other in the event that one administrative domain's connection to its provider fails. These contractual commercial agreements between administrative domains are crucial in shaping the structure of the Internet and the end-to-end performance characteristics. The commercial agreements among ASs play an important role in deciding how traffic flows in the Internet and are thus critical to understanding the structure of the Internet and end-to-end performance. However, such information is not publicly available due to privacy concerns and commercial reasons.

BGP is a path-vector protocol that constructs paths by successively propagating updates between pairs of BGP-speaking routers that establish BGP peering sessions [7]. All such attributes can influence the route selection decision. Some of the attributes, such as ORIGIN, AS_PATH, and NEXT_HOP, are mandatory. By representing the path at the AS level, BGP hides the details of the topology and routing inside each network. BGP is *incremental*, meaning that every BGP update message indicates a routing change. In addition, BGP is *policy oriented*. Rather than selecting the route with the shortest AS path, routers can apply complex policies to influence the selection of the *best* route for each prefix and to decide whether to propagate this route to its neighbors.

In general, routing policies include import and export policies [8]. Import

policies specify whether to deny or permit a received route and assign a local preference indicating how favorable the route is. Export policies allow ASs to determine whether to propagate their best routes to neighbors. An AS uses import policies to transform incoming route updates. These import policies include denying an update, or permitting an update and assigning a local preference to indicate how favorable the path is. After applying the import policies for route updates from a BGP session, an AS saves all the imported updates in its BGP routing table. The AS then follows a route selection process that picks the best route for each prefix. Each AS sends only its best route for a prefix to a neighbor. Export policies allow an AS to determine whether to send its best route to a neighbor and, if it does send the route, what hint it should send to its neighbor for using the route.

3.4 Peering and Transit

Commercial Internet relationships can be categorized into peering relationships and transit relationships. [Figure 3.3](#) demonstrates the routing table exchanged between two peering routers. In this scenario, ISPs A and B exchange their customers' routing information. The peering links will only be used for carrying customer traffic. The transit relationship is shown in [Figure 3.4](#). The customer ISP-C announces routes to its provider and then to the whole Internet. At the same time, the provider provides ISP-C routes for reaching the remainder of the Internet. The traffic from the customer is carried by the larger ISP to the Internet [9].

[Figure 3.3](#) Illustration of peering relationship.

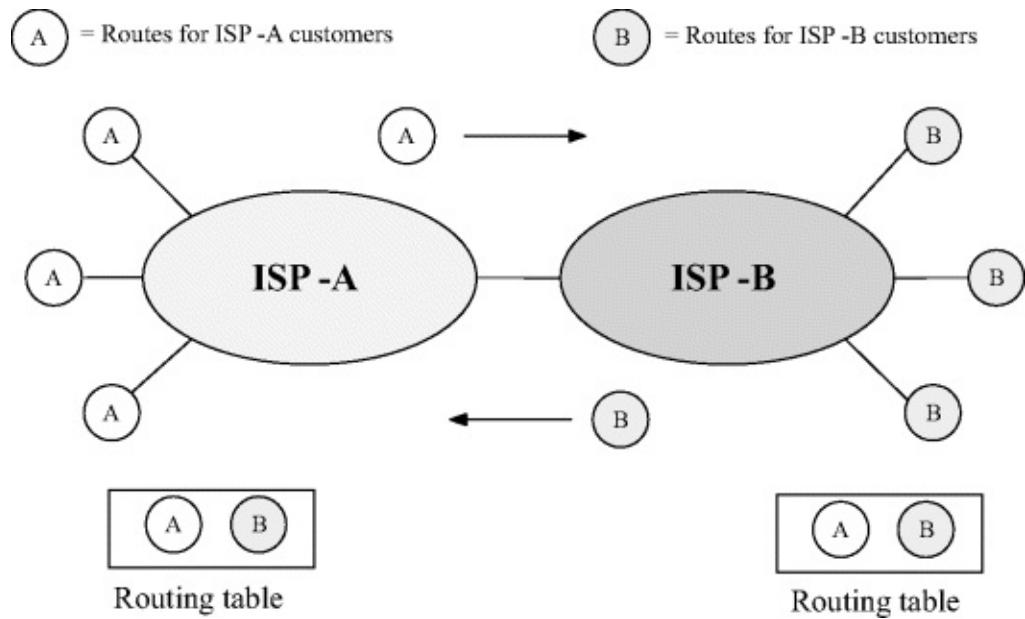
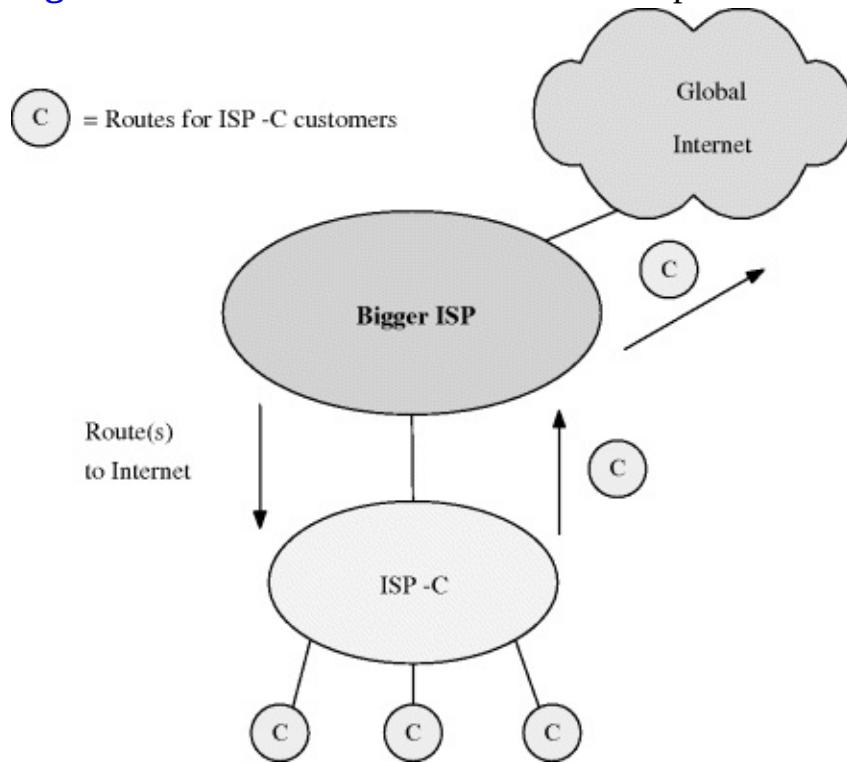


Figure 3.4 Illustration of transit relationship.



These exporting rules indicate that an AS path should be valley free, meaning that after a provider-to-customer edge or a peer-to-peer edge the AS path cannot traverse customer-to-provider edges or another peer-to-peer edge. More specifically, valid routing policies permit AS paths in the form of *customer-provider**, *peer-peer?*, and *provider-customer** (known as the AS path “valley-free” rule), where “*” represents zero or more occurrences of

such a type of AS edge and “?” represents at most one occurrence. In other words, a provider-to-customer or peer-to-peer edge can only be followed by provider-to-customer or sibling-to-sibling edges [10], [11].

For example, national ISPs A and B are connected to their customer, regional ISP C. Although ISPs A and B are connected through ISP C, ISP A cannot reach ISP B via ISP C, since C as a customer does not provide transit services between its providers. Even if ISPs A and B can reach each other via other ISPs, the end-to-end performance characteristics between A and B cannot be inferred from the relationship between A and C and C and B. The delay between A and B is independent of the total delay between A and C and C and B. Therefore, information about AS relationships is an important factor of Internet topology. Typically, ASs want to filter and avoid propagating routes that use a small (customer) AS to transit between two larger ASs [12]. However, it has been found that some advertised AS paths do not conform to the valley-free property. It is not sound to assume that ASs advertise routes correctly in the first place. Detecting routes that violate policy remains a daunting problem in inter-domain routing.

References

1. Miniwatts Marketing Group Internet World Stats, “World Internet Users and Population Statistics,” <http://www.internetworldstats.com/stats.htm>, Oct. 2010.
2. Norton, W.B., “Internet Service Providers and Peering,” draft version 2.7, Jun. 2002.
3. Bartholomew, S., “The Art of Peering,” BT Technology J., vol. 18, no. 3, July 2000.
4. St. Arnaud, B., et al., “Optical BGP Networks,” white paper, Jul. 2000.
5. Pease, R., “Florida is Site of First Optical Internet Exchange,” Lightwave, Jan. 2001;
6. Huston, G., “Interconnection, Peering, and Settlements,” white paper, developed from Chapter 14 of the ISP Survival Guide, John Wiley & Sons, 1998.
7. Greene, B.R., “L2 Internet Exchange Point (IXP) using a BGP Route Reflector,” white paper, draft version 0.4, Dec. 2000.

- 8.** Halabi, S. and McPherson, D., Internet Routing Architectures, 2nd Edition, Cisco Press, 2001.
- 9.** Cukier, K.N., “Peering and Fearing: ISP Interconnection and Regulatory Issues,” presented at Conf. on the Impact of the Internet on Comm. Policy, Harvard Information Infrastructure Project, Dec. 1997.
- 10.** Norton, B., “Interconnection Strategies for ISPs,” white paper, version 2.0.
- 11.** Cook, G., “Scaling the Internet via Exchange Points,” The Cook Report on Internet, Cook Network Consultants, vol. 9, no. 10, Jan. 2001, pp. 7–15.
- 12.** Metz, C., “Interconnecting ISP networks,” IEEE Internet Computing, vol. 5, no. 2, pp. 74–80, Mar./Apr. 2001.

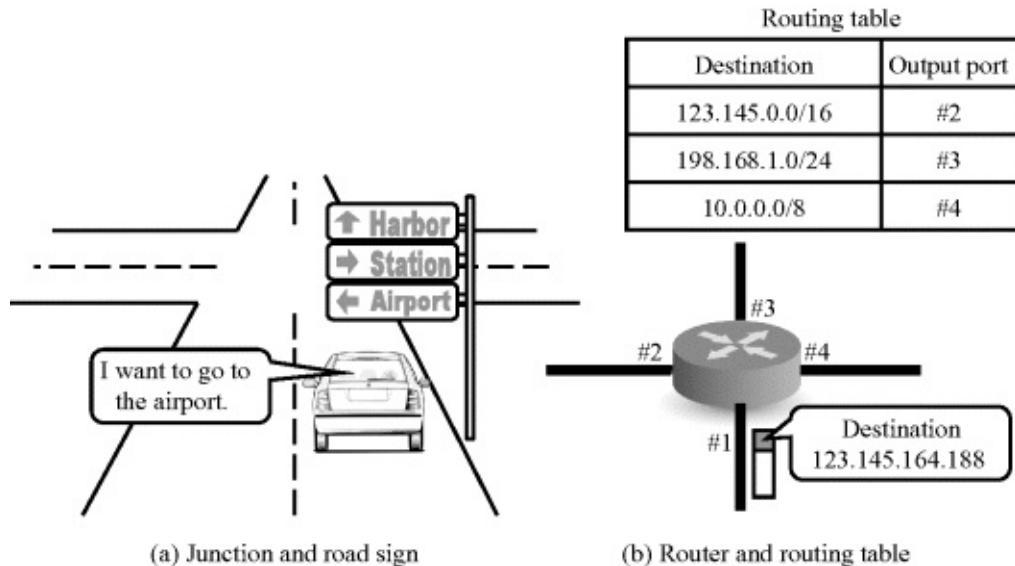
Chapter 4

IP Routing Protocols

Internet Protocol (IP) routing protocols are used to dynamically determine the best routes to send data from a source to a destination via routers or networks, while taking the network situation into account. A routing protocol shares the information with neighbor routers and throughout the network. IP routing protocols enable routers to dynamically build up and update their routing tables. The routing table contains destination information along with the corresponding output ports (i.e., the next hops) to which packets should be sent. Thus, packets are transmitted in a hop-by-hop manner, guided by the routing tables of each router on the route from the source to the destination.

The concept of the routing table is similar to a guide post on the road. In [Figure 4.1a](#), a car driver wants to go to the airport. At the junction, the driver turns to the left based on the road sign. In [Figure 4.1b](#), a packet arrives at the router. Its destination is 123.145.164.188. The router forwards the packet to output port 2 based on the information in the routing table.

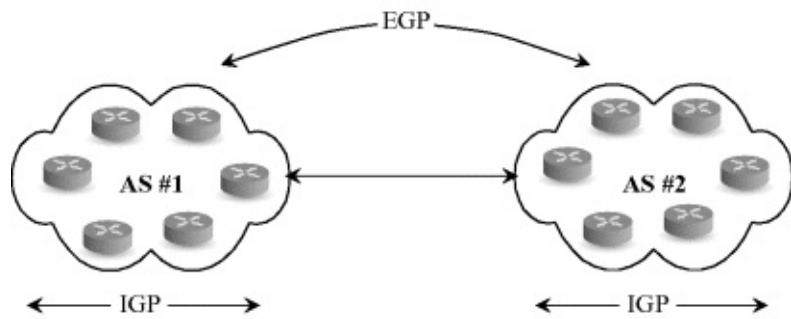
[Figure 4.1](#) Comparison of a routing table in a router with a guide post on the road.



This chapter describes IP routing protocols. Section 4.2 describes an

overview of routing protocols including Interior Gateway Protocol (IGP) and Exterior Gateway Protocol (EGP), as shown in [Figure 4.2](#). Section 4.3 describes the Routing Information Protocol (RIP) as an IGP. Section 4.4 describes the Open Shortest Path First (OSPF) protocol as another example an an IGP. Section 4.5 describes the Border Gateway Protocol (BGP) as an EGP.

Figure 4.2 Interior gateway protocol (IGP) and exterior gateway protocol (EGP).



4.1 Overview of Routing Protocols

Routing protocols are used to guide a data packet from its source to its destination along the best route based on network conditions. This is accomplished by creating and updating routing tables that enable neighboring routers to exchange routing information. If the network conditions change, for example a link failure occurs, the routing tables on the affected routes can be automatically and dynamically updated. This type of routing is called *dynamic routing*. The opposite of dynamic routing is *static routing*, where network administrators manually configure routing tables. If the network size grows large, it is difficult to manfully create routing tables. In addition, if network conditions change, it is impossible to immediately and consistently update all routing tables in the network.

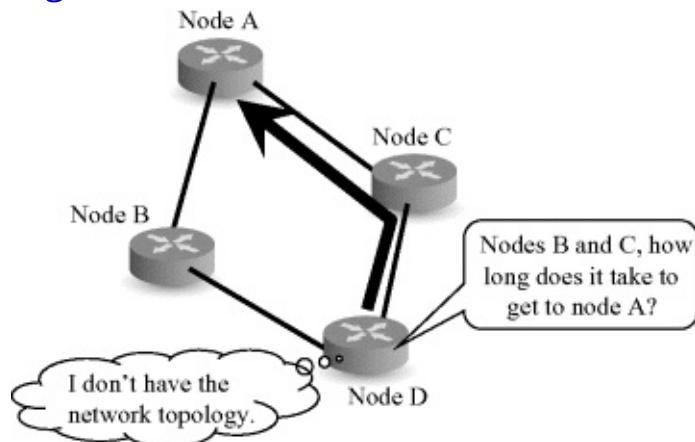
4.1.1 Interior Gateway Protocol

IGP is used to find the route within an autonomous system (AS), which is a collection of routers and hosts controlled by a single administrative entity. IGPs can be divided into two categories: distance-vector protocols and link-state protocols.

4.1.1.1 Distance Vector Protocols

Each router determines the best route and builds a routing table using routing information from neighboring nodes. Examples of distance-vector protocols include the Routing Information Protocol (RIP) [1, 2], the Interior Gateway Routing Protocol (IGRP), and the Enhanced Interior Gateway Routing Protocol (EIGRP). The RIP was standardized by the Internet Engineering Task Force (IETF), The IGRP and EIGRP were developed by Cisco Systems. [Figure 4.3](#) shows an example of the distance-vector protocol. There are four nodes in the same AS. Node D finds the route to node A using information about the distance from neighboring nodes B and C. Node A chooses the route D → C → A based on the information.

[Figure 4.3](#) Distance vector.

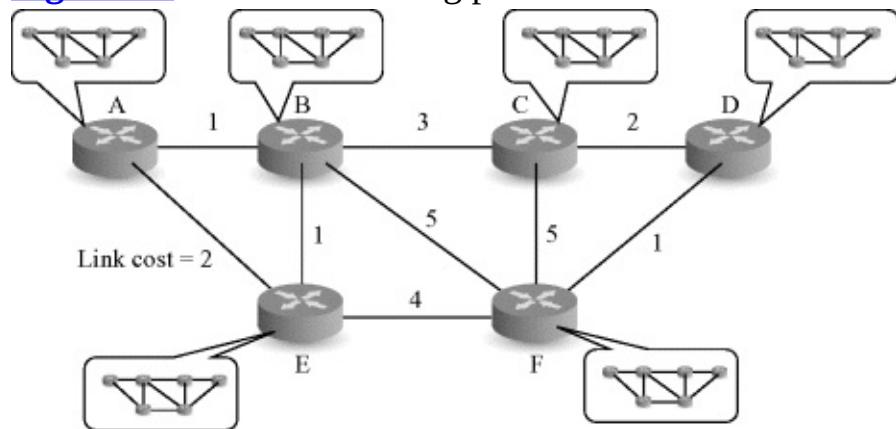


4.1.1.2 Link-State Protocols

Each router advertises link-state packets, which contain topology information about routers, links, and their costs, by flooding them to all routers in the network. Routers build a single topology map by receiving the link-state packets. As a result, all the routers in the network have the same common network topology information. Each router computes the shortest path based on link cost and uses that to build a routing table. Examples of link-state protocols include the Open Shortest Path First (OSPF) [3] and Intermediate System to Intermediate System (IS-IS). The OSPF was standardized in IETF, and IS-IS was specified as an international standard within the Open Systems Interconnection (OSI) reference design.

[Figure 4.4](#) shows an example of a link-state routing protocol. Each node knows the network topology. It computes the shortest path by itself.

Figure 4.4 Link-state routing protocol.

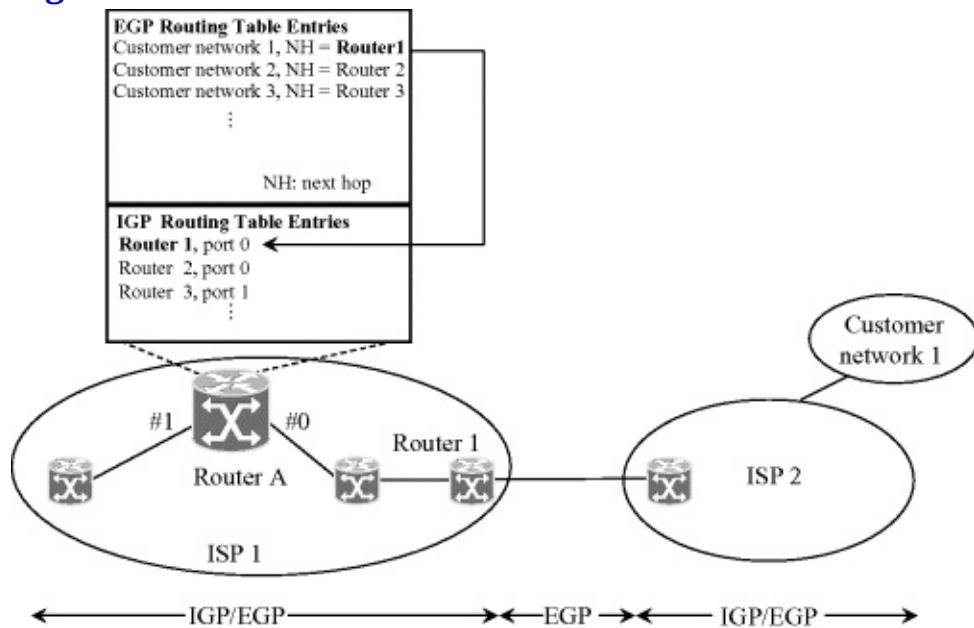


4.1.2 Exterior Gateway Protocol

EGP is used to find the route between different ASs. An example of EGP is the Border Gateway Protocol (BGP).

[Figure 4.5](#) shows how an IP packet is routed based on the two types of routing tables created by IGP and EGP in each IP router. Router A in ISP 1 tries to find the next hop to forward a packet destined to customer network 1. The EGP routing table says, “If you want to reach customer network 1, go to router 1.” At this point, the ISP IP core router checks the route to router 1 using the IGP routing table. The IGP routing table says, “If you want to reach router 1, go to port 0”.

Figure 4.5 IGP and EGP in IP routers.



The key features of IGP and EGP are compared in [Table 4.1](#).

Table 4.1 Comparisons of Features between IGP and EGP.

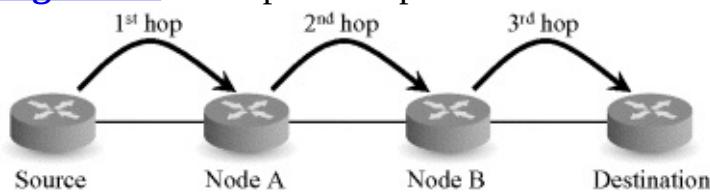
	IGP	EGP
Destination	Enables complete any-to-any reachability inside AS.	Enables reachability to external destinations based on policies.
Route by	Computes the shortest paths.	Attach and process multiple attributes to each route.
Information exchanging	Reliably floods all topology data.	Control distribution of routes.
Node discovery	Automatic.	Configured peer connections.
Information details	Only infrastructure prefixes.	Customer and internet prefixes.
Converge time	Fast. $O(\text{second})$.	Slow.
Operates on	Routers and hosts.	Border routers and routers that need to know about multiple exit points in AS.

4.2 Routing Information Protocol

RIP [1, 2] is a protocol in IGP that uses a distance-vector routing algorithm to identify the shortest path to any given destination addresses. RIP uses a single routing metric (i.e., a hop count) to measure the distance between source and destination routers. Each hop in a route from the source node to the destination routers is assigned a hop count value, which is typically set to one. When a router receives a routing update that contains a new or changed destination network entry with the corresponding metric value from a neighbor router, the router increase the metric value by one. If the increased metric value is lower than the current value that the router keeps in the routing table, or there is no information about the destination, the neighbor router is considered as the next hop and the routing update with the metric value is entered in the routing table.

[Figure 4.6](#) shows an example of the number of hops. The data travels from the source to the destination via node A and node B. The distance metric is three for this route.

Figure 4.6 Example of hops.



4.2.1 Routing Information Protocol Header Format

The protocol is first defined by RIP version 1 [1]. However, the information about the subnet is not included at that point. RIP version 2 (RIP2), which is discussed in more detail in [2], includes the ability to carry subnet information to support classless inter domain routing (CIDR). A route tag is also included in RIP2 to distinguish internal routes from external routes.

RIP is a User Datagram Protocol (UDP) based protocol [1]. Each router that runs RIP has a routing process that sends and receives via UDP port number 520. [Figure 4.7](#) shows the header format of both RIP and RIP2. The description of each field is explained below.

- *Command* indicates whether the packet is a request or a response.
- *Version* specifies the RIP version used.
- *Address family identifier* (AFI) specifies the address family used. The AFI for the IP is 2.
- *Route tag* provides a method to distinguish internal routes (learned by RIP) from external routes (learned from other protocols).
- *IP address* specifies the IP address for the entry.
- *Subnet mask* contains the subnet mask for the entry.
- *Next hop* indicates the IP address of the next hop.
- *Metric* indicates how many inter network hops (routers) have been traversed in the trip to the destination. This value is between 1 and 15 for a valid route; 16 indicates an unreachable route.

[Figure 4.7](#) Header of RIP and RIP2.

0	8	16	
Command	Version	Zero	Ethernet header
Address family identifier (AFI)		Zero	IP header
	IP address		UDP header
	Zero		RIP header
	Zero		
	Metric		

(a) RIP version 1 (RIP)

Command	Version	Zero	Ethernet header
Address family identifier (AFI)		Route tag	IP header
	IP address		UDP header
	Subnet mask		RIP2 header
	Next hop		
	Metric		

(b) RIP version 2 (RIP2)

4.2.2 Update of Routing Table in Routing Information Protocol

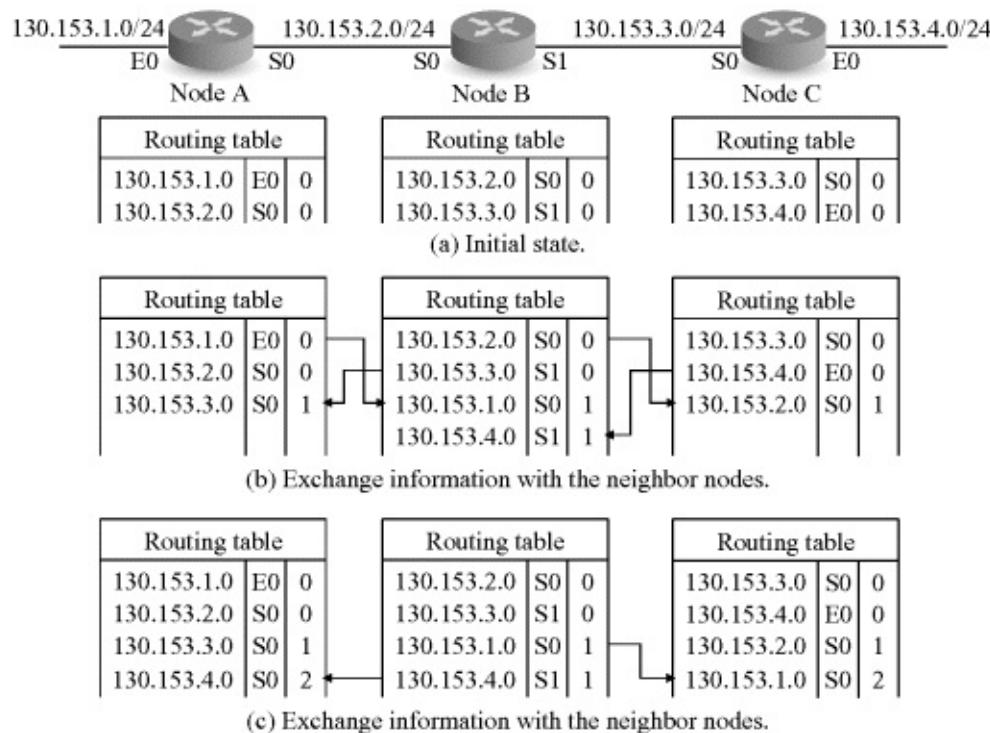
The mechanism for updating routing tables in RIP is as follows:

- Routing-update messages are sent to neighboring nodes at regular intervals (e.g., every 30 seconds) and when the network topology changes.
- The metric value for the destination that is received from the neighboring node is increased by one. If the increased metric value is the lowest among values for the same destination, the neighbor node is selected as the next hop router.
- Routers maintain only the best route that has the lowest metric value for each destination.
- After updating its routing table, the router immediately transmits the routing updates to inform the neighboring routers of the changes. These updates are sent independently of the regularly scheduled updates that routers send.
- The maximum number of hops in a route is 15. If a router receives an updated metric from neighboring routers that when increased by 1 exceeds the maximum number of hops, the network destination is considered unreachable.

[Figure 4.8](#) shows an example of updating the routing table. At the initial

state ([Figure 4.8a](#)), each router adds the networks that directly connect to its interfaces. Node A connects to networks 130.153.1.0 and 130.153.2.0 via ports E0 and S0, respectively. Node B connects to networks 130.153.2.0 and 130.153.3.0 via ports S0 and E0, respectively. Finally, node C connects to networks 130.153.3.0 and 130.153.4.0 via ports S0 and S1, respectively.

Figure 4.8 Updating of routing tables.



When routers reach the time to update, they exchange information with the neighboring routers ([Figure 4.8b](#)). Node A receives information about network 130.153.3.0 from node B, which is connected via S0. The distance, which means the number of hops, is one. Node B receives the information from nodes A and C. The updated addresses are 130.153.1.0 from node A and 130.153.4.0 from node C connected via the interfaces S0 and S1, respectively. The distance for both of them is one. Node C receives information about network 130.153.2.0 from node B. The interface connected to the network is S0. The distance is one.

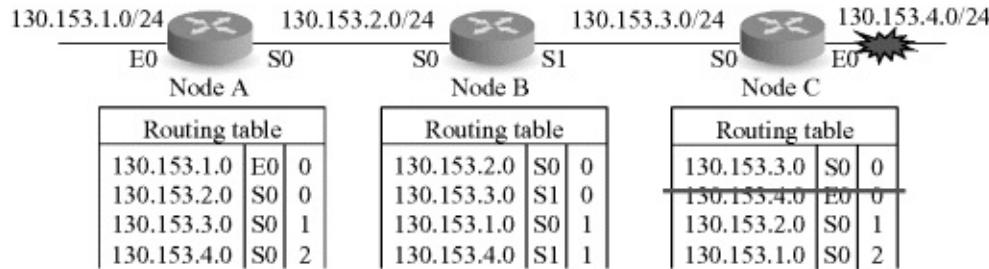
Once the routers reach the updating time again, node A receives information about 130.153.4.0 from node B. The distance to the network 130.153.4.0 becomes two, increasing by one from node B. Node B has no new information from neighboring nodes. Node C receives information about 130.153.1.0 from node B. The distance of 130.153.1.0 becomes two.

4.2.3 Maintenance of Routing Table in Routing Information Protocol

If any link in the network is broken, the routing table is updated. Since all of the routers are not synchronously updated, a router keeps announcing the broken link to neighboring routers. As a result, the distance of the broken link is counted to infinity, or 16 hops, and the link becomes unreachable.

[Figure 4.9](#) shows an example of the count-to-infinity problem. Each node has the routing information. The link that connects to interface E0 is broken. The information about 130.153.4.0 in the routing table of node C is deleted ([Figure 4.9a](#)). Node C receives the updated information about 130.153.4.0 from node B via interface S0 of node C ([Figure 4.9b](#)). The distance to 130.153.4.0 from node C becomes two. Next, node B receives the updated information about 130.153.4.0 from node C. The distance becomes three to 130.153.4.0 from node B ([Figure 4.9c](#)). Node A receives the information about 130.153.4.0 and updates the distance from two to four. Finally, the distance of 130.153.4.0 in each node reaches to infinity, or 16 hops ([Figure 4.9d](#)).

[Figure 4.9](#) Count-to-infinity problem.



(a) The link on the interface E0 of node C is broken.

Routing table		Routing table		Routing table	
130.153.1.0	E0 0	130.153.2.0	S0 0	130.153.3.0	S0 0
130.153.2.0	S0 0	130.153.3.0	S1 0	130.153.4.0	S0 2
130.153.3.0	S0 1	130.153.1.0	S0 1	130.153.2.0	S0 1
130.153.4.0	S0 2	130.153.4.0	S1 1	130.153.1.0	S0 2

(b) Node C gets the information about 130.153.4.0 from node B.

Routing table		Routing table		Routing table	
130.153.1.0	E0 0	130.153.2.0	S0 0	130.153.3.0	S0 0
130.153.2.0	S0 0	130.153.3.0	S1 0	130.153.4.0	S0 2
130.153.3.0	S0 1	130.153.1.0	S0 1	130.153.2.0	S0 1
130.153.4.0	S0 4	130.153.4.0	S1 3	130.153.1.0	S0 2

(c) Node B gets the information of 130.153.4.0 from node C.

Node A gets updated information from node B.

Routing table		Routing table		Routing table	
130.153.1.0	E0 0	130.153.2.0	S0 0	130.153.3.0	S0 0
130.153.2.0	S0 0	130.153.3.0	S1 0	130.153.4.0	S0 ∞
130.153.3.0	S0 1	130.153.1.0	S0 1	130.153.2.0	S0 1
130.153.4.0	S0 ∞	130.153.4.0	S1 ∞	130.153.1.0	S0 2

(d) Distance of 130.153.4.0 in all nodes becomes infinity.

Thus, it takes time to converge the routing information in the network if a network failure happens in RIP. This is called a *count-to-infinity problem*.

4.2.4 Split Horizon

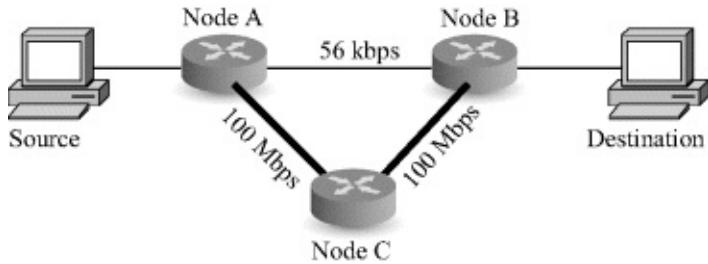
The count-to-infinity problem is one of the reasons why the maximum hop count of RIP for IP internetworks is set to 15 (16 indicates unreachable). Maximum hop count values higher than 15 would make the convergence time longer when counting to infinity. One solution to reduce the convergence time is called *split horizon*. It prohibits a router from announcing the updated information back to the direction from which it was received. If link failure occurs, the routers that connect to the link failure advertise a new metric, 16 (unreachable), to neighboring nodes.

4.2.5 Limitations of Routing Information Protocol

Although RIP is a simple protocol, it has some limitations.

- The shortest path is considered based only on the number of hops, although the speed in other routes may be faster than the route with the minimum number of hops. As in [Figure 4.10](#), the shortest path from the source node to the destination node is route A → B because there are fewer hops than in route A → C → B. However route A → C → B is faster than the route A → B.

Figure 4.10 Limitation of shortest path in RIP.



- The number of hops cannot exceed 16. Otherwise, the destination will be considered unreachable. The network size that RIP can be applied to is restricted.
- There is no concept of areas or boundaries.
- The count-to-infinity problem might occur.

4.3 Open Shortest Path First

OSPF [3] is a link-state routing protocol that is used as an IGP. Each router in the network has its own database of network topology, which is called a link-state database (LSDB). A shortest path first algorithm is used in order to build and compute the shortest path to all destinations based on the LSDB. The overview of the OSPF process is as follows:

- Initially, a router establishes the neighbor relationship with neighboring routers. Once the neighbor relationship is established, it is maintained by exchanging hello messages.
- A router advertises link states to neighboring routers including information about both links that originated from the router and from other neighboring routers. Thus, all routers exchange link-states, which is called *flooding*. Each router that receives a link-state update stores a copy in the LSDB and propagates the update to neighboring routers.

- A router computes the shortest paths to all destinations based on the LSDB.
- In the event of network changes, for example, the cost of a link is updated or a link is added or deleted, the updated information is flooded through link-state packets, and the shortest path is computed in each router.
- A router creates and updates the routing table based on the computed shortest paths.

4.3.1 Shortest-Path Algorithm

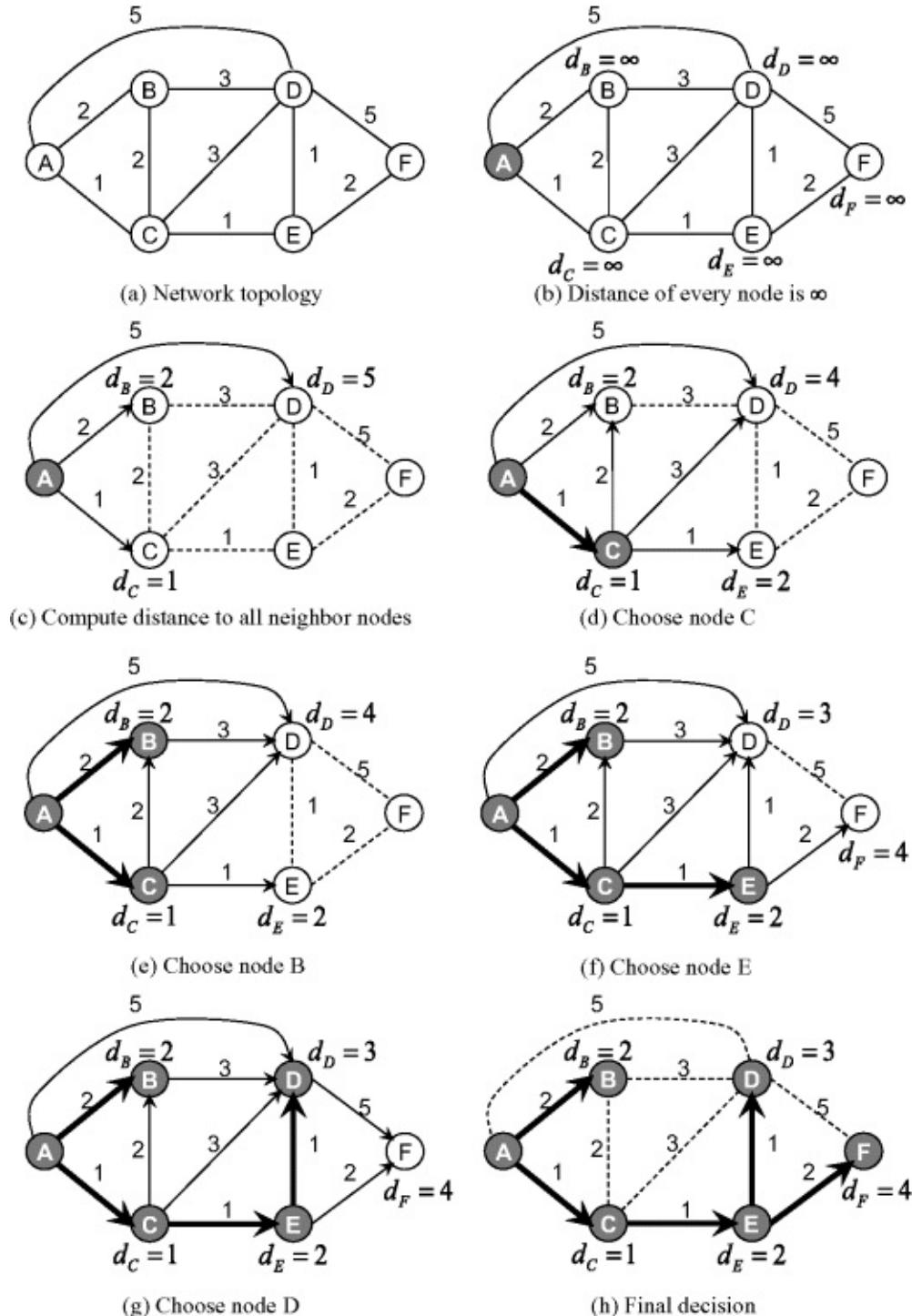
To compute the shortest path from a source router to a destination router, the Dijkstra algorithm [4] is used. An overview of the Dijkstra algorithm is described as follows:

- Step 1: Set the distance for the source node to zero, and set all other nodes to infinity.
- Step 2: Mark all nodes as unvisited. Set the source node as a current one.
- Step 3: For a current node, consider all its unvisited neighbors and compute their distance from the source node. If the distance is less than the previously recorded distance, the previous distance is replaced with the new one in the record. The previous hop is also updated with the new previous hop in the record.
- Step 4: Choose the unvisited node whose distance is shortest, and set it to a current node. Mark the previous current node as visited. A visited node will not be checked anymore. The recorded distance of the current node is the smallest one among unvisited nodes, and it is final.
- Step 5: If all nodes have been visited, the process is finished. Otherwise, repeat from Step 3.

[Figure 4.11](#) demonstrates the Dijkstra algorithm. There are six nodes in the network: nodes A to node F ([Figure 4.11a](#)). Node A, a source node, finds the shortest path to each node. Set the distance for node A to zero, and set all other nodes to infinity (Step 1) ([Figure 4.11b](#)). Initially, every node is marked as unvisited, and node A is set as a current node (Step 2) ([Figure 4.11c](#)). Node A has three neighbors: nodes B, C, and D. The distance to node B is two, $d_B = 2$; the distance to node C is one, $d_C = 1$; and the distance to node D

is five, $d_D = 5$ (Step 3).

Figure 4.11 Finding the shortest path using the Dijkstra algorithm from node A.



Node C, which has the shortest distance from node A, is selected as a current node (Step 4) ([Figure 4.11d](#)). Therefore, route A → C becomes the

shortest path between nodes A and C. Node A is marked as visited (Step 4). Neighboring nodes B, D, and E to node C are considered (Step 3). The distance of route $A \rightarrow C \rightarrow B$ is three. However, the current distance is $d_B = 2$, which is shorter than route $A \rightarrow C \rightarrow B$. Therefore, the distance from node A to B is $d_B = 2$, and the route is $A \rightarrow B$. The distance of route $A \rightarrow C \rightarrow D$ is four. It is shorter than route $A \rightarrow D$. The new distance from node A to D is $d_D = 4$, and the new route becomes $A \rightarrow C \rightarrow D$. The distance of route $A \rightarrow C \rightarrow E$ is two, $d_E = 2$.

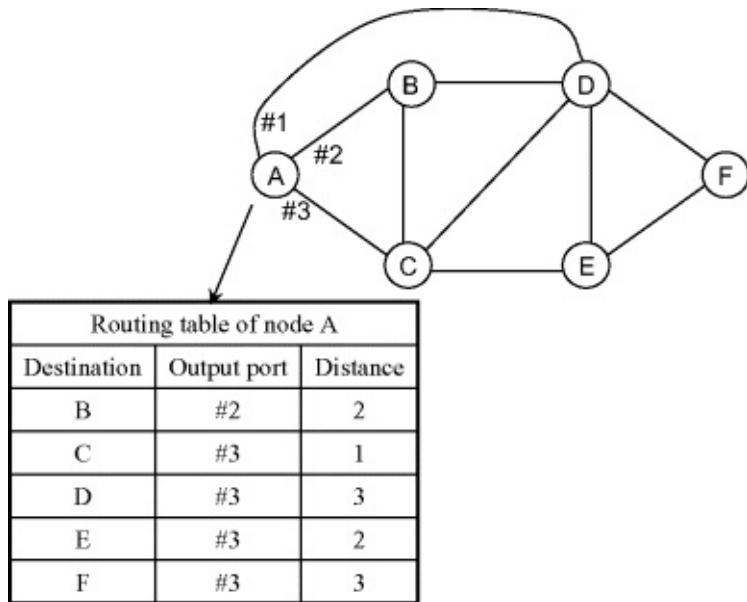
In [Figure 4.11e](#) node C is marked as visited (Step 4). The shortest distances are $d_B = 2$ and $d_E = 2$. Node B is selected as a current node out of two (Step 4). The distance of route $A \rightarrow B \rightarrow D$ is five, which is longer than the recorded route, $A \rightarrow C \rightarrow D$. The d_D still remains the same value, four (Step 3).

In [Figure 4.11f](#), node B is marked as a visited node (Step 4). Node E is selected as a current node because d_E has the smallest value among unvisited nodes (Step 4). The distance of route $A \rightarrow C \rightarrow E \rightarrow D$ is three, which is smaller than the recorded d_E . $d_E = 3$ is a new record. The distance of route $A \rightarrow C \rightarrow E \rightarrow F$ is four, $d_F = 4$ (Step 3).

In [Figure 4.11g](#) node E is marked as a visited node (Step 4). Node D becomes a current node (Step 4). The distance of route $A \rightarrow C \rightarrow E \rightarrow D \rightarrow F$ is eight, which is longer than $d_F = 4$. To reach node F, route $A \rightarrow C \rightarrow E \rightarrow F$ still remains (Step 3). By repeating the same process, all the nodes becomes visited (Step 5) and the algorithm is finished.

[Figure 4.11h](#) shows the final decision about the shortest paths from node A to all nodes. [Figure 4.12](#) shows the routing table obtained for node A.

[**Figure 4.12**](#) Routing table of node A.

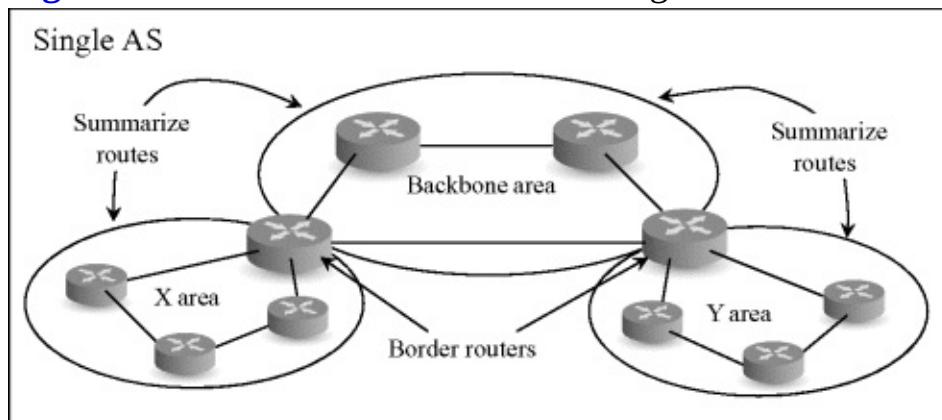


4.3.2 Hierarchical Routing

If the size of a network becomes large, it takes a significant amount of time to advertise the link-state information throughout the whole network, which is an AS and requires a large database to maintain the LSDB in each router. This also leads to a large number of entries in the routing table. To solve this problem, OSPF adopts a two-level hierarchical routing structure.

The network in a single AS is divided into several areas, as shown in [Figure 4.13](#). One of the areas is specified as a *backbone area*. All other areas are connected to the backbone area by *border area routers*. When a router in area X communicates with another router in area Y, a packet from the source router is transmitted to the destination router via area border routers through the backbone area.

[Figure 4.13](#) Two-level hierarchical routing structure.



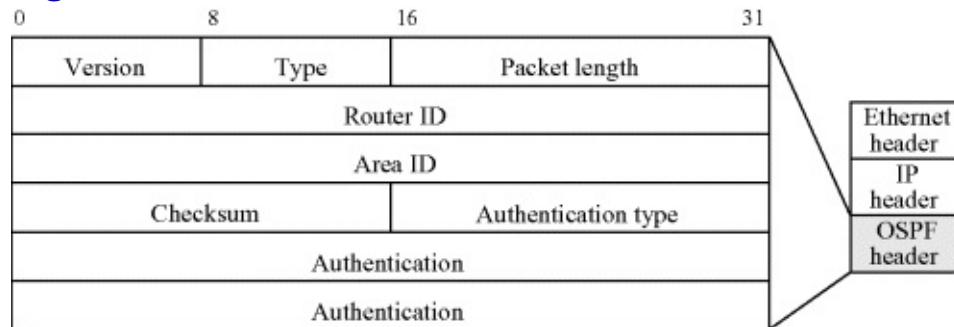
In the two-level hierarchical routing structure, the link states are distributed only within each area. Area border routers have link states in both the connected local area(s) and backbone area. The link states in one area are invisible to another area, but they are summarized and exchanged between areas by the area border routers. This can reduce the sizes of the LSDB and the routing table, making the network stable against topology changes. Note that the shortest path computation is performed within each area independently. Therefore, a route across areas may not always be optimal.

4.3.3 Open Shortest Path First Packet Format

OSPF packets run over IP. The IP protocol number of OSPF is 89. [Figure 4.14](#) illustrates the packet headers for OSPF version 2. The meaning of each field is explained below.

- *Version* indicates the version of OSPF.
- *Type* refers to one of the five types of OSPF packets: hello, database description (DBD), link-state request (LSR), link-state update (LSU), and link-state acknowledgment (LSack).
- *Packet Length* is the size of the OSPF message including the OSPF header.
- *Router ID* is the ID of the source router.
- *Area ID* is the OSPF area ID where the packet originated.
- *Checksum* is used for packet header error detection.
- *Authentication type* identifies the authentication procedure to be used for the packet.
- *Authentication* is used with the authentication type. A more detailed discussion is available in [3].

[Figure 4.14](#) Packet header of OSPF version 2.



4.3.4 Comparison of Routing Information Protocol and Open Shortest Path First

The differences between RIP and OSPF are shown in [Figure 4.15](#).

- *Number of hops*: RIP is limited to only 15 nodes, while there is no limitation in OSPF.
- *Metric (distance/cost)*: The distance in RIP is based on the number of hops. In OSPF, metrics are given by network administrators that take into account bandwidths, delays, traffic load, and so on.
- *Update time*: RIP updates the routing table periodically. OSPF instantaneously propagates after the information is changed.
- *Convergence time*: RIP converges slower than OSPF.
- *Authentication*: Authentication is not supported in RIP, while there are different methods for authentication in OSPF.
- *Area concept*: There is no concept of area in RIP. Two-level hierarchical routing structure is adopted in OSPF.

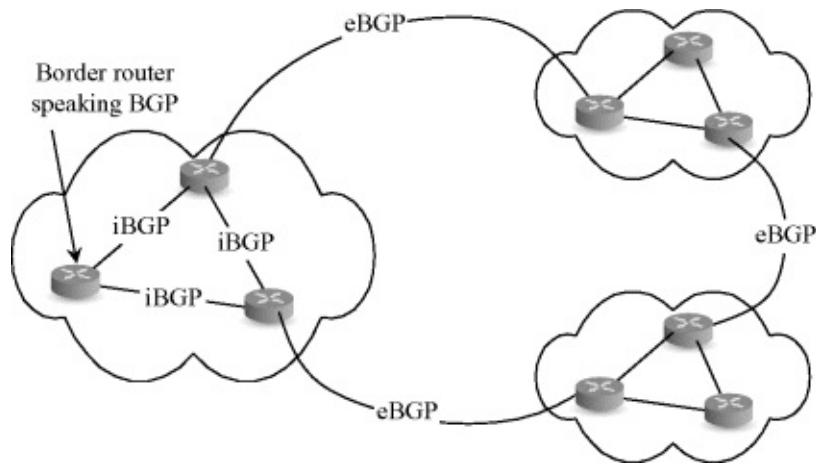
[Figure 4.15](#) Comparison of RIP and OSPF.

	RIP	OSPF
Number of hops	Maximum 15 nodes	No limit
Concept of distance	Number of hops	Delays and link costs
Update time	Periodic	Instantaneously propagation
Convergence time	Slow	Fast
Authentication	No	Yes
Area concept	No	Yes

4.4 Border Gateway Protocol

Border gateway protocol (BGP) [5] is an EGP that decides the direction of information between different ASs based on cost, performance, security, availability, traffic type, peer arrangements, and other factors. BGP runs over TCP, specified by TCP port number 179, using path vectors for policy routing and loop prevention. It supports route aggregation, variable length subnet mask (VLSM), and classless inter domain routing (CIDR). Internal BGP (iBGP) is used when BGP is running within an AS, and external BGP (eBGP) is used between ASs, as shown in [Figure 4.16](#).

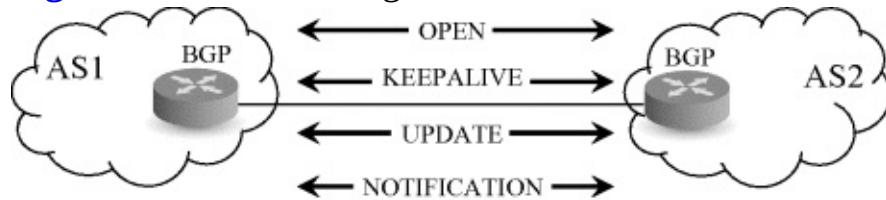
[Figure 4.16](#) iBGP and eBGP.



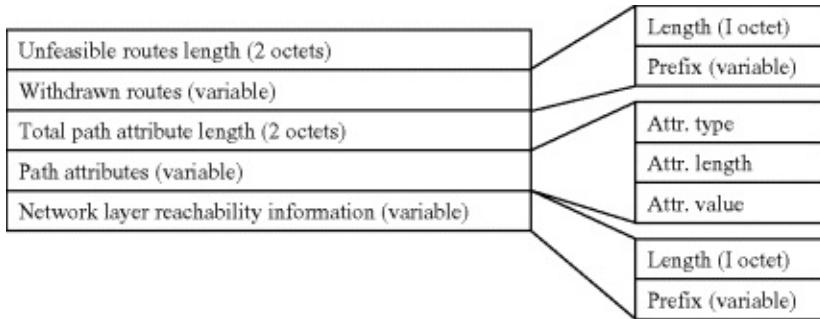
4.4.1 Border Gateway Protocol Message Flows

BGP neighbors, or peers, establish a TCP connection by manual configuration between routers. [Figure 4.17](#) depicts the BGP message flows. After the connection is established, each peer sends an OPEN message as the first message. If the OPEN message is accepted, a KEEPALIVE message will be sent back. The KEEPALIVE message maintains the connection. It is sent periodically; 60 seconds is the default interval. An UPDATE message is sent when information about the BGP changes. This message is used to advertise a single feasible route to a peer or to withdraw multiple unfeasible routes. The graph that describes the relationships of ASes is built based on this message. [Figure 4.18](#) displays the format of the UPDATE message header. If any error condition is detected, a NOTIFICATION message that contains cause of error is sent and the BGP connection is immediately terminated.

[Figure 4.17](#) BGP message flows.



[Figure 4.18](#) BGP update message format.



4.4.2 Border Gateway Protocol Policy Selection Attributes

BGP makes a decision about the best route using several parameters such as local preference, AS path length, multi exit discriminator (MED), next hop, and communities. A route is selected when the same route information is received from multiple peers. The best route is noted in the routing table. A peer sends the information along only the best route. The route selection process is composed of a set of rules applied in sequence.

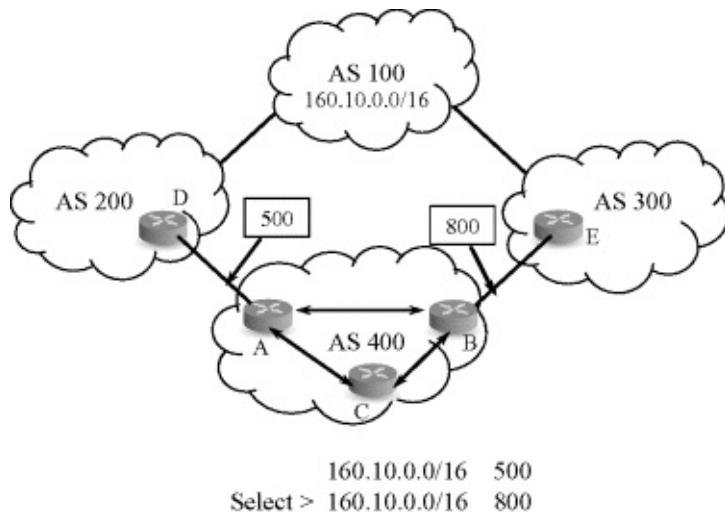
- *Local preference*: Choose the highest value.
- *AS path length*: Choose the shortest length.
- *MED*: Choose the lowest value.
- *Cost IGP route to the BGP next hop*: Choose the lowest cost.
- *Tie breaking*: Choose one randomly.

Details of each attribute are explained as follows.

4.4.2.1 Local Preference

There is a degree of preference used to select a particular route when multiple routes exist. If more than one route is possible to reach the destination AS, the router will choose the exit with the highest local preference. [Figure 4.19](#) shows an example of local preference. Node C can reach AS100 from AS400 via two exits: nodes A and B. Node C chooses to exit through node B because the local preference of node B is higher than that of node A.

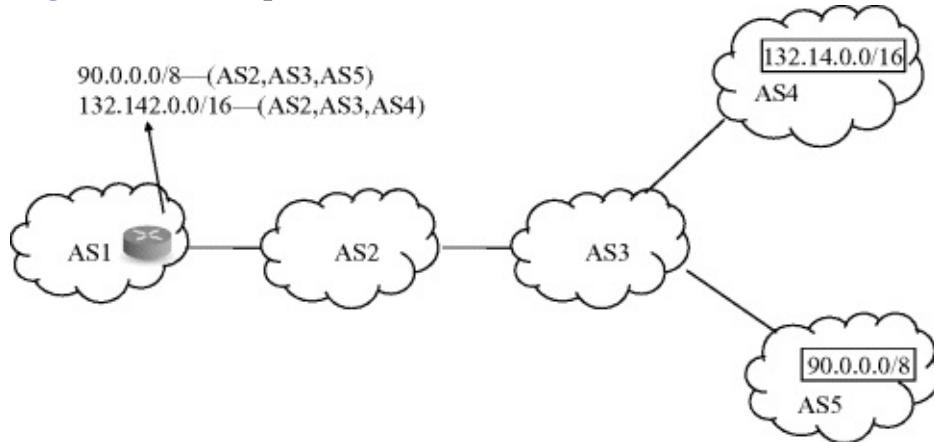
[Figure 4.19](#) Local preference.



4.4.2.2 AS Path Length

The AS path is the sequence of ASs on the route from the source AS to its destination AS. A mechanism for loop detection is provided. Policies may be applied based on the AS path. [Figure 4.20](#) shows the AS path from AS1 to AS4 and AS5 through AS2 and AS3. AS path length is the number of ASs that the information travels along the AS path. The router prefers a lower AS path length.

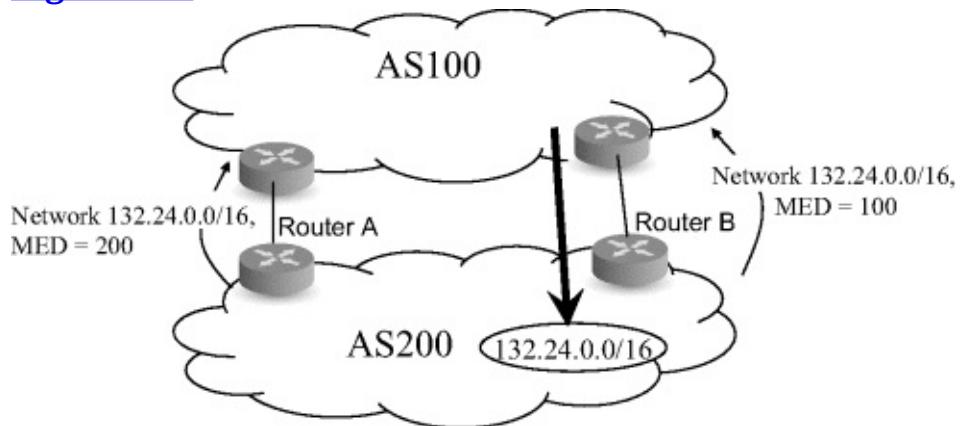
[Figure 4.20](#) AS path.



4.4.2.3 Multi Exit Discriminator (MED)

It is a hint to external neighbors about the preferred route into an AS when there are multiple routes. The lower MED value is preferred. [Figure 4.21](#) shows that router B is selected because the MED is lower than that of router A.

Figure 4.21 Multi exit discriminator.



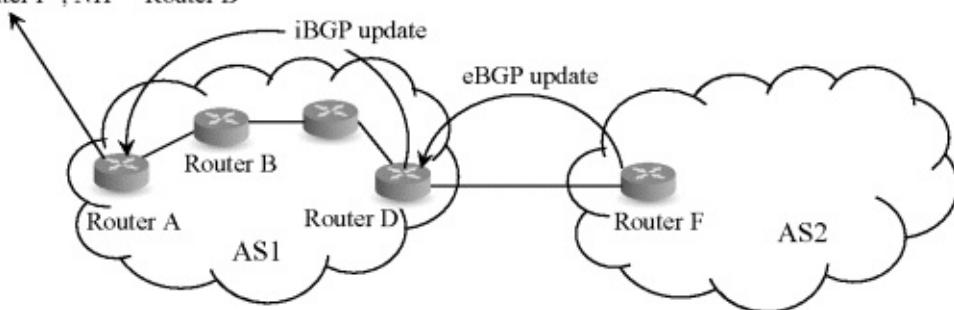
4.4.2.4 Next Hop

Next hop is refers to the IP address of the connection between peers. For iBGP, the information of the eBGP next hop is distributed among the local AS. [Figure 4.22](#) shows the concept of next hop. Router D gets updated information from router F via eBGP because they are located in different ASs. Router A gets updated information within the same AS from router D by iBGP. At router A, the BGP next hop is router F and the IGP next hop is router B.

Figure 4.22 Next hop.

BGP: 132.142.0.0/16, NH = "Router F"

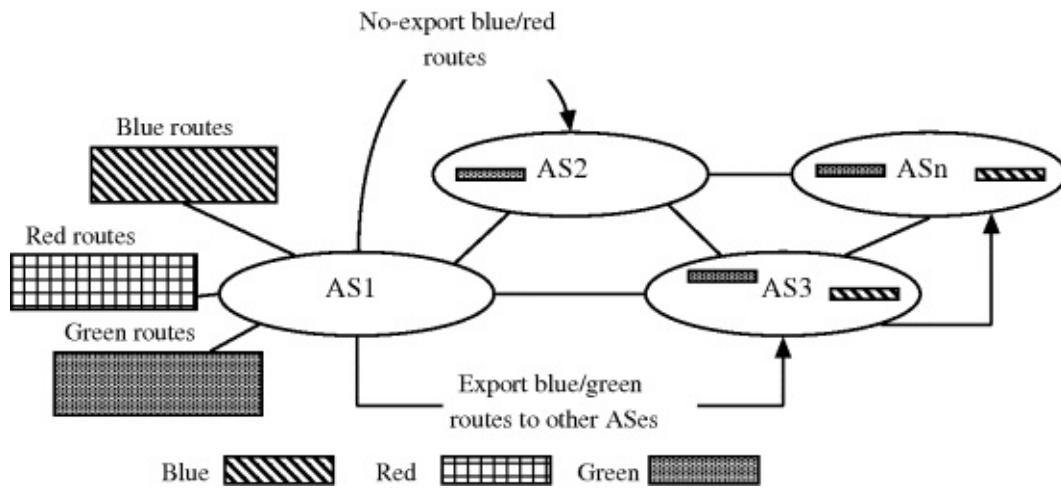
IGP: "Router F", NH = Router B



4.4.2.5 Communities

Communities provides a way to group ASs [6]. Sometimes this is called route coloring. All routes with the same color are processed in a similar manner. A single policy is applied to all routes within the same community. In [Figure 4.23](#), AS1 tells AS3 to pass the blue and green routes to ASn.

Figure 4.23 Communities.



References

1. Hedrick, C., “Routing Information Protocol,” IETF RFC 1058, Jun. 1988.
2. Malkin, G., “Routing Information Protocol,” IETF RFC 2453, Nov. 1998.
3. Moy, J., “OSPF Version 2,” IETF RFC 2328, Apr. 1998.
4. Dijkstra, E.W., “A note on Two Problems in Connexion with Graphs,” Numerische Mathematik 1, pp. 269-271, 1959.
5. Rekhter, Y. and Li, T., “A Border Gateway Protocol 4 (BGP-4),” IETF RFC 1771, Mar. 1995.
6. Chandra, R., Traina, P. and Li, T., “BGP Communities Attribute,” IETF RFC 1997, Aug. 1996.

Chapter 5

Multiprotocol Label Switching

This chapter describes Multiprotocol Label Switching (MPLS) technologies, which enable networks to perform traffic engineering, resident communication, virtual private networking, Ethernet emulation, and so on. First, an overview of MPLS is provided. Next, the functions and mechanisms of MPLS are described. Finally, MPLS applicabilities are discussed.

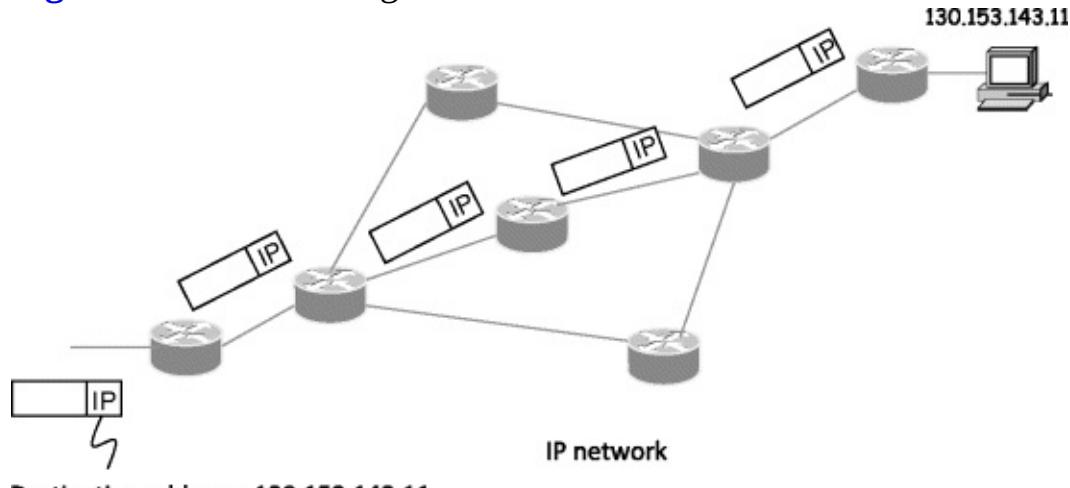
5.1 Overview

A key purpose of MPLS is to provide a tunneling mechanism that uses labels. MPLS is an Internet Engineering Task Force (IETF) standard intended to enhance the forwarding performance and traffic engineering intelligence of packet-based networks including Asynchronous Transfer Mode (ATM) and Internet Protocol (IP). MPLS communication is connection oriented, while IP datagram communication is connectionless. In the connection-oriented communication, a connection is established before starting communication. In connectionless communication, communication immediately starts by exchanging datagrams between source and destination hosts without establishing a connection. MPLS carries IP and non-IP payloads, and can operate over any data link layer. The layer of MPLS is considered to be between the data link layer (layer 2) and the network layer (layer 3). MPLS separates the control plane from the forwarding plane. MPLS enables the IP control plane to run on devices that cannot understand IP or recognize packet boundaries. The architecture of MPLS is discussed in more detail in [1].

An IP packet is forwarded in an IP network using the IP destination address included in the IP header in a hop-by-hop manner, as shown in [Figure 5.1](#). An IP packet that is destined for the destination address of 130.153.143.11 is transmitted to the IP network. When each IP router receives the IP packet, the router checks the header including the destination address. It then searches the next hop for the destination address in the routing table, where longest prefix matching is performed. The routing table may be automatically and

dynamically updated using routing protocols such as OSPF and BGP, or manually configured by the network administrators. Then, the IP packet is forwarded to the next hop. In the same way, the IP packet is transmitted from the source to the destination.

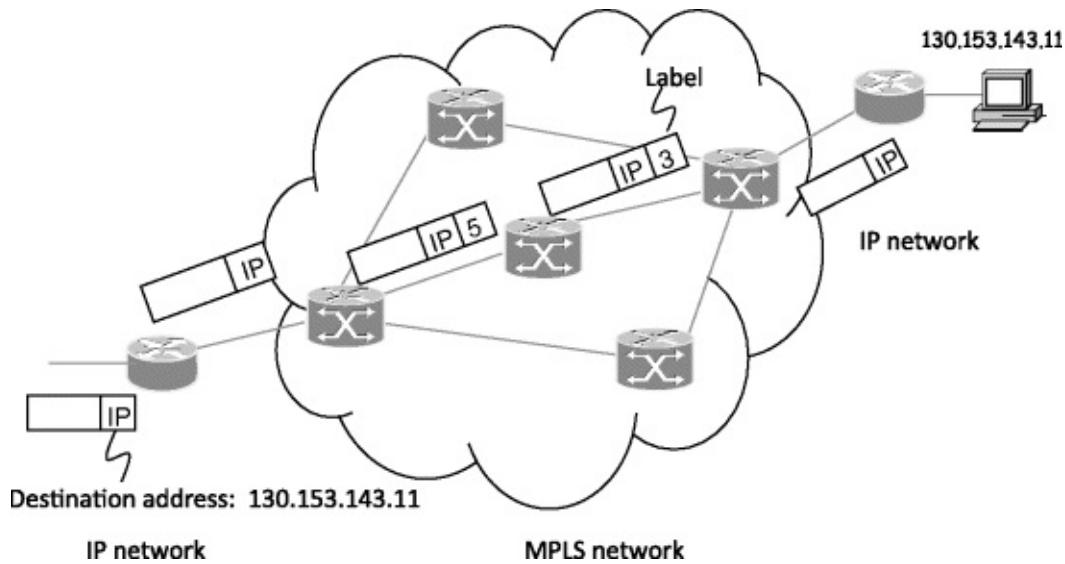
Figure 5.1 IP forwarding.



Destination address: 130.153.143.11

In an MPLS network, a label is attached to a packet for use in forwarding ([Figure 5.2](#)). The label number is uniquely assigned in each link before starting communications. This label assignment is equivalent to configuring a tunnel, which is called a label-switched path (LSP). An ingress router in the MPLS network attaches a label to each IP packet that is transmitted from an IP network and forwards the packet to the next hop. Thus the packet is transmitted through the LSP. The label operation at the ingress router in the MPLS network is called *push*. A transit node in the MPLS network switches the incoming label to the outgoing label and forwards the packet to the next hop. The label operation in the transit router in the MPLS network is called *swap*. An egress router in the MPLS network takes the label out, in which the operation is called *pop*, and forwards the packet to an IP network. This means that the packet gets out of the LSP.

Figure 5.2 MPLS forwarding.



Let us compare a label in an MPLS network with an IP address in an IP network. An IP address specifies a corresponding destination. An IP address must be unique throughout the Internet. By using the IP address, an IP packet is routed in a hop-by-hop manner after the longest prefix matching process is performed to find the next hop. On the other hand, a label specifies a corresponding virtual connection that is associated with a Forwarding Equivalence Class (FEC). A label must be unique through each link. An IP packet with an attached label is switched and transmitted along an associated LSP, and the label is taken out and attached at each transit router.

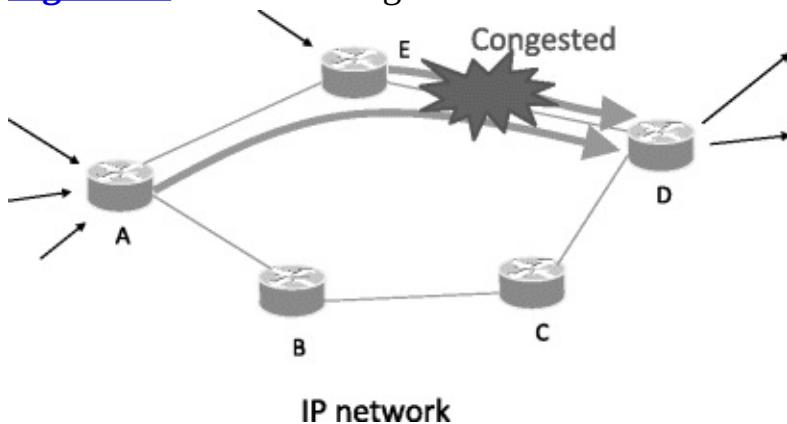
IP packets entering an MPLS network are divided into multiple classes called FECs, that are associated with one or more LSPs over which packets are transmitted to the egress router in the MPLS network. The class is called FEC. The classification is performed based on information from the header of the IP packet including the source and destination IP addresses, the source and destination port numbers such as TCP and UDP, and the protocol number. An example of FEC is shown in [Figure 5.3](#). Once IP packets entering the MPLS network are related to each FEC, corresponding labels are attached and they are forwarded to an associated LSP. Thanks to FEC, MPLS ensures scalability that supports aggregation into a forwarding class based on the IP header information.

Figure 5.3 Forwarding Equivalence Class (FEC).

FEC	SA	DA	SP	DP	PID
FEC 1	132.168.33.0/24	130.153.143.0/24	*	80	TCP
FEC 2	132.168.33.0/24	130.153.143.0/24	*	69	UDP
FEC 3	*	130.153.160.0/24	*	*	*
...

One of MPLS's applications is traffic engineering. MPLS enables flexible and explicit control of traffic routes to avoid network congestion, taking current traffic demands into consideration, IP routing protocols such as OSPF and IS-IS, do not support an explicit routing control mechanism. [Figure 5.4](#) shows an example of network congestion occurring in OSPF. In OSPF, a metric (weight, cost, or distance) is assigned to each link. Traffic routes between source and destination routers are selected based on the shortest path. In other words, the route is selected so that total metrics over the route can be minimized. In [Figure 5.4](#), consider that there is traffic between router A and router D, and between router E and router D.

[Figure 5.4](#) Network congestion in OSPF.

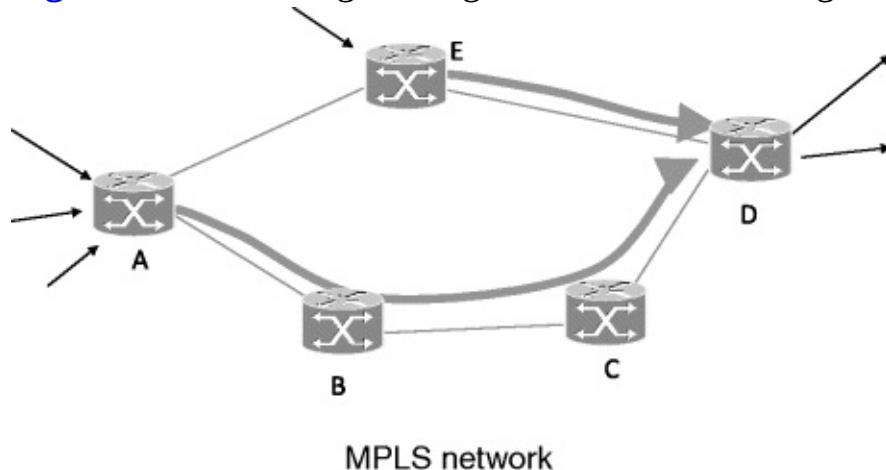


As a packet destined for router D from router A is routed through A-E-D and one destined for router D from router E is routed through E-D, the link between routers D and E becomes congested. In OSPF, the only way to control the traffic route is to adjust the link metrics in the network. However, network congestion cannot be entirely avoided by adjusting the link metrics.

In the MPLS network, it is possible to control the route of LSPs by assigning labels. [Figure 5.5](#) shows an example where network congestion is avoided in the MPLS network. The LSP carrying traffic from router A to router E is routed through A-B-C-D, while the LSP carrying traffic from router E to router D is routed through E-D. As a result, traffic volume passing through the link between routers E and D decreases, and the network

congestion is avoided.

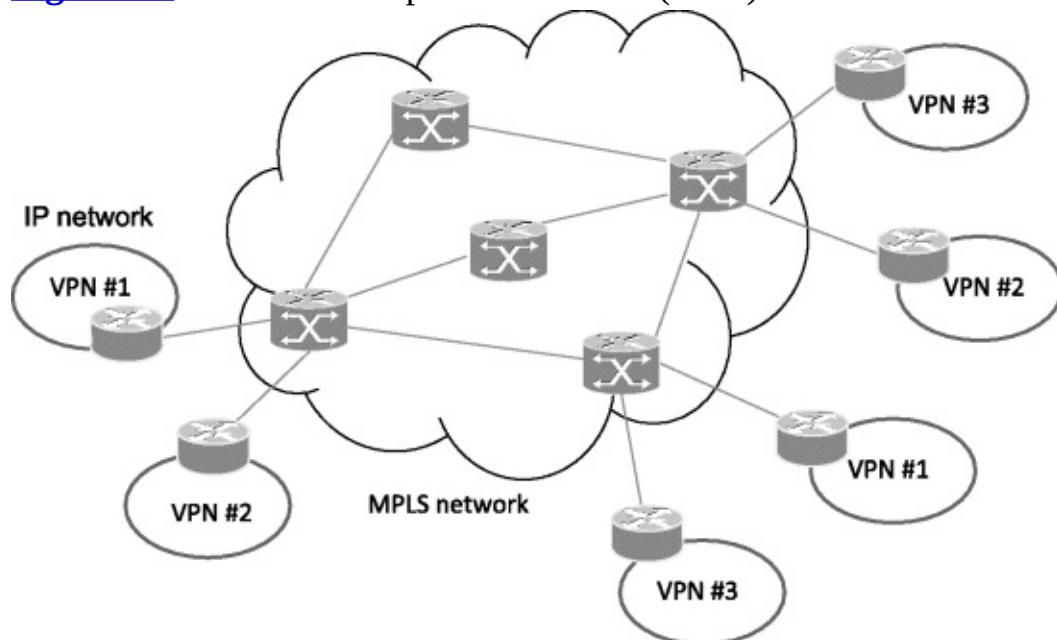
Figure 5.5 Traffic engineering to avoid network congestion.



MPLS network

Another application of MPLS is virtual private network (VPN) services. A network operator constructs a common MPLS network and provides logically separated networks for each user ([Figure 5.6](#)). Users can access their own IP network provided virtually through MPLS without being aware of the MPLS network. In each VPN, users can assign their IP address in their own way. As two different VPNs are logically independent, it is possible for IP addresses assigned in one VPN to be the same as addresses assigned in another VPN.

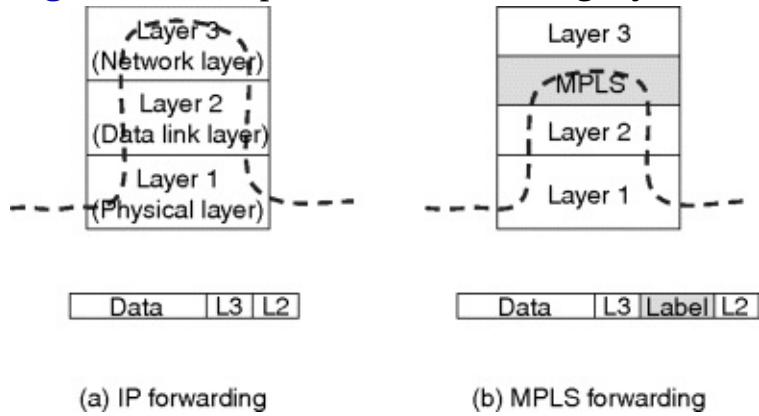
Figure 5.6 MPLS virtual private network (VPN).



5.2 Functions and Mechanisms

MPLS carries IP and non-IP payloads and can operate over any data link layer. The MPLS forwarding layer lies between the data link layer (layer 2) and the network layer (layer 3), as shown in [Figure 5.7](#). A label is attached to the top of the network layer packet (IP packet).

[Figure 5.7](#) Comparison of forwarding layers.



[Figure 5.8](#) shows label structures for different data link layers, Ethernet, Packet Over Synchronous Optical Network (POS), and Asynchronous Transfer Mode (ATM). For Ethernet and POS, a *shim header* is defined. The shim header includes a label (20 bits), 3 experimental (EXP) bits, bottom of stack S (1 bit), and time-to-live (TTL) (8 bits). Since each label is 20 bits, each link in the MPLS network accommodates at most 2^{20} LSPs. EXP bits are used to indicate the class of service information and are mapped from the IP packet at the ingress router into the IP packet at the egress router in the MPLS network. If S bit is set, it indicates that the label is on the bottom of the label stack. The TTL bits are mapped from the IP packet at the ingress router. The TTL bits in the shim header are decremented at each hop. The bits are mapped back into the IP packet at the egress router. In ATM networks, the combination of a Virtual Path identifier (VPI) and Virtual Channel Identifier (VCI) become the label for an MPLS LSP. A virtual path (VP) and a virtual channel (VC) are specified by the VPI and VCI assigned by each link, respectively. As shown in [Figure 5.9](#), an LSP goes across Ethernet, ATM, and POS networks. MPLS can operate over any data link layers.

[Figure 5.8](#) Label structures for different data link layers.

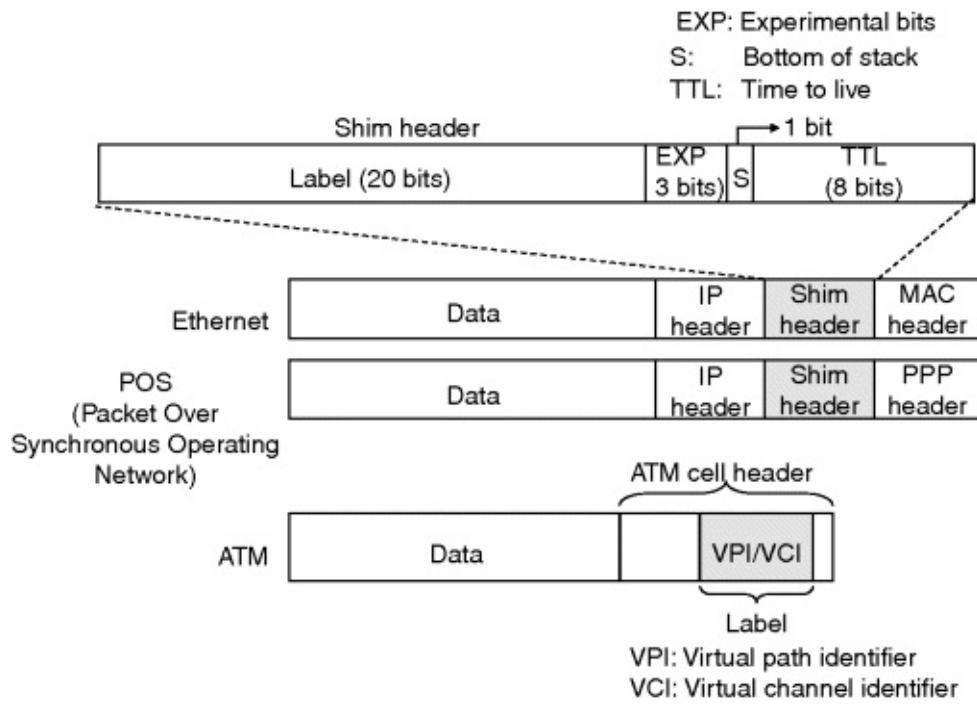
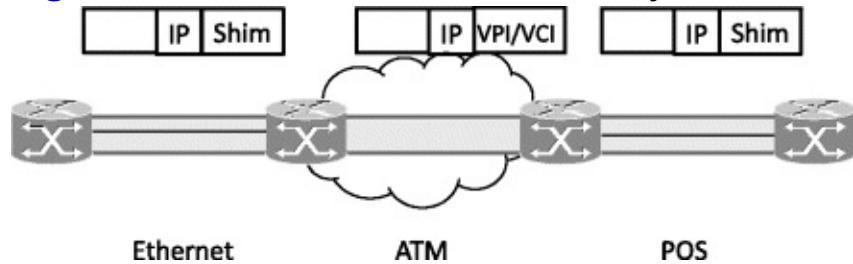
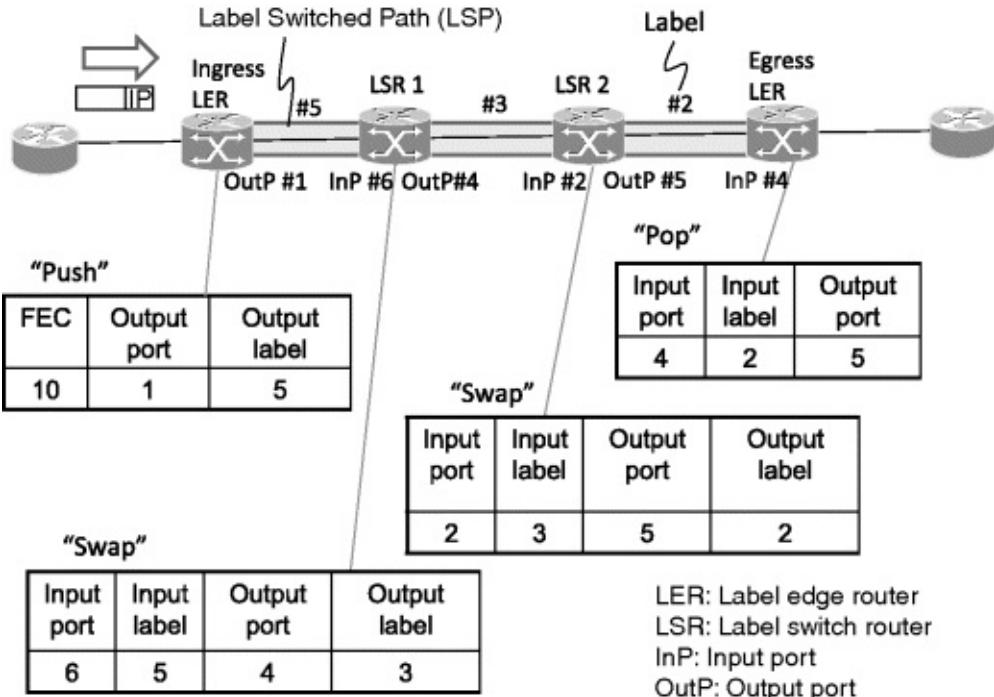


Figure 5.9 LSP over different data link layers.



The MPLS terminology and label operation are explained in [Figure 5.10](#). When an IP packet enters an MPLS network, a router receiving the packet searches a corresponding FEC based on the IP header information, and attaches a label associated with the FEC. The router is called an ingress label edge router (LER). A transit router of an LSP is called a label switching router (LSR). A router at the end point of the LSP is called an egress LER.

Figure 5.10 Label operation at LER and LSR.

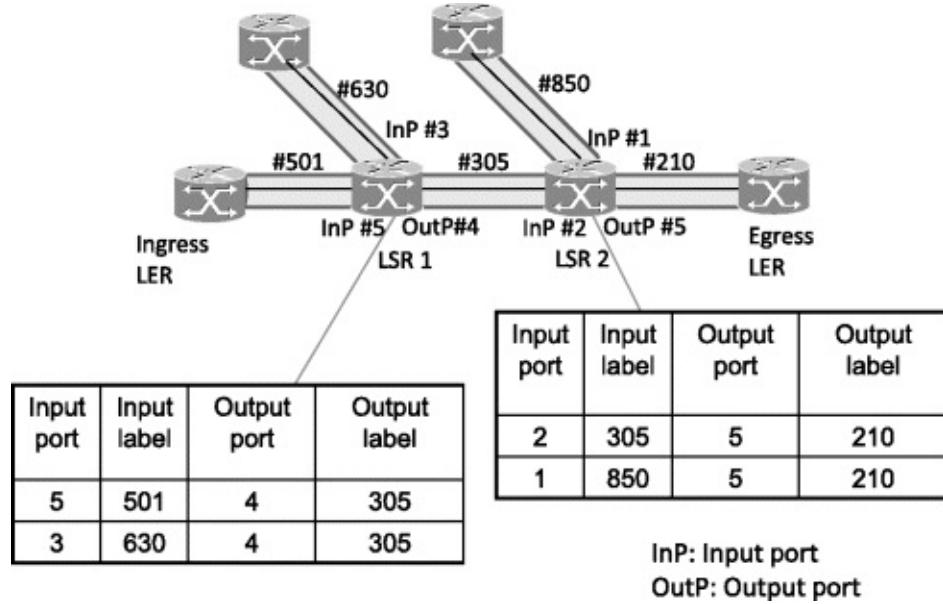


Label forwarding tables along an LSP are configured in an MPLS network, as shown in [Figure 5.10](#). The ingress LER receives an IP packet and relates the header information to a corresponding FEC by referencing an FEC table, as shown in [Figure 5.3](#). In this example, the IP packet is related to FEC 10. Then, the ingress LER attaches a label whose value is 5, with the label operation called *push*, and forwards the packet to output port 1. LSR 1, which is a transit router over the LSP, receives the packet with label 5 from input port 6, attaches the packet to label 3, and forwards it to output port 4. This label operation is called *swap*. Note that input and output numbers are uniquely assigned at each router. LSR 2 also executes label swap. LSR 2 receives the packet with label 3 from input port 2, attaches the packet to label 2, and forwards it to output port 5. The egress LER, which is the end point of the LSP, receives the packet with label 2 from input port 4, takes the label out, and forwards the packet to output port 5. The IP packet exits the LSP at the egress and is forwarded from the MPLS network to the IP network. This label operation is called *pop*. As an alternative way to pass through the same LSP, since it is not necessary for the packet to have a label between LSR 2 and the egress LER, which is the last hop link, LSR 2 may pop the label.

If packets coming in from different ingress LERs belong to the same FEC and share the same physical link, labels can be merged ([Figure 5.11](#)). This is called *label merging*. Label merging reduces the number of required labels in

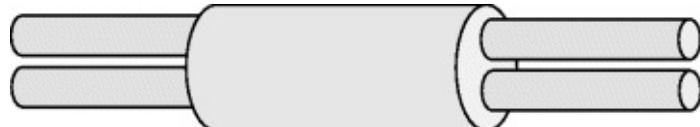
the MPLS network, and relaxes operational complexity. In this example, label merging is performed at LSRs 1 and 2. At LSR 1, input labels 501 and 630 from different input ports are merged into output label 305, which is associated with output port 4. At LSR 2, input labels 305 and 850 from different input ports are merged into output label 210, which is associated with output port 5.

Figure 5.11 Label merging.



Multiple LSPs can be accommodated under one LSP ([Figure 5.12](#)). This is called *LSP hierarchy*. The hierarchical LSP structure simplifies network management in the MPLS network. Multiple LSPs that belong to the same quality of service (QoS) class can be accommodated under one LSP. If the large LSP accommodating the small LSPs in [Figure 5.12](#) transits at an LSR, the number of LSPs that should be handled by the LSR is reduced.

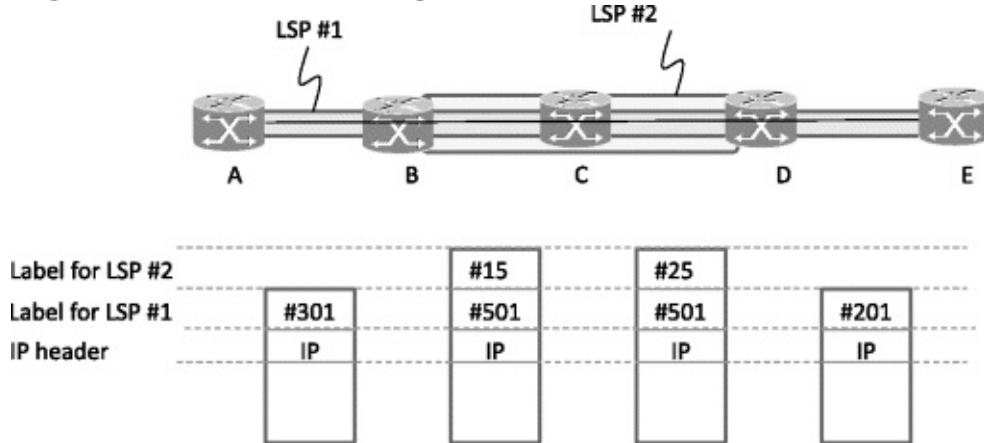
Figure 5.12 LSP hierarchy.



To provide an LSP hierarchy, a mechanism that attaches multiple labels to an IP packet is required. This is called *label stacking*. An example of label stacking is shown in [Figure 5.13](#). LSP 2 accommodates LSP 1. LSP 1 is set along route A-B-D-E. LSP 2 is set along route B-C-D. Router A pushes label 301 for LSP 1. Router B swaps input label 301 with output label 501 for LSP

1, and pushes label 15 for LSP 2, on top of label 501 of LSP 1. Router C swaps input label 15 with output label 25 for LSP 2, but it does not deal with label X of LSP 1. Router C does not have to keep any information about LSP 1 accommodated for LSP 2. Therefore, label 501 for LSP 1 is unchanged. Router D pops label 25 for LSP 2, and swaps input label 501 with output label 201 for LSP 1. Router E pops label 201 for LSP 1. S bit is set at each label for LSP 1 in the shim header, as defined in [Figure 5.8](#), because it is on the bottom of the label stack. Router C has only to swap one label for LSP 2, regardless of the number of accommodated LSPs in LSP 2. This simplifies the network management in the MPLS network.

[Figure 5.13](#) Label stacking.



5.3 Applicabilities

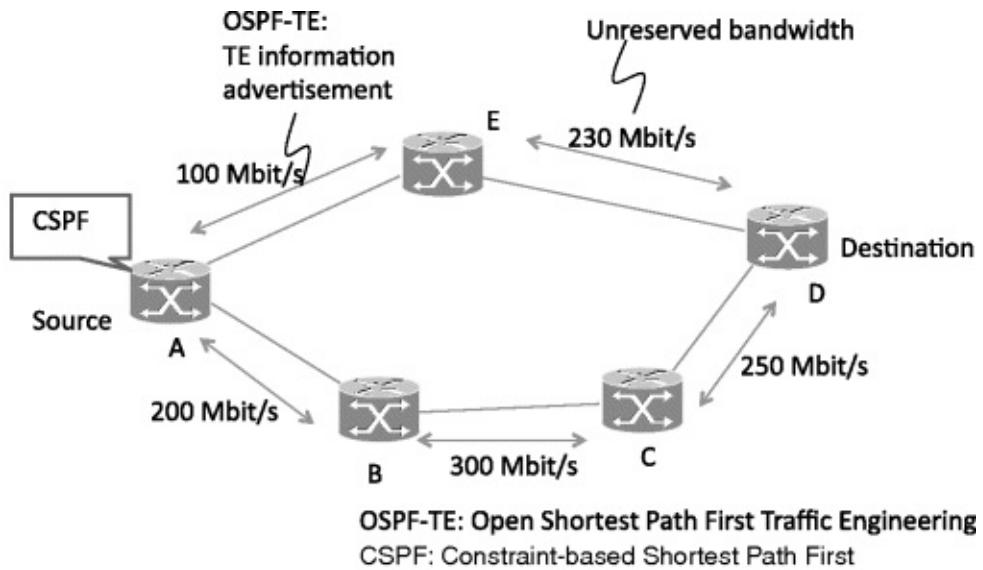
This section describes MPLS applicabilities including traffic engineering, resilience, and Ethernet emulation.

As described in Section 5.2, traffic engineering is one of the key MPLS applications. First, it is important to discuss how traffic engineering is achieved in the MPLS network. OSPF that is used in an IP network [2], is extended to distribute more detailed attributes of links [3]. A link that is specified by the extended attributes is called a *traffic engineering (TE) link*. The extended attributes for a TE link include TE metric, maximum bandwidth, maximum reservable bandwidth, unreserved bandwidth, and distractive group. All routers share attributes of each TE link in the MPLS network. When an LSP is requested to be set up between source and destination routers, the source router computes the shortest path that satisfies the requirements based on the distributed TE link information. The path

computation is called *Constraint-based Shortest Path First* (CSPF). By setting up the LSP based on CSPF policy, any network congestion is avoided and QoS is maintained in the MPLS network. On the other hand, since IP packets are routed based on the shortest path first policy in an IP network, some network congestion can occur, and QoS may not be maintained.

[Figure 5.14](#) shows examples of routes selected by the CSPF policy. Assume that the TE metrics of all the links are set to 1 and unreserved bandwidths for each link are given, as depicted in [Figure 5.14](#). Consider that an LSP is requested to be set up between routers A and D. Three computation examples are shown here. First, when the requested bandwidth is 100 Mbit/s, no link is a bottleneck in the network. The shortest path is found as route A-E-D. Second, when the requested bandwidth is 200 Mbit/s, link AE whose unreserved bandwidth is 100 Mbit/s is eliminated in the path computation because it does not meet the requirement. As a result, the shortest path in the network without link AE is route A-B-C-D. Third, when the requested bandwidth is 250 Mbit/s, links AE and AB whose unreserved bandwidths are less than requested are eliminated in the path computation. No route is found to meet the requirement.

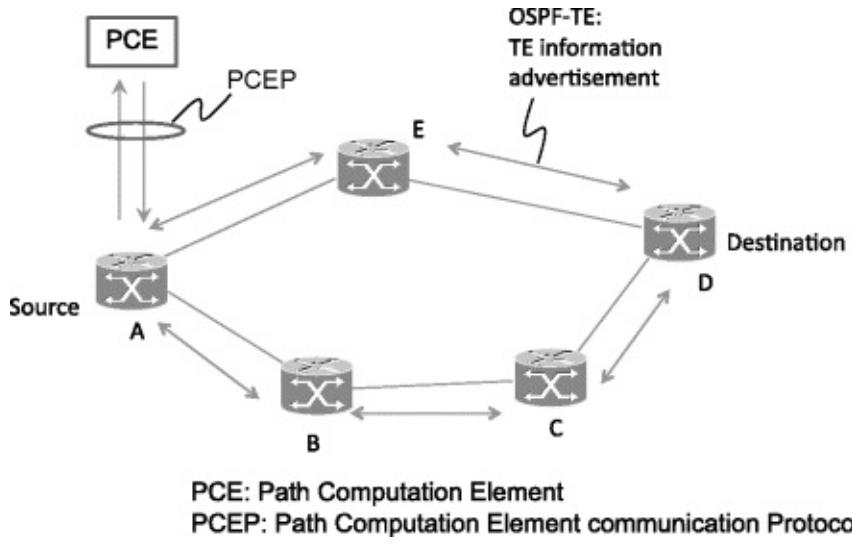
[Figure 5.14](#) Constraint-based Shortest Path First incorporated with OSPF-TE.



Requested bandwidth [Mbit/s]	Selected route
100	A-E-D
200	A-B-C-D
240	No path exist

To enhance the CSPF-based path computation in the MPLS network, the architecture of a Path Computation Element (PCE) is introduced [4]. The CSPF function that is available on each router to compute the LSP route is logically separated from the router as a PCE ([Figure 5.15](#)). When an LSP ingress router is requested to be set up, the LSP as a client, requests a PCE, as a server, to compute the route. This protocol between a client and a server is called PCE Communication Protocol (PCEP) [5]. The applicabilities of PCE include inter-domain TE [6, 7], inter layer TE [8], and global optimization of LSPs [9].

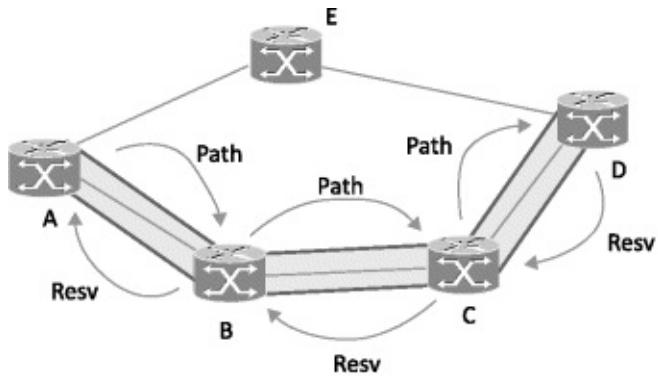
[**Figure 5.15**](#) Path Computation Element.



An LSP is set up by using a signaling protocol of the Resource reSerVation Protocol with TE extensions (RSVP-TE) [10]. RSVP-TE is an extended protocol from RSVP [11] that was originally developed to support QoS for IP flows. RSVP-TE supports the establishment of explicitly routed LSPs with resource reservation. RSVP-TE also supports smooth rerouting of LSPs, preemption, and loop detection.

[Figure 5.16](#) shows an example of the LSP set up by RSVP-TE. An ingress router determines the route on which an LSP should be set up by path computation, by the CSPF function at the router, by PCE, or through configuration by the network administrator. In [Figure 5.16](#), the explicit route of A-B-C-D is determined. To setup an LSP, a *Path* message is transmitted from the ingress router to the egress router along the determined route. After the egress router receives the *Path* message, a *Resv* message is transmitted from the egress router to the ingress router along the same route of the *Path* message. During the LSP setup procedure (i.e., in the transmission of the *Path* and *Resv* messages) at the routers along the route, the states of the LSP are created, the labels are assigned for each link, and the requested bandwidth including its priority class is reserved. The states of the LSP are maintained by periodically exchanging the *Path* and *Resv* messages, or *Hello* messages. Therefore, RSVP-TE is called a soft-state protocol. The LSP is terminated when the ingress router sends a *PathTear* message, or the periodic message exchange stops.

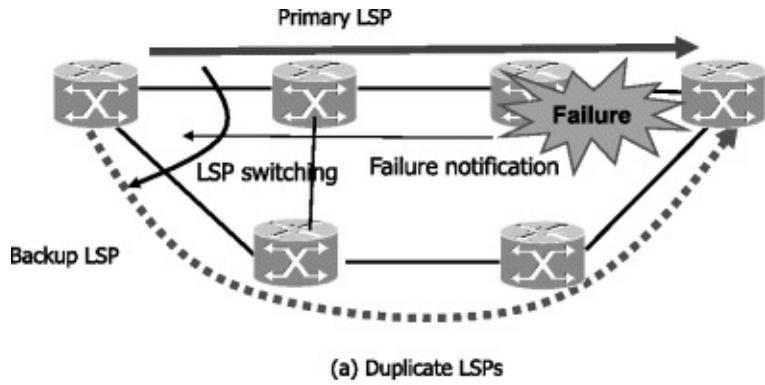
Figure 5.16 RSVP.



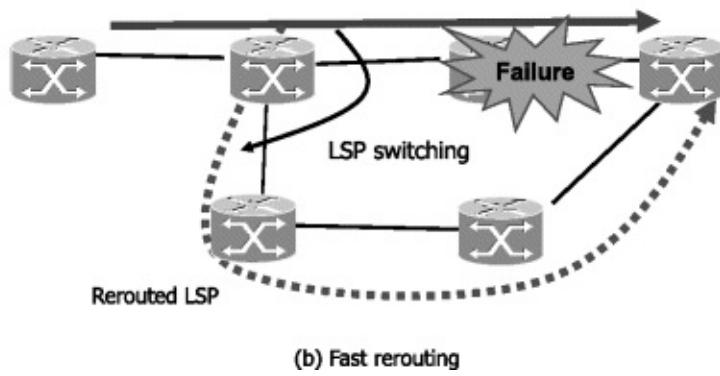
Explicit route: A-B-C-D

MPLS makes a network more resilient compared with IP networks. [Figure 5.17](#) shows two examples of the mechanisms supporting network resiliency. The first mechanism is to provide duplicate LSPs (i.e., primary and backup ones) between the ingress and egress routers, as shown in [Figure 5.17a](#). The duplicated LSPs are managed as one *tunnel* in the MPLS network [10]. The routes of the duplicated LSPs can be specified as link (node) disjoint routes and do not share any link (node) in the network. These LSPs are set up prior to a network failure. If the primary LSP is damaged by a network failure, the traffic passing through it is switched to the backup LSP at the ingress router. Therefore, the traffic is saved. From a user's point of view, the tunnel is not changed. This mechanism is simple, as only the ingress router performs the switching action after it receives the failure notification from the failure point. However, if the failure point is far from the ingress router, it takes a long time for the ingress router to switch LSPs.

[Figure 5.17](#) MPLS resilience.



(a) Duplicate LSPs



(b) Fast rerouting

To solve the problem of the duplicate LSPs, an alternative mechanism called *fast rerouting* [12], can be adopted, as shown in [Figure 5.17b](#). In contrast to the duplicate LSPs, any transit router in the fast rerouting mechanism can switch the damaged LSP to the rerouted LSP. The router that is nearest to the failure point among the upstream routers that are determined as a switching router in advance can perform the switching once it receives the failure notification. This mechanism is more complicated than the first one, but fast rerouting can be achieved.

MPLS supports Ethernet emulation from an edge to an edge in the MPLS network ([Figure 5.18](#)). The local area network (LAN) at each site is extended to the edge of the wide area network (WAN). The WAN network then provides a emulated private Ethernet link to establish a single LAN. [Figure 5.19](#) shows the layer structure for the Ethernet emulation comparing a typical structure of IP over MPLS. Ethernet frames for one LAN are counseled by MPLS packets. The counseled frames are transmitted through the MPLS WAN to another LAN.

[Figure 5.18](#) Pseudo wire emulation edge to edge.

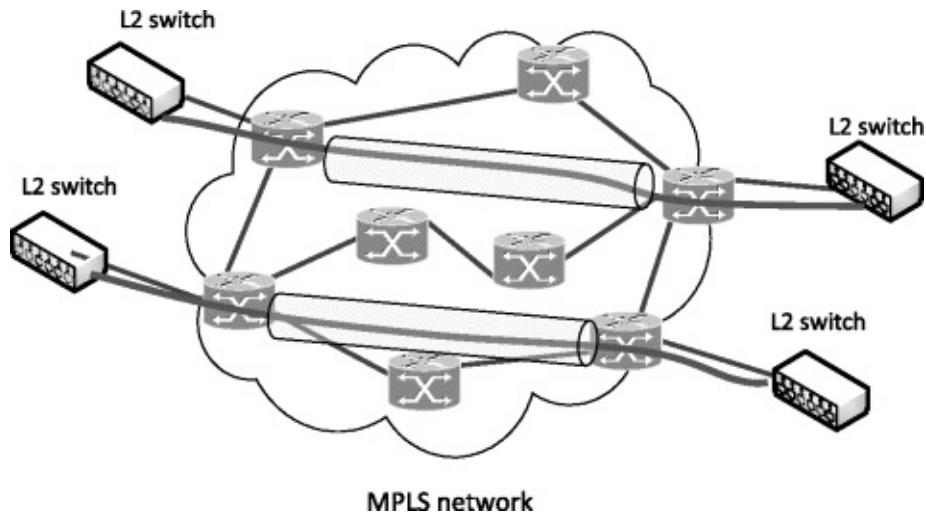
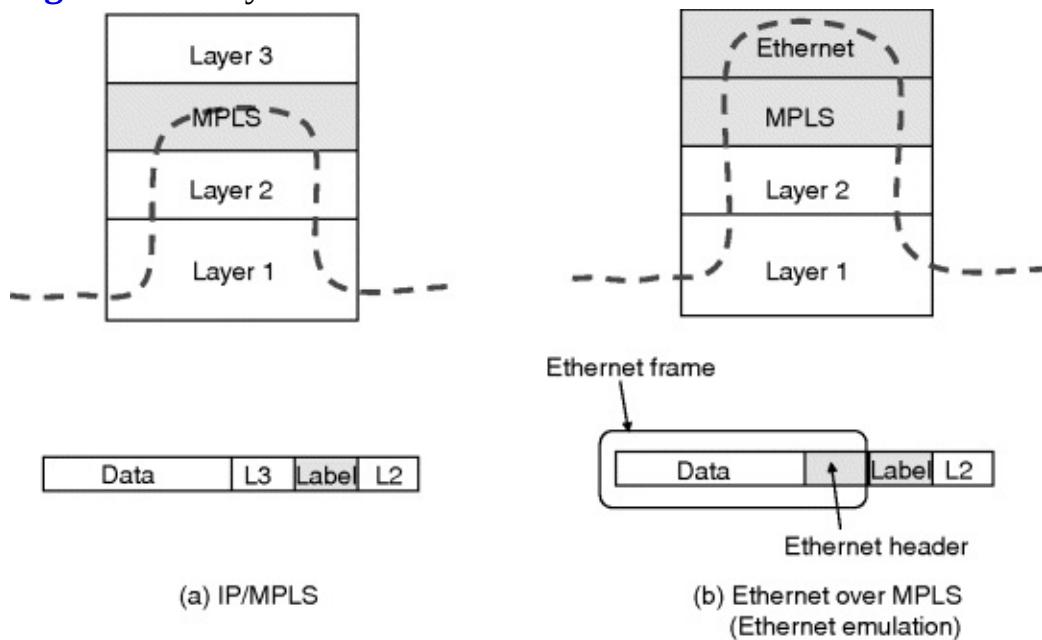


Figure 5.19 Layer structures for IP over MPLS and Ethernet emulation.



References

1. Rosen, E., Viswanathan, A. and Callon, R., "Multiprotocol Label Switching Architecture," IETF RFC 3031, Jan. 2001.
2. Moy, J., "OSPF Version 2," IETF RFC 2328, Apr. 1998.
3. Katz, D., Kompella, K. and Kompella, K., "Traffic Engineering (TE) Extensions to OSPF Version 2," IETF RFC 3630, Sep. 2003.
4. Farrel, A. and Vasseur, J.P., "A Path Computation Element (PCE)-Based

- Architecture,” IETF RFC 4655, Aug. 2006.
- 5.** Vasseur, J.P. and Le Roux, J.L. (Editors), “Path Computation Element (PCE) Communication Protocol (PCEP),” IETF RFC 5440, Mar. 2009.
- 6.** Vasseur, J.P., Zhang, R., Bitar, N. and Le Roux, J.L., “A Backward-Recursive PCE-Based Computation (BRPC) Procedure to Compute Shortest Constrained Inter-Domain Traffic Engineering Label Switched Paths,” IETF RFC 5441, Apr. 2009.
- 7.** Oki, E., Takeda, T. and Farrel, A., “Extensions to the Path Computation Element Communication Protocol (PCEP) for Route Exclusions,” IETF RFC 5521, Apr. 2009.
- 8.** Oki, E., Takeda, T., Le Roux, J.L. and Farrel, A., “Framework for PCE-Based Inter-Layer MPLS and GMPLS Traffic Engineering,” IETF RFC 5623, Sep. 2009.
- 9.** Lee, Y., Le Roux, J.L., King, D. and Oki, E., “Path Computation Element Communication Protocol (PCECP) Requirements and Protocol Extensions in Support of Global Concurrent Optimization,” IETF RFC 5557, Jul. 2009.
- 10.** Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V. and Swallow, G., “RSVP-TE: Extensions to RSVP for LSP Tunnels,” IETF RFC 3209, Dec. 2001.
- 11.** Braden, R., Zhang, L., Berson, S., Herzog, S. and Jamin, S., “Resource ReSerVation Protocol (RSVP) –Version 1, Functional Specification,” IETF RFC 2205, Sep. 1997.
- 12.** Pan, P., Swallow, G. and Atlas, A. (Editors), “Fast Reroute Extensions to RSVP-TE for LSP Tunnels,” IETF RFC 2090, May 2005.

Chapter 6

IP Quality Of Service

6.1 Introduction

The Internet was originally designed to provide best-effort service to existing applications. Nevertheless according to the original description of the Internet Protocol (IP), packets could receive different services as they pass over network nodes (now known as IP routers). Thus, packets would be able to receive, service differentiation or varying quality of service (QoS). With today's voice and video applications, real-time services are increasingly in demand, and the varying levels of service raise the need for further traffic handling differentiation. This chapter discusses different issues and solutions proposed to provide this traffic service differentiation for IP networks.

6.2 Quality of Service in IP Version 4

In IP version 4 (IPv4), a datagram header includes the type-of-service (TOS) field (8 bits) to indicate the service that the datagram may receive at supporting routers as originally defined in RFC791 1. The TOS bits provide the following choices:

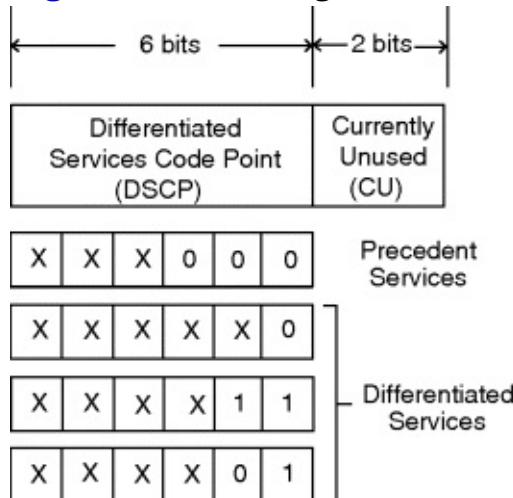
- *Bits 0–2*: Precedence (111 = Network Control, 110 = Internetwork Control, 101 = CRITIC/ECP, 100 = Flash Override, 011 = Flash, 010 = Immediate, 001 = Priority, 000 = Routine)
- *Bit 3*: 0 = Normal Delay, 1 = Low Delay 77
- *Bit 4*: 0 = Normal Throughput, 1 = High Throughput
- *Bit 5*: 0 = Normal Reliability, 1 = High Reliability
- *Bit 6*: 0 = Normal Cost, 1 = Minimize Monetary Cost (defined by RFC 1349)
- *Bit 7*: Never defined

However, the most recent update now calls this field the *differentiated services* field. The Internet Engineering Task Force (IETF) has subdivided

differentiated services into several classes of traffic. [Figure 6.1](#) shows the bits used for service differentiation. In this interpretation, the first six bits define the *codepoint* subfield, and the last two bits are not used.

- When the three right-most bits are zero, the three left-most bits are interpreted as the precedence bits in the service type interpretation. This precedence defines eight levels of priority for the datagram.
- When those right-most bits are nonzero, the six bits define 56 services based on the priority assignment from the Internet or local authorities. The first category contains 24 service types; the second and the third categories contain 16 each. These categories are assigned by the IETF, local administration, and for experimental purposes.

[Figure 6.1](#) Encoding in advanced marking scheme.



In the Differentiated Services Code Point (DSCP) notation xxxxxx, x may equal to 0 or 1. The left-most bit is numerated as bit 0 of the differentiated services (DS) field (as shown in [Figure 6.1](#)), and the right-most bit is numerated as bit 5.

The two-bit currently unused (CU) subfield is reserved. The value of the CU bits are ignored by DSCP-compliant nodes when determining the per-hop behavior to be applied to a received packet.

6.3 Integrated Services

TOS bits determine the treatment that datagrams receive from routers (and possibly other network equipment). The flows comprised by a set datagrams

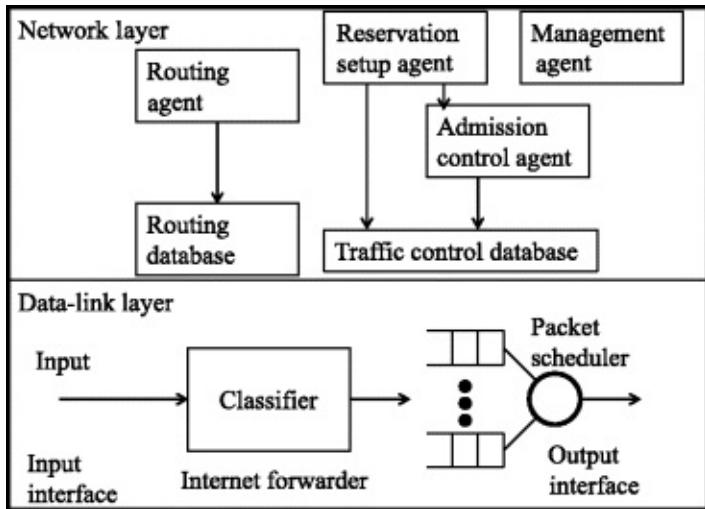
with similar features, such as those from the same source-destination hosts, are also considered in datagram treatment. Integrated services (IntServ) is a paradigm that considers the provisioning of different treatment to flows by a network (or network equipment in the network) [2]. IntServ requires identifying and mapping of flows to the required QoS parameters. Examples of QoS parameters are reliability, delay, jitter, and bandwidth.

Network operators request the ability to control the sharing of bandwidth on a particular link among different traffic classes. In this case, the traffic can be divided into a few administrative classes; a minimum percentage of the link bandwidth is assigned to each under overloading conditions, while allowing unused bandwidth to be available at other times. These classes may represent different user groups or different protocol families, for example. Such a management facility is commonly called *controlled link sharing*. With IntServ, the included service models are best-effort service (actually existing for the non-QoS services), guaranteed service, and controlled-load service.

In this service model, resources (i.e., bandwidth) must be explicitly managed in order to meet application requirements. This implies that *resource reservation* and *admission control* are required building blocks of the infrastructure. In the original proposal of IntServ, the term *guarantee* was defined broadly; the bounds can be either absolute or statistical, strict or approximate. However, defining the actual parameter is left to be the user because it must be sufficiently predictable to make the application operate in an acceptable way over the duration of time determined by the user.

The network operator can then guarantee the parameter requested by the user under specific traffic conditions that are to be defined and policed. This framework includes four components: the packet scheduler, the admission control routine, the classifier, and the Resource Reservation Protocol (RSVP). The integration of these components is depicted in [Figure 6.2](#).

Figure 6.2 Building block of the IntServ model.



6.3.1 Packet Scheduler

The packet scheduler manages the forwarding of different packet streams using a set of queues and perhaps other mechanisms, such as timers. The packet scheduler is implemented at the point where packets are queued. The details of the scheduling algorithm are used to allocate internal bandwidth; the algorithm can be engineered according to the services that are to be provided (sometimes left to the user).

6.3.2 Packet Classifier

For the purpose of traffic control (and accounting), each incoming packet must be mapped into a class; all packets of the same class receive the same treatment from the packet scheduler. If the service is to be determined by each flow requirement, the flow to which a packet belongs must be identified (or classified). Identifying and mapping the flow is performed by the classifier. Class selection may be based upon the contents of the existing packet header(s) and/or some additional classification (identifiable) information carried by each packet.

A class might include a broad category of flows (e.g., all video flows or all flows attributable to a particular organization) or only a single flow. A class is an abstraction that may be local to a particular router; the same packet may be classified differently by different routers (or networks) along the path. For example, backbone routers may choose to map many flows into a few aggregated classes, while routers nearer the periphery, where aggregation is less intense, may use a separate class for each flow.

6.3.3 Admission Control

Admission control is a decision algorithm that a router or host uses to determine whether a new flow can be granted the requested QoS without impacting earlier guarantees. Admission control is invoked at each node to make a local accept or reject decision at the time a host requests a real-time service along some path through the Internet. The admission control algorithm must be consistent with the service model and is a logical part of traffic control. It should be noted that admission control can be easily (unnecessarily) taken for policing or enforcement, which is a packet-by-packet function at the edge of the network to ensure that a host does not violate its promised traffic characteristics. In addition to ensuring that QoS guarantees are met for admitted flows, admission control is concerned with enforcing administrative policies on resource reservations. Admission control also plays an important role in accounting and administrative reporting for network provisioning (and pricing).

The forwarding path of the router is executed for every packet. The forwarding path is divided into three sections: input driver, Internet forwarder, and output driver. The Internet forwarder interprets the internetworking protocol header appropriate to the protocol suite (e.g., the IP header for TCP/IP, or the Connectionless Network Protocol [CLNP] header for Open Systems Interconnection [OSI]. For each packet, an Internet forwarder executes a suite-dependent classifier and then passes the packet and its class to the appropriate output driver.

6.3.4 Resource Reservation Protocol (RSVP)

A reservation protocol, such as RSVP, is necessary to create and maintain a flow-specific state in the endpoint hosts and in routers along the path of a flow. In other words, RSVP is a signalling protocol to communicate the application's requirements to network elements along the path and to convey QoS management information between network elements. RSVP is defined in RFC2205 [3] and its use in RFC2210 [4].

In order to state its resource requirements, an application must specify the desired QoS using a list of parameters called a *flowspec* [3]. The flowspec is carried by the RSVP, passed to admission control to test for acceptability, and used to parameterize the packet scheduling mechanism.

Because RSVP is designed to be used with a variety of QoS control

services, and because the QoS control services are designed to be used with a variety of setup mechanisms, a logical separation exists between the two specifications. The RSVP specification does not define the internal format of those RSVP protocol fields, or objects, that are related to invoking QoS control services. The objects can carry different information to meet different application and QoS control service requirements.

RSVP requests resources for simplex flows (i.e., it requests resources in only one direction). Therefore, RSVP treats a sender as logically distinct from a receiver, although the same application process may act as both a sender and a receiver at the same time. RSVP operates on top of IPv4 or IPv6, occupying the place of a transport protocol in the protocol stack. However, RSVP does not transport application data but is rather an Internet control protocol, like Internet Control Message Protocol (ICMP), Internet Group Management Protocol (IGMP), or routing protocols.

RSVP is designed to operate with unicast and multicast routing protocols. An RSVP process consults the local routing database(s) to obtain routes. In the multicast case, for example, a host sends IGMP messages to join a multicast group and then sends RSVP messages to reserve resources along the delivery path(s) of that group. Routing protocols determine where packets get forwarded; RSVP is only concerned with the QoS of those packets that are forwarded in accordance with routing.

In order to efficiently accommodate large groups, dynamic group membership, and heterogeneous receiver requirements, RSVP makes receivers responsible for requesting a specific QoS. The RSVP protocol then carries the request to all the nodes (routers and hosts) along the reverse data path(s) to the data source(s), but only as far as the router where the receiver's data path joins the multicast distribution tree.

During reservation setup, an RSVP QoS request is passed to two local decision modules: admission control and policy control. Admission control determines whether the node has sufficient available resources to supply the requested QoS. Policy control determines whether the user has administrative permission to make the reservation. If both checks succeed, parameters are set in the packet classifier and in the link-layer interface (e.g., in the packet scheduler) to obtain the desired QoS. If either check fails, the RSVP program returns an error notification to the application process that originated the request.

Because the membership of a large multicast group and the resulting

multicast tree topology are likely to change with time, the RSVP design assumes that the RSVP and traffic control states are to be built and destroyed incrementally in routers and hosts. For this purpose, RSVP establishes a *soft state*, that is, RSVP sends periodic refresh messages to maintain the state along the reserved path(s). In the absence of refresh messages, the state automatically times out and is deleted.

An elementary RSVP reservation request consists of a *flowspec* together with a *filter spec*; this pair is called a *flow descriptor*. The *flowspec* specifies a desired QoS. The filter spec along with a session specification define the set of data packets (i.e., the flow) to receive the QoS determined by the *flowspec*. The *flowspec* is used to set parameters in the node's packet scheduler or other link-layer mechanism, while the filter spec is used to set parameters in the packet classifier. Data packets that are addressed to a particular session but do not match any of the filter specs for that session are handled as best-effort traffic.

The *flowspec* in a reservation request will generally include a service class and two sets of numeric parameters: (1) an *Rspec* (R for reserve) that defines the desired QoS, and (2) a *Tspec* (T for traffic) that describes the data flow. The formats and contents of *Tspecs* and *Rspecs* are determined by the integrated service models (RFC 2210) and are generally opaque to RSVP.

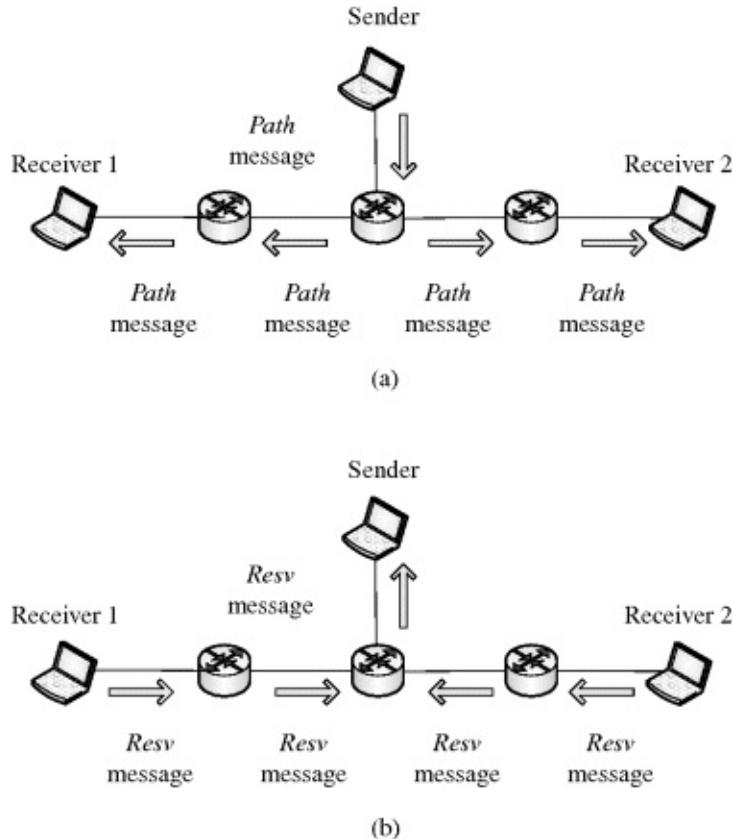
For simplicity, the basic filter spec format defined in the present RSVP specification has a very restricted form: the sender's IP address and optionally the UDP/TCP port number from the *SrcPort* field. *TSpec* is carried from the sender to intermediate network elements and the receiver(s) by RSVP, but is never modified by intermediate elements within the network.

There are two fundamental RSVP message types: *Path* collects the state information from senders to receivers, and *Resv* reserves the path. [Figure 6.3](#) shows an example of *Path* and *Resv* messages in RSVP.

- Each RSVP sender host transmits RSVP *Path* messages downstream along the unicast or multicast routes provided by the routing protocol(s), following the paths the data may follow. These *Path* messages store the *path state* in each node along the way. This *path state* includes at least the unicast IP address of the previous hop node, which is used to route the *Resv* messages hop-by-hop in the reverse direction ([Figure 6.3a](#)).
- Each receiver host sends *Resv* messages upstream toward the selected senders. These messages follow the reverse of the path(s)

the data packets will use. The *Resv* packets create and maintain the reservation state in each node along the path(s) ([Figure 6.3b](#)).

Figure 6.3 Sending (a) *Path* and (b) *Resv* messages in RSVP.



6.4 Differentiated Services

As routers at the core handle many aggregated flows, it is possible to find a significant number of groups of flows that have similar requirements. These flows may be aggregated to decrease the number of states to be maintained, similar to per-flow management performed by IntServ. As discussed earlier. Through RSVP, IntServ allows sources and receivers to exchange signaling messages that establish additional packet classification and forwarding state on each node along their path. In the absence of state aggregation, the amount of state on each node scales in proportion to the number of concurrent reservations, and this can be large on high-speed links.

A solution to this, the Differentiated Services (DiffServ or DS) service model can be adopted in the core of the network to aggregate the states of different flows [5]. DiffServ is based on a simple model where traffic

entering a network is classified, possibly conditioned, and then assigned to different behavior aggregates. Each behavior aggregate is identified by a single DS codepoint (discussed at the beginning of this chapter). Within the core of the network, packets are forwarded according to the per-hop behavior associated with the DS codepoint.

Each packet carries a DS field to identify the behavior aggregate assigned to the flow owning the packet. This is the field originally known as TOS in IPv4 (and referred to as DS field in IPv6), as shown in [Figure 6.1](#). The DS field is comprised of the DSCP and the two CU subfields. The DSCP subfield defines the *Per-Hop Behavior* (PHB) that a flow is aggregated into. With this field, a DiffServ-capable router (network node) uses the DSCP subfield as an index to a table that defines the management service the carrying packet is to receive.

The PHB is classified as follows:

- Default PHB (DE PHB); This is a best-effort service that the Internet provides by default and is compatible with the original TOS field.
- Expedited Forwarding PHB (EF PHB): This PHB provides the following services:
 - Low loss
 - Low latency
 - Ensured bandwidth.
- Assured Forwarding PHB (AF PHB): This PHB has the objective to deliver a packet with high assurance as long as the class traffic does not exceed the traffic profile of the node. However, a packet might still be discarded.

To support such PHB, traffic may need to be conditioned. DiffServ considers traffic conditioners such as meters, markers, shapers, and droppers. These are briefly defined as follows:

- *Meter* verifies if the incoming flow matches the negotiated traffic profile to access a PHB. The meter can use several methods to assert this determination. An example is the token bucket technique.
- *Marker* marks a packet that is detected as using best-effort delivery (DSCP: 00000) or down marks a packet based on information received by the meter (e.g., the packet of a flow overusing the negotiated profile).

- *Shaper* uses the information received from the meter to re-shape the traffic if it is not compliant with the negotiated profile.
- *Dropper* discards the packets of a flow that severely violates the negotiated profile.

6.5 Quality of Service with Nested Differentiated Services Levels

IntServ and DiffServ are well-known alternatives for QoS provisioning. These two paradigms differ in level of accuracy on service provisioning and QoS granularity for scalable implementation. Therefore neither one can satisfy a large number of service requirements. Whether the IntServ over DiffServ model can fill this void remains an open issue, and so it is necessary to map each individual flow's end-to-end QoS requirements from the IntServ model to the DiffServ model. Hence, flows mapped to a single service class get similar end-to-end QoS guarantees, even when the requirements of each individual flow differ. Therefore, QoS granularity is decreased, depriving user' benefits and lowering network utilization.

Implementing a nested DiffServ model is one way to solve this problem; each group of flows can have a subset of requirements. This model can be combined with the explicit endpoint admission control (EEAC) scheme that represents the nested-DiffServ service levels as service vectors (SV) [6]. Consider n service classes $S = (S_0, S_1, \dots, S_{n-1})$ provided by each link in a network. One flow, going from the source to the destination via m nodes or $m - 1$ links may choose service s_i ($s_i \in S$) at router i . The service at s_i , may be different from service s_j selected by router j . The service vector hereby is defined as $s = (s_0, s_1, \dots, s_{m-1}, s_m)$. The aim of service vectors is to find the suitable service classes in a single path so as to maximize

$$(6.1) \quad G = \max(U - C)$$

where U is the utility function and C is the cost. This combination of service vectors decouples the provisioning of end-to-end QoS at each router, thus resulting in an intermediate level of granularity and complexity between per-flow and per-group levels. The EEAC scheme can be performed in two phases: the probing (or exploring) phase to determine the link state [7], and the data transmission phase, which is performed after the probing (and call

acceptance) processes. In the probing phase, the end host sends probing packets to the destination host to collect the SV information, which includes the service states of the routers along the end-to-end path. After receiving feedback from the end server and retrieving the state from the probing packets, the end host compares all possible service class combinations for this specific path and computes the utilization and cost to find the most suitable service classes to be used at each router per flow basis. The selected service vector is marked in the data packets during the data transmission phase. Each router checks the vector and provides the cost of the corresponding QoS service in it. This EEAC-SV model improves the QoS granularity to $O(p^q)$, where p is the number of routers and q is the number of service classes in the network (or end-to-end path). The flexibility feature increases the probability of minimizing the cost for the user and network utilization for the service provider. However, this scheme, similar to other endpoint admission control models, assumes that the path is preselected so that the probing path and data transmission path are always fixed. This assumption simplifies the analysis, but it may not be accurate in considering a real network. In routing mechanisms, the above QoS provisioning scheme may not be able to provide the flexibility achievable by EEAC if SVs are not considered in the path calculation.

In the remainder of this discussion, Open Shortest Path First (OSPF) [8] is considered as the most widely used QoS routing model. In some cases, a misused OSPF may cause false routing, which results in low utilization of the network resources and high cost. To solve this problem, OSPF is combined with SVs during the path selection phase [9]. Because a link is the connection between two routers, the different service classes available for a link can be detected by the neighboring routers and disseminated by OSPF in a timely fashion.

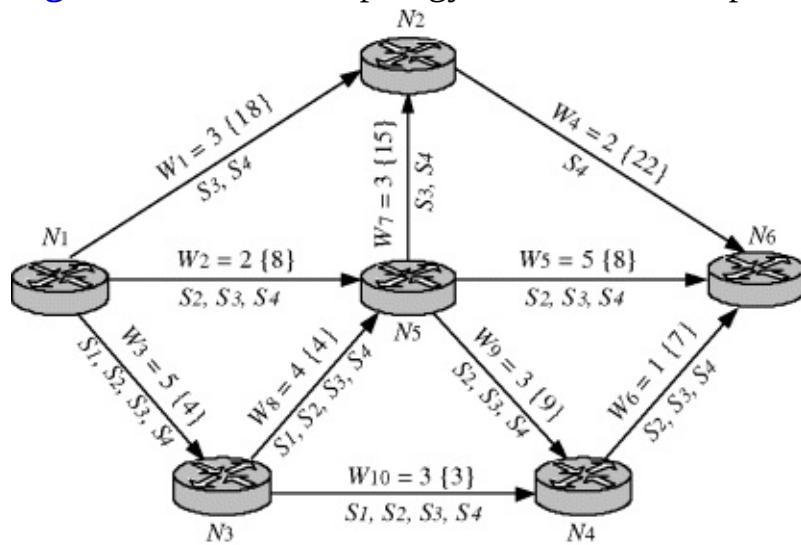
6.5.1 Drawbacks of Explicit Endpoint Admission Control with Path Selection

Without loss of generality, OSPF can be used to select a path for the EEAC scheme. At present, however, some vendors simply set the weight of links in OSPF to be inversely proportional to the capacity of the link. This configuration cannot reflect the accurate QoS state of the network. Some other setting methods reported in the literature [10, 11] are weights mapped

to the combination of QoS parameters like delay and available bandwidth. However, there is currently no prominent definition of weight setting according to QoS because the multi-dimensional feature of QoS results in complex weight setting. [Figure 6.4](#) shows an example of a simple network with routers $N = (N_1, N_2, N_3, N_4, N_5, N_6)$. Assume the delay of the path is determined by the QoS requirement. The service class $S = (S_1, S_2, S_3, S_4)$ is thus categorized by the delay of the link as shown below. The cost C of the service is represented as:

Here, $W_i, i \in (1, 10)$, as shown in the graph, is the weight of each link recorded by OSPF. Now let us assume that a flow from N_1 to N_6 has a delay request of less than 25 ms. OSPF will select the shortest path as $P_1 = (N_1, N_2, N_6)$ according to the addition of weights on the path. The EEAC-SV scheme then executes the probing processes along P_1 , but no SV may satisfy the delay request for less than 25 ms, thus the request is denied. However, the rest of the paths, such as $P_2 = (N_1, N_5, N_6)$ and $P_3 = (N_1, N_3, N_5, N_6)$, can satisfy the user's request with the proper service class selection. Furthermore, it is easy to see that the SV (S_2, S_2) with P_2 and the lowest cost ($C = 3 + 3 = 6$) is the optimal solution. Therefore, the EEAC-SV scheme suffers from false routing in this case.

[Figure 6.4](#) Network topology of the first example.



As another example, set the link weight as the function of delay, as shown in brackets in [Figure 6.4](#), or:

$$W_i = f(\text{delay}) = [\text{delay}] .$$

Then $P_4 = (N_1, N_3, N_4, N_6)$ is the shortest path found by OSPF, which makes EEAC select (S_1, S_2, S_2) or (S_2, S_1, S_2) as the solution (where the cost is 12). However, the optimal answer in this case is $P_2 = (N_1, N_5, N_6)$ with service (S_2, S_2) in tandem (where cost is 6). The nonoptimal solution increases the cost and diminishes the network utilization.

6.5.2 OSPF-Based Adaptive and Flexible Quality of Service Provisioning

To overcome the above problem, a new architecture based on OSPF was proposed as a combination of OSPF and SV [9]. In the following description, the weight of link i (W_i) is represented as a vector that contains all the available service classes the link can provide, documented as:

$$(6.2) \quad W_i = (S_1, S_2, \dots, S_k) \quad s.t. \quad (S_1, S_2, \dots, S_k) \in S$$

Generally, the service class in a QoS model is defined by the function of various QoS parameters. We note the service class as $S_i = (Q_i^1, Q_i^2, \dots, Q_i^k)$, where $Q_i^j, j \in \{1, \dots, k\}$ is the j^{th} QoS component in the i^{th} service class. For instance, a service class may be defined as delay, jitter, packet loss, or available bandwidth). In this way, different service classes share the common network resource so they can be compared by the QoS parameters and sorted in a linear order. In our model, only the highest available service class needs to be marked as the weight of the link. However, the services of the lower classes can also be provided. Here, the amount of data used to represent the state class is reduced, that is, the overhead of the link-state update is decreased. Similar to the original OSPF protocol, the link states are exchanged by link-state advertisement (LSA) packets among the routers in the network. This ensures that every router has the same QoS link-state database. When the user's request comes, the edge router (i.e., source node) selects the shortest path that satisfies [Equation 6.1](#). All the service classes lower than or equal to the weight of the link are candidates for selection. The edge router here is different than in

generic QoS routing; the router not only selects the path the data goes through but also the service classes of the link as requested. If the user's requirement can be satisfied, the SV as the selected service classes are marked in the data packets during data transmission. The traversed routers read the service class from each packet and provide the corresponding service. If no SV can be found to fulfill user demand, the request is denied.

As for the link-state update, the router estimates the state of the connected link and launches the state update mechanism when any of the states change. However, the estimation of the performance of each service cannot be accurate because if the state is updated too frequently these updates generate large traffic overhead. Besides, to prevent the LSA packets and the following data packets from increasing their overhead, the values of the QoS parameters in each service class are classified into several service levels, so that $(\log_2 M)$ bits can represent M service classes.

OSPF sets a link to disseminate its state information every 30 minutes. This update interval may be too large for the architecture if high data rate flows are considered because the dynamic changes of the QoS parameters may cause the state already known to other routers to become outdated. To overcome this problem, the update is triggered when the state of a link crosses a service level boundary, which is called a *class-based triggering mechanism* [12].

6.5.3 Combination of Security and Quality of Service

It is well known that the security level of a given communication depends on the individual user. Therefore, it is difficult to evaluate security uniformly and classify its service level. In order to shape the problem, the security protection capability of the router can be estimated and linearly mapped to the level of security satisfaction of users. The most widely considered security capabilities of the router can be listed as encryption, denial-of-service (DoS) detection, authentication, and virus filtering. Besides these, more security components can be extended in this framework. Thus, the following criteria were created to evaluate them, but other standards can be equally applicable:

- *Encryption K_e* : Measured by the bit length of the encryption key

(e.g., 64 bits, 128 bits) and strength of cryptographic algorithm (e.g., RSA, DES).

- *Virus scanning K_v* : Measured by the number of viruses and worms that the anti-virus software can detect and the false-alarm ratio.
- *DoS detection K_d* : Measured by the false-positive ratio of intrusion detection system (IDS) under uniform DoS attack testing.
- *Authentication K_a* : Measured according to the robustness of the authentication mechanism, and might contain a weak or strong password, biometric, and smart cards with on-board display and input interfaces.

Here, link security state is expressed as a vector K_a, K_v, K_d, K_a . From the user's viewpoint, the security of link i means adding the above four components:

$$(6.3) \quad K_i = a_1 K_e^i + a_2 K_v^i + a_3 K_d^i + a_4 K_a^i$$

Let us assume there are n links in the path under study, and that the security level of the path is the link with minimum security:

$$(6.4) \quad K_p = \min(K_1, K_2, \dots, K_n)$$

$a_i, i \in (1, 2, 3, 4)$ is the sensitivity weight of the individual security component, as a user is concerned about the different security components that are specific to a given situation. The security protection capability mentioned here is expected to change at a lower rate than other QoS parameters, so the security values are updated with a low frequency (e.g., once every 24 hours). During each update interval, this value is considered to remain constant in each service class. This decreases the computation work of routers and produces no increase in the complexity of the link-state update.

As mentioned above, [Equation 6.1](#) is utilized for service vector selection. The cost of the security service S_i in each router is related to the processor occupancy time and strength level $C_p(S_i)$, the occupied memory $C_m(S_i)$, the bandwidth $C_b(S_i)$, and the disk space $C_d(S_i)$ for a database to store virus or DoS attack patterns. The cost function of the security service is the sum of all of them:

$$(6.5) \quad C_{\text{security}}(S_i) = C_p(S_i) + C_m(S_i) + C_b(S_i) + C_d(S_i).$$

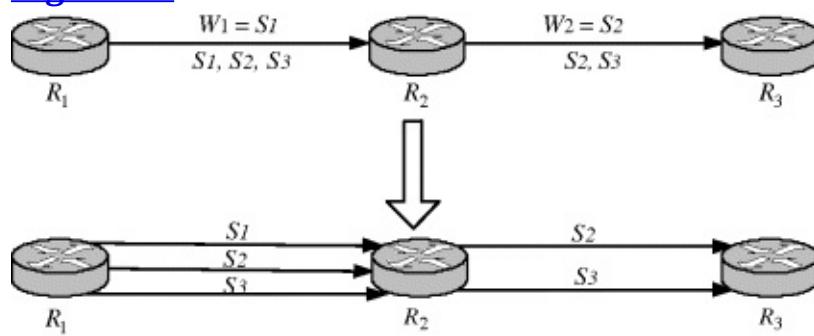
6.5.4 Path Selection Algorithm Analysis

Generally data flows can specify their QoS requirements in terms of four parameters: the available bandwidth B_{req} , the maximum jitter request J_{req} , the maximum delay request D_{req} , and the minimum security requirement K_{req} . The path and SV selection problem thus can be described as the problem to maximize [Equation 6.1](#) as long as it is eligible for the selected path p_j ,

$$\begin{aligned} B_p(P_j) &\geq B_{req}, \quad D_p(P_j) \leq D_{req} \\ (6.6) \quad J_p(P_j) &\leq J_{req}, \quad K_p(P_j) \geq K_{req} \end{aligned}$$

From the user's perspective, the utility function U reflects the degree of user satisfaction with the QoS service. A user's QoS requirement can be elastic or inelastic. With elastic demand the user can tolerate some degree of service deterioration if QoS provisioning is lower than the user's expected constraint; while inelastic means otherwise. In this discussion, inelastic QoS requirements such that U is either 1 or 0 are considered. Maximizing [Equation 6.1](#) is equivalent to minimize cost function C (i.e., the multi-constrained lowest-cost routing with multiservice is selectable), which is an NP-complete problem [13] First, let us consider the several service classes in each link. To convert their multiple-to-one relationship to one-to-one mapping, each service class is regarded as a virtual link, as [Figure 6.5](#) shows.

[Figure 6.5](#) The illustration of virtual links with service classes.



To simplify the above problem, user constraints are categorized into two different classes, and the algorithm is used to analyze each type of constraint. The first class called concave or bottleneck constraint includes cases about available bandwidth and security. The cases can be solved by using an extension of the Dijkstra algorithm, as in [Figure 6.6](#). Assume a directed graph

$G = (V, E)$, where V is the set of nodes and E is the set of links. s and d are the source node and destination node, respectively. For the m th service class and $|E|$ links in the graph G , the time complexity of the preprocess part is $O(m |E|)$; the time complexity of the main selection part is $O(n^2)$. Thereby the total complexity of this algorithm is $O(n^2)$, which has the same order of complexity as the original Dijkstra algorithm.

Figure 6.6 The path selection algorithm for concave constraint.

QoS-Routing (G, s, d, K_{req})

Preprocess Part

for each $e \in E$

$temp = \phi$

 Set m service classes

for each $k \in m$

if $K_{e^k} \geq K_{req}$ **then**

$temp := temp \cup \{k\}$

 Recode virtual links with lowest service class

$E := E - \{e^k\}, \forall k \in m$ except $k = \min \{temp\}$

Main Selection

Dijkstra(E, s, d)

The other class is called additive constraint and includes cases about delay and jitter. Assume the set of feasible paths from node s to d is F . The cost function is C . Thus the problem is defined as:

$$C = \min \{F\}$$

$$s.t. \forall p_j \in F, D_{p_j} \leq D_{req} \text{ (or } J_{p_j} \leq J_{req})$$

The delay-constrained-least-cost (DCLC) problem is about determining the path with the lowest cost in a delay-constrained environment. Many mechanisms are proposed to solve the problem in polynomial time, among which k shortest paths (KSP) is a good solution. The KSP scheme is based on Jimenez and Marzal's Recursive Enumeration Algorithm (REA) [14]. The idea is to list k shortest paths from s to d with increasing costs of weight in a

directed graph. The algorithm first invokes Dijkstra's shortest path algorithm to build the shortest path tree. Each path from s to current node v is the concatenation of the path from s to $\text{pre}(v)$ and the link $(\text{pre}(v), v)$, while $\text{pre}(v)$ is the adjacent predecessor node. The k_{th} shortest path $\pi^K(v)$ is thus selected from the candidate set $C^K(v)$ according to the following generalized Bellman's equation.

$$C^k(v) = \begin{cases} \pi^1(u) \cdot v \forall u \in \text{pre}(v) & \text{if } k = 1 \& v \neq s, \\ & \text{or } k = 2 \& v = s; \\ (C^{k-1}(v) - \pi^g(u) \cdot v) \cup \pi^{g+1}(u) \cdot v & \text{otherwise;} \end{cases}$$

$$\pi^k(v) = \begin{cases} s & \text{if } k = 1 \& v = s; \\ \text{argmin}_{\pi \in C^k(v)} L(\pi) & \text{otherwise;} \end{cases}$$

where $\pi^{k-1}(u) = \pi^g(u) \cdot v$ and $L(\pi)$ is the weight of the path π .

The above recursive computation to obtain the k shortest paths solution is finished in $O(m + Kn \log (m/n))$ time. To avoid the worst case when K is large, the following algorithm was proposed for the DCLC problem [9]. Once the delay constraint is above the threshold, the k th longest path is found based on the longest path tree instead of the shortest one.

The algorithm in [Figure 6.7](#) was simulated in the 32-node bidirectional network by running 10,000 requests [15]. Each link is replaced by three virtual links to represent the service classes. Without loss of generality, the delay of the virtual link is uniformly distributed from 1 to 500. The source and destination node is 1 and 30. Here, h is set to 0.5. The delay constraint set is from 150 to 1500 with 50 between intervals. [Figure 6.8](#) shows the average number of iterations k , where k is found without setting the upper bound of k and considering that there are optimal feasible paths. The figure shows that the number of iterations is not large and does not increase without limit when the delay constraint increases. In reality, as the service class is distributed uniformly among links, the variety of construction paths decreases. This means that k is actually smaller than shown in the figure.

[Figure 6.7](#) The path selection algorithm for additive constraint.

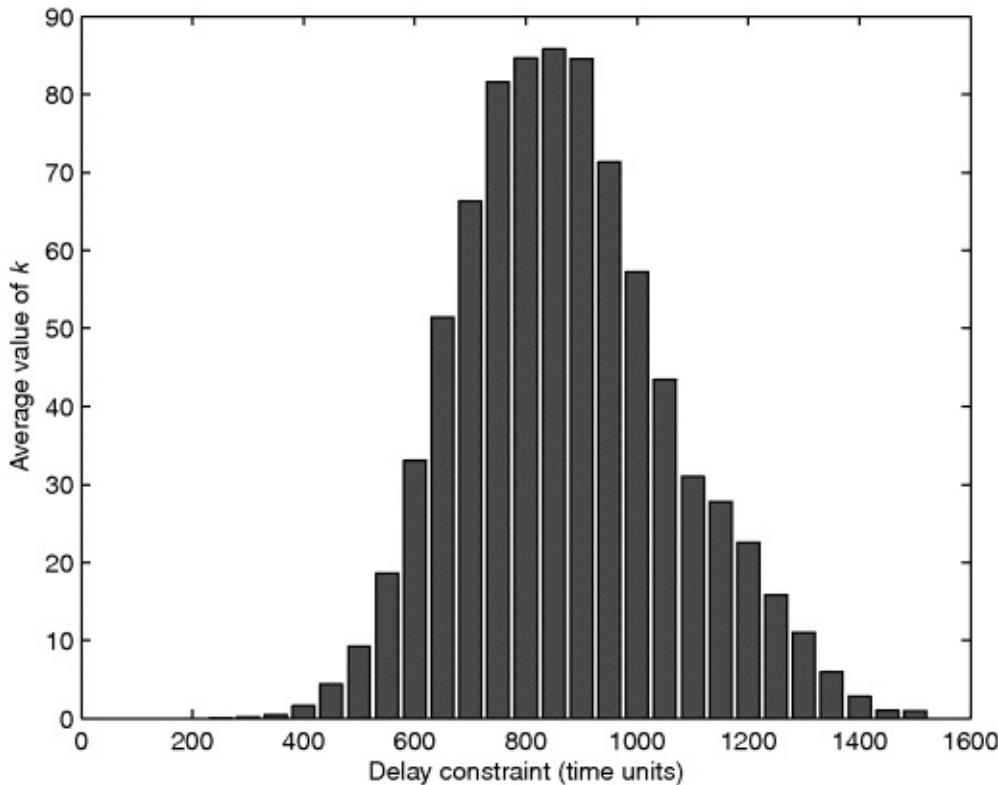
Build Shortest and Longest Path Tree

```
pathshortest = Dijkstra(V, E);
pathlongest = Dijkstra(V, E);
threshold = h * (Delay(pathshortest)
                  + Delay(pathlongest));
```

k Paths Selection

```
if Delayconstraint < Delay(pathshortest) then
    Request is denied
else if Delayconstraint ≤ threshold
    invoke k shortest path algorithm
else if Delayconstraint ≥ threshold
    invoke k longest path algorithm
cost = min(cost(πi(d)), ∀i ∈ k)
```

Figure 6.8 Average number of iterations of the path selection algorithm in 32-node network.



References

- 1.** Information Sciences Institute, University of Southern California, “Internet Protocol,” IETF RFC791 Sep. 1981.
- 2.** Braden, R., Clark, D. and Shenker, S., “Integrated Services in the Internet Architecture: An Overview,” IETF RFC1633, Jun. 1994.
- 3.** Braden, R., Zhang, L., Berson, S., Herzog, S. and Jamin, S., “Resource ReSerVation Protocol (RSVP),” IETF RFC2205, Sep. 1997.
- 4.** Wroclawski, J., “The Use of RSVP with IETF Integrated Services,” IETF RFC2210, Sep. 1997.
- 5.** Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z. and Weiss, W., “An Architecture for Differentiated Services,” IETF RFC2475, Dec. 1998.
- 6.** Yang, J., Ye, J. and Papavassiliou, S., “A Flexible and Distributed Architecture for Adaptive End-to-End QoS Provisioning in Next Generation Networks,” IEEE JSAC, vol. 23, no. 2, pp. 321-333, Feb. 2005.
- 7.** Qin, Z., Rojas-Cessa, R. and Ansari, N., “Distributed Link-State Measurement for QoS Routing,” Proc. IEEE Military Communications (MILCOM 2006), Washington, DC, p. 5, Oct. 23-25, 2006.
- 8.** Moy, J., “OSPF version 2,” IETF RFC2328, Apr. 1998.
- 9.** Qin, Z., Rojas-Cessa, R. and Ansari, N., “OSPF-Based Adapted and Flexible Security-Enhanced QoS Provisioning,” Proc. IEEE Sarnoff Symposium 2006, Princeton NJ, pp. 5, 26-28, March 2006.
- 10.** Apostolopoulos, G., Williams, D., Kamat, S., Guerin, R., Orda, A. and Przygienda, T., et al., “QoS Routing Mechanisms and OSPF Extensions,” IETF RFC 2676, Aug. 1999.
- 11.** Fortz, B. and Thorup, M., “Optimizing OSPF/IS-IS Weights in a Changing World,” IEEE JSAC, vol. 20, no. 4, pp. 756-767, May 2002.
- 12.** Apostolopoulos, G., Guérin, R., Kamat, S. and Tripathi, S.K., et al., “Quality-of-service based routing: A Performance Perspective,” ACM SIGCOMM Computer Communication Review, vol. 28, no. 4, pp. 17-28, Oct. 1998.
- 13.** Wang, Z. and Crowcroft, J., “Quality of Service Routing for Supporting Multimedia Applications,” IEEE JSAC, vol. 14, no. 7, pp. 1228-1234, Sep.

1996.

- 14.** Jimenez, V. and Marzal, A., “Computing the k Shortest Paths: A New Algorithm and an Experimental Comparison,” *Lecture Notes in Computer Science*: 1668, pp. 15-29, 1999.
- 15.** Chen, S. and Nahrsted, K., “On Finding Multi-Constrained Paths,” *IEEE ICC*, vol. 2, pp. 874-899, June 1996.

Chapter 7

IP Multicast and Anycast

The majority of the traffic on the Internet is *unicast*: one source device sending data to one destination device. However, it is possible to have one source device sending the same data to a group of devices; this is called *multicasting* or a *multicast* service. The Internet Protocol (IP) also supports *multicasting*. This service type can be considered a more efficient alternative to broadcasting because only selected hosts can be designated receivers.

7.1 Addressing

7.1.1 Multicast Addressing

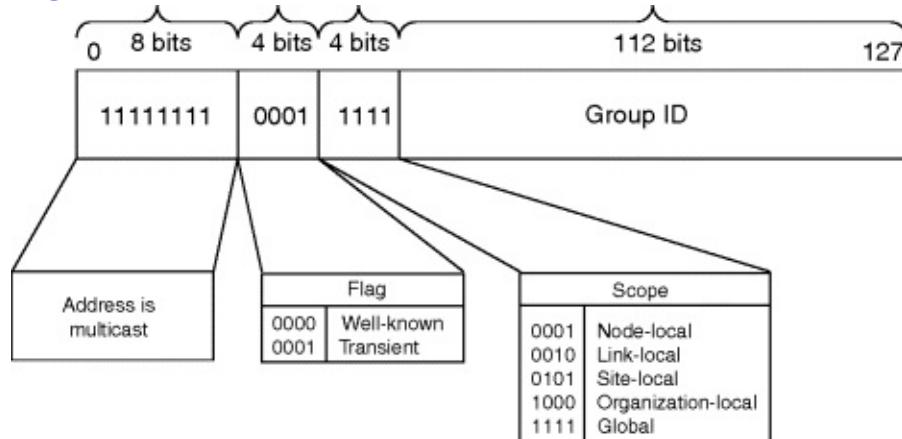
The classful IP addressing scheme in IP version 4 (IPv4) sets aside 1/16 of the address space for multicast addresses; this group of addresses is known as Class D. Multicast addresses are identified by the pattern 1110 in the first four bits, which corresponds to a first octet of between 224 and 239. The full range of multicast addresses is from 224.0.0.0 to 239.255.255.255. Since multicast addresses represent a group of IP hosts (or interfaces), which is sometimes called a host group, they can only be used as the destination of a multicast datagram and never as the source.

Under IPv4, this block of addresses is defined with this objective and covers several multicast services and applications [1]. This large multicast class is subdivided into blocks according to how far the packets are allowed to travel and the type of services provided. The address can be combined with specific values of the time-to-live (TTL) to limit the distance (in the number of routers) that specific multicast packets travel. An example of a multicast service using this address block is the Network Time Protocol (NTP) [2, 3], which has the address 224.0.0.1.

Under IPv6, multicast addresses are allocated from a defined multicast block. This is 1/256 of the address space, consisting of all addresses that begin with 11111111 in the binary notation or FF in hexadecimal notation, as

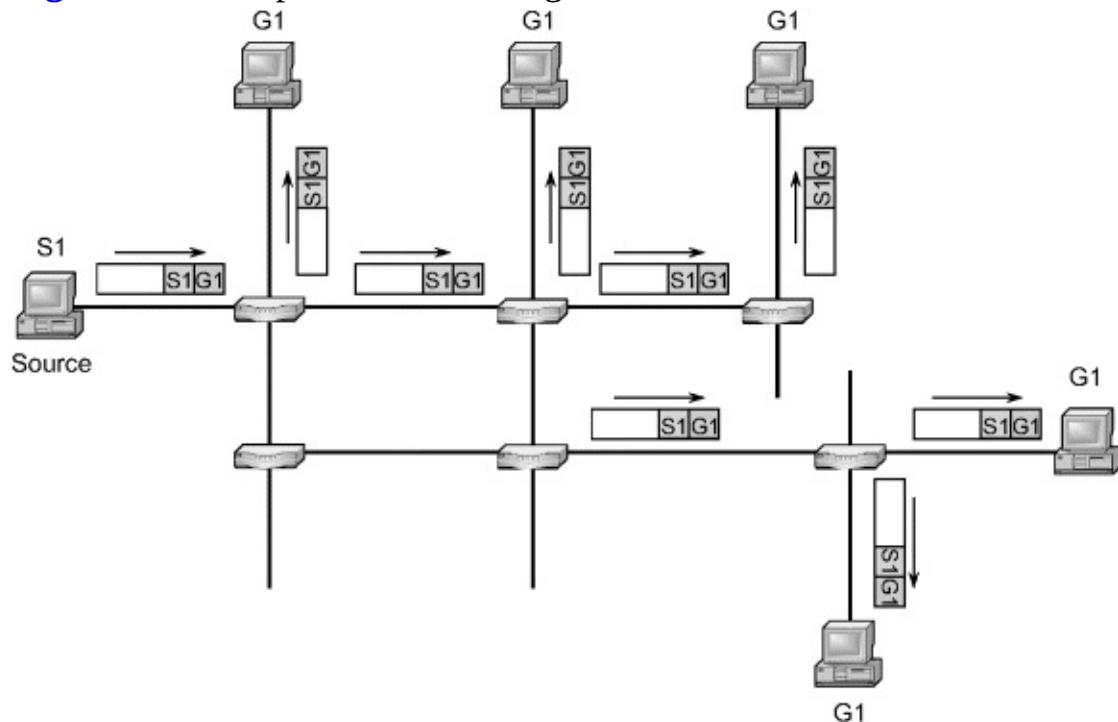
shown in [Figure 7.1](#).

Figure 7.1 Format of IPv6 multicast address.



Multicast services enable a network to communicate to multiple destinations using a smaller number of packets, therefore using network bandwidth more efficiently. [Figure 7.2](#) shows an example of multicasting. In this example, a multicast packet starts from the source S1 and goes to all destinations that belong to group G1. In multicasting, when a router receives a packet, it may forward the packet through several of its interfaces.

Figure 7.2 Example of multicasting.



In broadcast communication, the relationship between the source and

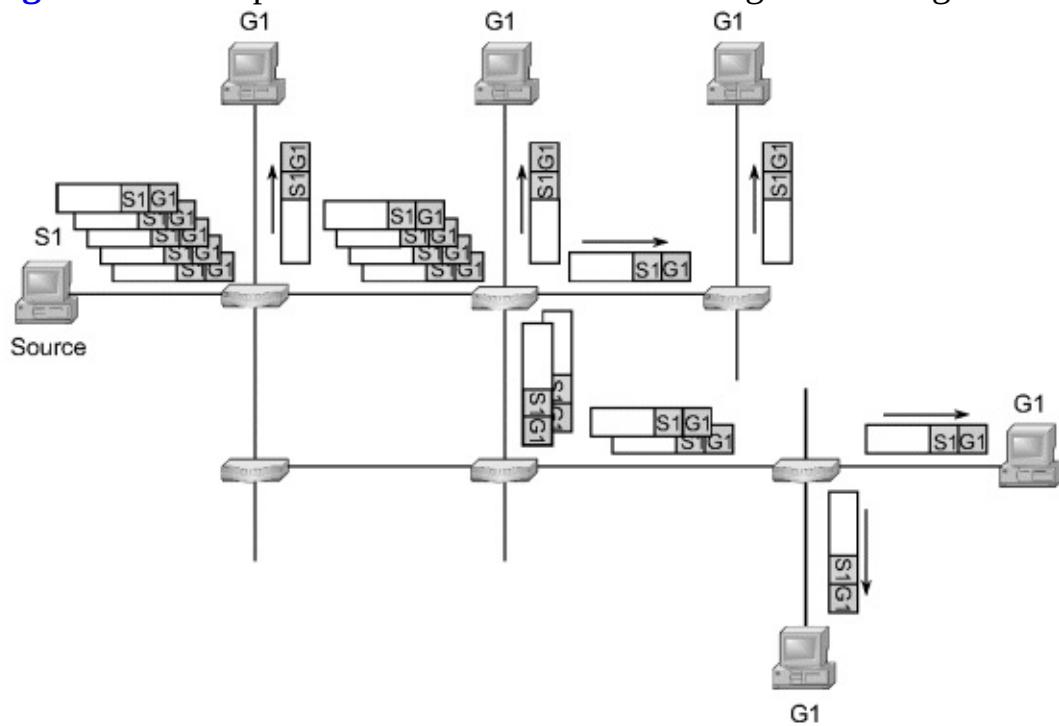
destination is from one to all; there is a single source but all hosts are the destinations.

7.1.2 Differences between Multicasting and Multiple Unicasting

Multicasting starts with a single packet generated from the source that is replicated by the routers that support multicasting. The destination address in each packet is the same for all replicated packets. The motivation of using multicasting is higher efficiency as one single copy of the packet travels between any two routers.

In multiple unicasting, as shown in [Figure 7.3](#), several packets are sent from the source. For example, if there are five destinations, the source sends five packets and each has a different unicast destination address. In this case, multiple copies might be transmitted between two routers. A classical example of multicasting emulation is a person sending an e-mail message to a group of people, i.e., multiple unicasting.

[Figure 7.3](#) Example of multicast emulation through unicasting.



In general, when the same information is received by two or more hosts, multicasting is motivated by:

- Higher efficiency as multicasting requires less bandwidth than multiple unicasting. This efficiency is observed not only in the saved bandwidth but also in having packets arrived almost at the same time by many of the receivers. Sources of multicast packets are also unloaded of the transmission of packets as a single packet may suffice for all receivers.

7.2 Multicast Routing

7.2.1 Optimal Routing: Shortest-Path Trees

The process of finding an end-to-end path for delivering a packet to the destination is based on finding the shortest path. The set of end-to-end paths from a source node forms a shortest-path tree. The root of the tree is the source and the leaves are the possible destinations. In general, the objective is to find the path from the root to each destination as being the shortest path. However, multicast routing may be different from unicast routing as a multicast packet is destined to multiple receivers.

7.2.2 Unicast Routing

In unicast routing, packet sources find the next hop information (interface where a packet can be forwarded) from finding the shortest path to the possible destination. In this process, each router communicates with other routers to find the path length to other routers. A routing table summarizes the path length estimation for each router. This table indicates the next hop address for each possible destination.

7.2.3 Multicast Routing

A multicast packet may be destined to hosts in one or more networks. Forwarding a single packet to members of a group is also expected to follow a shortest-path tree [4] as in unicast routing. If there are n multicast groups, n shortest-path trees may be required, and this can make routing and the routing table at each router complex. In general, multicast routing may be seen as multiple unicast routing trees, also called the source-based tree approach [5], which can use unicast routing to obtain all needed trees, or else,

find a tree that can be shared by not only all possible destinations of a multicast group, but also by all multicast groups in a network, also called the shared-tree approach.

7.2.3.1 Source-Based Tree

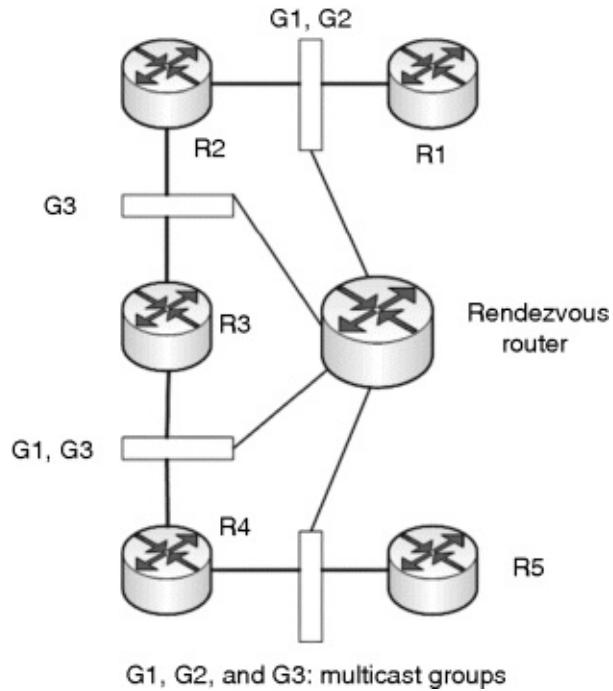
In this approach, each router calculates one shortest-path tree for each multicast group. The shortest-path tree for a group indicates the next hop for each destination within the multicast group.

For example, if the number of groups is n , each router calculates m shortest-path trees. The routing table stores one or many next hops for each group. If the number of groups is large, the table grows very large. However, the processes to find the trees may well follow an existing unicast routing algorithm.

7.2.3.2 Group-Shared Tree

In this approach, instead of each router having n shortest-path trees, there is only one tree in charge of disseminating multicast packets for single and multiple groups. As finding this tree looks complex, a designated router called the center core or rendezvous router takes the responsibility of distributing multicast traffic. The rendezvous router, as shown in [Figure 7.4](#), has n shortest-path trees in its routing table. If a router other than the rendezvous router receives a multicast packet, it encapsulates the packet in a unicast packet and sends it to the rendezvous router. This router decapsulates the multicast packet and route the packet according to the multicast routing table.

[Figure 7.4](#) Rendezvous router for shared-tree multicast routing.



7.3 Routing Protocols

[Figure 7.5](#) shows a classification of the routing protocols. Some of these protocols are described in the following sections.

[Figure 7.5](#) Classification of multicast routing protocols.

Source-Based Tree	Group-Shared Tree
MOSPF	CBP
DVMRP	PIM-SM
PIM-DM	

7.3.1 Multicast Open Shortest Path First (MOSPF)

In multicast routing, a node advertises to every group that has a subscriber on the link. The information about the group memberships is collected by using the Internet Group Management Protocol (IGMP). Each router running IGMP solicits membership status to the hosts sharing a link. When a router receives all link state packets (LSPs), it creates n topologies, one per group. From these n topologies, n shortest-path trees are made using Dijkstra's algorithm. In each routing table of a router, there is one shortest-path tree for each multicast group. The disadvantage of this protocol is the time and space needed to create and save the many shortest-path trees. The solution is to

create the trees only when needed. When a router receives a packet with a multicast destination address, it runs the Dijkstra algorithm to calculate the shortest-path tree for that group. The result can be cached in case there are additional packets for that destination.

The Open Shortest Path First (OSPF) protocol, designed originally for unicasting, has been modified to perform routing for multicasting. The packet formats in Multicast OSPF (MOSPF) are the same as in OSPF version 2. One additional option has been added to the Options field that appears in OSPF Hello packets, Database Description packets, and all link-state advertisements. This new option indicates multicast capability of a router or network. The presence of this new option is ignored by all non multicast routers.

- *T-bit* describes the router's TOS capability.
- *E-bit* ensures that all members of a stub area agree on the configuration of that area. Autonomous system (AS) external link advertisements are not flooded into or through OSPF stub areas.
- *MC-bit* describes the multicast capability of various pieces of the OSPF routing domain.

To support MOSPF, one of OSPF's link-state advertisements (LSA) has been modified and a new LSA has been added. The format of the router's LSA has been modified to include a flag that indicates whether the router is a wild-card multicast receiver. The type field in the router's LSA contains the following information:

- *bit B* indicates that the router is an area border router. These routers forward unicast data traffic between OSPF areas.
- *bit E* indicates that the router is an AS boundary router. These routers forward unicast data traffic between Autonomous Systems.
- *bit V* indicates that the router is an endpoint of an active virtual link which uses the described area as its Transit area.
- *bit W* indicates that the router is a wild-card multicast receiver. These routers receive all multicast datagrams, regardless of destination. Inter-area multicast forwarders and inter-AS multicast forwarders are sometimes wild-card multicast receivers.

A new LSA, called the group-membership-LSA, has been added to pinpoint multicast group members in the link-state database. This new advertisement is neither flooded nor processed by non multicast routers.

7.3.2 Distance Vector Multicast Routing Protocol

The Distance Vector Multicast Routing Protocol (DVMRP) [6] is an interior gateway protocol, suitable for use within an autonomous system. It is derived from the Routing Information Protocol (RIP). DVMRP differs from RIP, in which RIP thinks in terms of routing and forwarding packets to a particular destination. The purpose of DVMRP is to keep track of the return paths to the source of multicast packets. After a packet is forwarded, the routing information of a DVMRP router disappears and the router is ready for a new tree calculation. The routing table is then temporary. DVMRP uses a process based on four decision-making strategies. Each strategy is built on its predecessor and is explained below.

Flooding.

In flooding, a router receives a packet and sends it out from every interface except for the one from which it was received. This strategy is the simplest but it actually works as a broadcast and loops can be created. It is then necessary to eliminate duplicated packets.

Reverse Path Forwarding.

Reverse path forwarding (RPF) is based on unicast routing by identifying the reverse path, which is the shortest path to the source. The identified interfaces become part of the shortest-path trees.

Reverse Path Broadcasting.

Reverse Path Broadcasting (RPB) is different from RPF as RPB identifies a designated router in addition to the interface with the shortest path to the source. In this case, if packets come through different routers to the shortest-path tree, the packet coming through the designated router is considered.

Reverse Path Multicasting.

Reverse Path Multicasting (RPM) is very similar to RPB but it only considers those groups and networks with active users (RPM broadcast to all interfaces). This increases the efficiency of RPM. RPM is achieved using pruning and grafting. Pruning is the process of removing links toward networks with no group members. Grafting is the process of adding links toward networks with new group members or subscriptions. IGMP can be

used to assist in performing these processes.

Protocol Description.

DVMRP uses IGMP to exchange routing datagrams. DVMRP datagrams are composed of two portions: a small, fixed length IGMP header and a stream of tagged data. The fixed-length IGMP header of DVMRP messages is shown in [Figure 7.6](#). The version is 1. The type for DVMRP is 3. The subtype is one of the following:

1. Response: The message provides routes to some destination(s).
2. Request: The message requests routes to some destination(s).
3. Non membership report: The message provides non membership report(s).

[Figure 7.6](#) Format of the DVMRP IGMP header.

0	4	8	16	31
Version	Type	Subtype		Checksum

7.3.3 Core-Based Tree (CBT) Protocol

The Core-Based Tree (CBT) Protocol [7] is a group-shared protocol that uses a core router as the root of the routing tree. CBT is a tree center-based protocol using one tree per group. One of the routers in the tree is called the core. The core router receives multicast packets that are distributed to the other networks or routers. In this way, the core router becomes an important component of the shared tree.

All CBT control messages have a common fixed-length header. The packet header format is shown in [Figure 7.7](#).

[Figure 7.7](#) CBT control packet header.

0	4	8	16	31
Version	Type	Address length		Checksum

This CBT specification is version 2. The CBT packet types are as follows:

- Type 0: HELLO
- Type 1: JOIN_REQUEST
- Type 2: JOIN_ACK
- Type 3: QUIT_NOTIFICATION
- Type 4: ECHO_REQUEST

- Type 5: ECHO_REPLY
- Type 6: FLUSH_TREE
- Type 7: BOOTSTRAP MESSAGE (OPTIONAL)
- Type 8: CANDIDATE CORE ADVERTISEMENT (OPTIONAL)

The other fields are *Addr Length*, which indicates the address length in bytes of unicast or multicast addresses carried in the control packet, and *Checksum*, which indicates the 16-bit one's complement of the sum of the entire CBT control packet.

7.3.4 Protocol-Independent Multicast

Protocol-Independent Multicast (PIM) [8] consists of two versions of the protocol: Protocol-Independent Multicast Dense Mode (PIM-DM) and Protocol-Independent Multicast Sparse Mode (PIM-SM). These are described in the following.

PIM-DM.

[9] This protocol can adopt a unicast routing protocol (e.g., RIP or OSPF). This protocol can be considered for networks where multicast routers are plenty. The protocol builds routing trees in a similar way to DVMRP, using RFP. It can prune branches of the routing tree that has no subscribers.

PIM-SM.

[10] This protocol, also independent of the unicast protocol adopted, can be used when not all routers support multicast forwarding. To work around this limitation, it adopts a shared tree routing, where a rendezvous router can be used.

7.3.5 Simple Multicast Routing Protocol

The Simple Multicast Routing Protocol (SMRP) is a transport-layer protocol developed to route multimedia data streams over AppleTalk networks. It supports Apple Computer's QuickTime Conferencing (QTC) technology. SMRP provides connectionless, best-effort delivery of multicast datagrams and relies on underlying network-layer protocols for services. In particular, SMRP facilitates the transmission of data from a single source to multiple destinations. SMRP is designed to enable routers and end stations to

exchange multicast packets over network-layer protocols. SMRP provides the capability to manage multicast address assignment and enables a single source to send data addressed to a unique multicast group address. Receivers join this group if they are interested in receiving data for this group.

SMRP Multicast Address Management.

SMRP addressing is based on the local network of a creator endpoint. An SMRP address consists of two parts: a three-byte network number and a one-byte socket number. Each local network is configured with a range of unique network numbers. In network number mapping, network numbers must be assigned to local nets for SMRP and must be unique throughout an entire internetwork. Each local net can be assigned any contiguous range of three-byte network numbers. The number of multicast groups available for a local network is the number of network numbers assigned multiplied by 254. Network numbers can be configured or mapped from the network number of underlying network-layer protocols. Unique network number ranges can be reserved for supported network protocols. In the case of multicast address mapping, SMRP multicast addresses must be mapped to network-layer multicast addresses, and these in turn are mapped to data link-layer multicast addresses. For each network-layer type, a block of multicast addresses must be obtained for SMRP. In the best case, these addresses will be mapped directly. In most cases, a direct mapping is not possible, and more than one SMRP multicast address is mapped to a single network-layer multicast address. The manner in which multicast addresses are mapped to network-layer addresses is network-layer-dependent. When SMRP transport-layer multicast addresses do not map directly to network-layer multicast addresses, filtering of the SMRP multicast addresses is required. When network-layer multicast addresses do not map directly to data link-layer multicast addresses, the network layer is expected to filter out multicast addresses that have not been subscribed.

SMRP Multicast Transaction Protocol.

SMRP involves a multicast transaction protocol (MTP) that provides for three transaction types: node, endpoint, and simultaneous node–endpoint. Communication between adjacent nodes and between nodes and endpoints occurs through request–response transactions. Responses always are unicast. MTP provides for the retransmission of requests or responses in case of

network errors. Only hello and designated node-request packets are sent as multicast messages; all others are unicast. Endpoint-to-node requests are sent as multicasts, while node-to-endpoint requests are sent as either unicast or multicast.

7.4 Anycasting

Anycast is a network addressing and routing scheme whereby data is routed to the nearest or most convenient destination as viewed by the routing topology [11]. The term *anycast* is intended to echo the terms *unicast*, *broadcast*, and *multicast* with a subtle difference. On the Internet, anycast is usually implemented by using BGP to simultaneously announce the same destination IP address range from many different places on the Internet. This results in packets addressed to destination addresses in this range being routed to the nearest point on the network announcing the given destination IP address. In the past, anycast was suited to connectionless protocols (generally built on UDP), rather than connection-oriented protocols such as TCP that keep their own state. However, there are many cases where TCP anycast is now used. With TCP anycast, there are cases where the receiver selected for any given source may change from time to time as optimal routes change, silently breaking any conversations that may be in progress at the time. These conditions are typically referred to as a *pop switch*. To correct this issue, proprietary advance within custom IP stacks now allow for healing of stateful protocols where required. For this reason, anycast is generally used as a way to provide high availability and load balancing for stateless services such as access to replicated data; for example, DNS service is a distributed service over multiple geographically dispersed servers. In IPv4 to IPv6 transitioning anycast addressing may be deployed to provide IPv4 compatibility to IPv6 hosts. This method, called 6to4, uses a default gateway with the IP address 192.88.99.1 [12]. This allows multiple providers to implement 6to4 gateways without hosts having to know each individual provider's gateway address. In some situations of anycast deployment on the Internet there is a difference between local and global nodes. Local nodes are often more intended to provide benefit for the direct local community. Local node announcements are often announced with the no-export BGP community to prevent peers from announcing them to their peers (i.e., the announcement is kept in the local area). Where both local and global nodes

are deployed, the announcements from global nodes are often AS prepended (i.e., the AS is added a few more times) to make the path longer so that a local node announcement is preferred over a global node announcement.

7.4.1 Architectural Issues

Adding anycasting to the repertoire of IP services requires some decisions to be made about how to balance the architectural requirements of IP with those of anycasting. The first and most critical architectural issue is how to balance IP's stateless service with the desire to have an anycast address represent a single virtual host. The best way to illustrate this problem is with a couple of examples. In both of these examples, two hosts (X and Y) are serving an anycast address and another host (Z) is using the anycast address to contact a service.

In the first example, suppose that Z sends a UDP datagram addressed to the anycast address. Now, given that an anycast address is logically considered the address of a single virtual host, should it be possible for the datagram to be delivered to both X and Y? The answer to this question has to be yes, delivery to both X and Y is permissible. IP is allowed to duplicate and misroute datagrams, so there are clearly scenarios in which a single datagram could be delivered to both X and Y. The implication of this conclusion is that the definition of anycasting in an IP environment is that IP anycasting provides best-effort delivery of an anycast datagram to one, but possibly more than one, of the hosts that serve the destination anycast address.

In the second example, suppose that Z sends two datagrams addressed to the anycast address. The first datagram gets delivered to X. To which host (X or Y) does the second datagram get delivered? It would be convenient for stateful protocols like TCP if all of a connection's datagrams were delivered to the same anycast address. However, because IP is stateless (and thus cannot keep track of where earlier datagrams were delivered) and because one of the goals of anycasting is to support replicated services, it seems clear that the second datagram can be delivered to either X or Y. Stateful protocols have to employ some additional mechanism to ensure that later datagrams are sent to the same host.

After considering the two examples, the definition of IP anycasting can be defined as a service that provides stateless best-effort delivery of an anycast datagram to at least one host, and preferably only one host, that serves the

anycast address. This definition makes it clear that anycast datagrams receive the same basic type of service as IP datagrams. While the definition permits delivery to multiple hosts, the goal is delivery to just one host.

7.4.2 Anycast Addresses

There appear to be a number of ways to support anycast addresses. Some use small pieces of the existing address space, and others require that a special class of IP addresses be assigned.

The major advantage of using the existing address space is that it may make routing easier. As an example, consider a situation where a portion of each IP network number can be used for anycasting. For example, a site could assign a set of its subnet addresses to be anycast addresses. If, as some experts expect, anycast routes are treated just like host routes by the routing protocols, the anycast addresses would not require special advertisement outside the site; the host routes could be folded in with the net route. If the anycast addresses are supported by hosts outside the network, then those hosts would still have to be advertised using host routes. The major disadvantages of this approach are:

1. There is not an easy way for stateful protocols like TCP to discover that an address is an anycast address.
2. It is more difficult to support Internet-wide well-known anycast address.

The Internet might establish that a particular anycast address is the logical address of the DNS server. Then host software could be configured at the manufacturer to always send DNS queries to the DNS anycast address. In other words, anycasting could be used to support autoconfiguration of DNS resolvers. The major advantages of using a separate class of addresses are that it is easy to determine if an address is an anycast address and well-known anycast addresses are easier to support. A major disadvantage is that routing may be more complex because the routing protocols may have to keep track of more anycast routes.

An intermediate approach is to take part of the current address space (say 256 Class C addresses) and make the network addresses into anycast addresses (and ignore the host part of the class C address). The advantage of this approach is that it makes anycast routes look like network routes, which are easier for some routing protocols to handle. The disadvantage is that it uses the address space inefficiently and so more severely limits the number of

anycast addresses that can be supported.

Carving anycast addresses from the existing address space seems more likely to cause problems in situations in which either applications mistakenly fail to recognize anycast addresses (if anycasts are part of each site's address space) or use the address space inefficiently (if network addresses are used as anycast addresses). The advantages of using anycast addresses for autoconfiguration seem compelling. It has been considered that anycast addresses could be a separate class of IP addresses. Since each anycast address is a virtual host address and the number of anycasting hosts seems unlikely to be larger than the number of services offered by protocols like TCP and UDP, the address space could be quite small.

7.4.3 Differences between the Services Offered by IP Multicasting and IP Anycasting

Semantically, the difference between IP multicasting and IP anycasting is that an anycast address is the address of a single (virtual) host and that the internetwork will make an effort to deliver anycast datagrams to a single host. There are two implications of this difference. First, applications sending data to anycast addresses need not worry about managing the TTLs of their IP datagrams. Applications using multicast to find a service must balance their TTLs to maximize the chance of finding a server while minimizing the chance of sending datagrams to a large number of irrelevant servers. Second, making a TCP connection to an anycast address makes perfectly good sense, while the meaning of making a TCP connection to a multicast address is unclear because TCP is not designed to support multicasting.

From a practical perspective, the major difference between anycasting and multicasting is that anycasting is a special use of unicast addressing while multicasting requires more sophisticated routing support. The important observation is that multiple routes to an anycast address appear to a router as multiple routes to a unicast destination, and the router can use standard algorithms to choose the best route.

Another difference between the two approaches is that resource location using multicasting typically causes more datagrams to be sent. To find a server using multicasting, an application is expected to transmit and retransmit a multicast datagram with successively larger IP TTLs. The TTL is initially kept small to limit the number of servers contacted. However, if no

servers respond, the TTL must be increased on the assumption that the available servers (if any) were farther away than was reachable with the initial TTL. As a result, resource location using multicasting causes one or more multicast datagrams to be sent toward multiple servers, with some datagrams' TTLs expiring before reaching a server. With anycasting, management of TTL is not required and so (ignoring the case of loss) only one datagram need be sent to locate a server. Furthermore, a datagram like this follows only a single path.

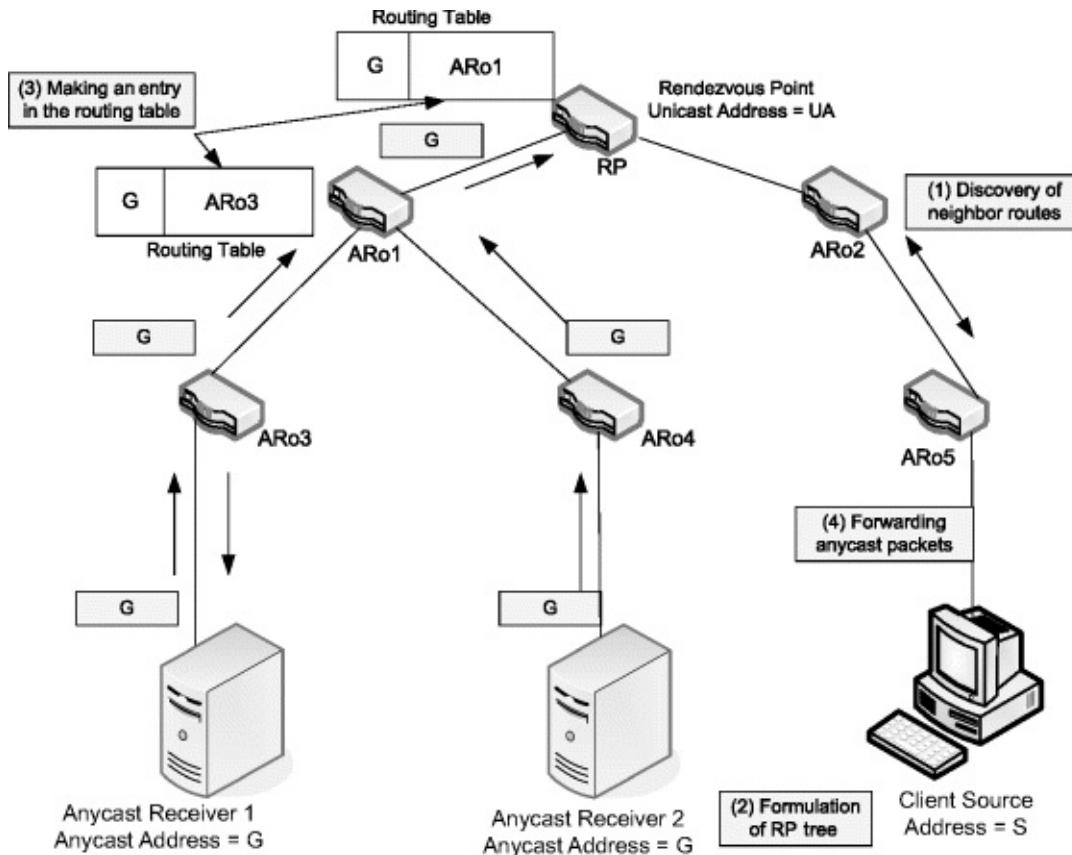
A minor difference between the two approaches is that anycast may be less fault tolerant than multicast. When an anycast server fails, some datagrams may continue to be mistakenly routed to the server, while if the datagram had been multicast, other servers would have received it.

7.5 IPv6 Anycast Routing Protocol: Protocol-Independent Anycast—Sparse Mode

Three types of multicast routing protocols are available today, and each multicast protocol has both advantages and disadvantages. Protocol-Independent Multicast—Sparse Mode (PIM-SM) [13], as one of the multicast routing protocols, has an advantage in global multicasting. In other words, PIM-SM is designed to be used in a network where multicast listeners are sparsely distributed. This model is very similar to the case where anycast receivers for a single anycast address are widely spread throughout the Internet.

[Figure 7.8](#) shows an overview of PIM-SM. In this figure we assume that all routers support PIM-SM (we refer to them as PIM routers). Also, upstream means the direction toward the root (i.e., RP) of the RPT, and downstream is the direction to receivers. RPT is constructed for each anycast address.

[Figure 7.8](#) Overview of PIM-SM.



References

1. Albanna, Z., Almeroth, K., Meyer, D. and Schipper, M., "Guidelines for IPv4 Multicast Address Assignments," IETF RFC 3171, Aug. 2001.
2. Mills, D.L., "Network Time Protocol (Version 1) Specification," IETF RFC 958, Sep. 1985.
3. Mills, D.L., " Network Time Protocol (Version 3) Specification," IETF RFC 1305, Mar. 1992.
4. Savola, P., "Overview of the Internet Multicast Routing Architecture," IETF RFC 5110, Jan. 2008.
5. Forouzan, B.A., TCP/IP Protocol Suite (Fourth Edition), McGraw Hill, 2010.
6. Waitzman, D., Partridge, C. and Deering, S., "Distance Vector Multicast Routing Protocol," IETF RFC 1075, IETF, Nov. 1988.
7. Ballardie, A., "Core Based Trees (CBT) Multicast Routing Architecture,"

IETF RFC 2201, Sep. 1997.

- 8.** Bhaskar, N., Gall, A., Lingard, J. and Venaas, S., “Bootstrap Router (BSR) Mechanism for Protocol Independent Multicast (PIM),” IETF RFC5059, Jan. 2008.
- 9.** Adams, A., Nicholas, J. and Siadak, W., “Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised),” IETF RFC 3973, Jan. 2005.
- 10.** Savola, P., Lehtonen, R. and Meyer, D., “Protocol Independent Multicast -Sparse Mode (PIM-SM) Multicast Routing Security Issues and Enhancements,” IETF RFC 4609, Aug. 2006.
- 11.** Partridge, C., Mendez, T. and Milliken, W., “Host Anycasting Service,” IETF RFC 1546, Nov. 1993.
- 12.** Huitema, C., “An Anycast Prefix for 6to4 Relay Routers,” IETF RFC 3068, June 2001.
- 13.** Farinacci, D. and Cai, Y., “Anycast-RP Using Protocol Independent Multicast (PIM),” IETF RFC 4610, Aug. 2006.

Chapter 8

Layer-2 Transport over Packet

This chapter presents several Layer-2 protocols. These protocols are used to encapsulate Layer-2 protocol data units (PDUs) for transmitting datagrams in the network. The chapter presents several examples of these encapsulation protocols, including Martini, Layer-2 Transport, and Pseudowire Emulation Edge-to-Edge.

8.1 Draft-Martini Signaling and Encapsulation

Draft-Martini encapsulation [1] describes methods for encapsulating the Layer-2 PDUs such as Frame Relay, Asynchronous Transfer Mode (ATM), and Ethernet across a Multiprotocol Label Switching (MPLS) network.

Martini drafts contributed to the Internet Engineering Task Force's (IETF's) Pseudo Wire Emulation Edge-to-Edge (PWE3) working group by Luca Martini of Cisco Systems (formerly employed by Level 3 Communications) and a number of other authors. These drafts define a method to transport a Layer-2 protocol across an MPLS network. The *Martini-trans* series defines the transport mechanism using the Label Distribution Protocol (LDP). The *Martini-encaps* series of drafts define how specific Layer-2 protocols are encapsulated prior to being carried in the MPLS network. Provision is made for the synchronous optical networking (SONET) and synchronous digital hierarchy (SDH), Ethernet, ATM and frame relay. This protocol has been superseded by the PWE3 working group specifications described in RFC 4447 [2].

8.1.1 Functionality

In an MPLS network, it is possible to use control protocols such as those specified in RFC4906 [3] to set up the *emulated virtual circuits* that carry the PDUs of Layer-2 protocols across the network. A number of these emulated

virtual circuits (VCs) may be carried in a single tunnel. This transport mode requires the encapsulation of Layer-2 PDUs. There are three layers of encapsulations that can be possible in such cases:

1. Tunnel header
2. Demultiplexer field
3. Emulated VC encapsulation

Draft-Martini specifies the emulated VC encapsulation for a number of Layer-2 protocols. Although different Layer-2 protocols require different information to be carried in this encapsulation, an attempt has been made to make the encapsulation as common as possible for all Layer-2 protocols.

8.1.2 Encapsulation

First consider an example where R1 is defined as the ingress router and R2 as the egress router. A Layer-2 PDU is received at R1, encapsulated at R1, transported, decapsulated at R2, and transmitted out of R2. The network may not transport the Layer-2 (Draft-Martini) encapsulation across the network; rather, the Layer-2 header can be stripped at R1 and reproduced at R2. This is done using information carried in a control word, provided by the Martini encapsulation, as well as information that may already have been signaled from R1 to R2. The following requirements may need to be satisfied when transporting Layer-2 protocols over an MPLS backbone:

- Sequentiality may need to be preserved.
- Small packets may need to be padded in order to be transmitted on a medium where the minimum transport unit is larger than the actual packet size.
- Control bits carried in the header of the Layer-2 frame may need to be transported.

The control word, shown in [Figure 8.1](#), addresses all three requirements. For some protocols this control word is required, and for others it is optional. For protocols where the control word is optional, implementations must support the action of sending (or not sending) a control word.

Figure 8.1 Format of the control word in Martini encapsulation.

					31
0	4	8	10	16	
Rsvd	Flags	00	Length	Sequence number	

The fields of the control word are the following: *RSVD* (4 bits), which is reserved for future use. These must be set to 0 when transmitting but are ignored upon receipt. The field *FLAGS* (4 bits) indicates protocol-specific flags. These are defined in the protocol-specific details below. The next 2 bits must be set to 0 when transmitting. *Length* (6 bits) provides the length field. If the packet's length (defined as the length of the Layer-2 payload plus the length of the control word) is smaller than 64 bytes, the length field is set to the packet's length. Otherwise, the length field is set to 0. The value of the length field, if nonzero, can be used to remove the padding. When the packet reaches the service provider's egress router, it may be desirable to remove the padding before forwarding the packet. The next 16 bits provide a sequence number that can be used to guarantee ordered packet delivery. The processing of the sequence number field is optional. The sequence number space is a 16-bit, unsigned circular space. The sequence number value 0 is used to indicate a not-sequenced packet.

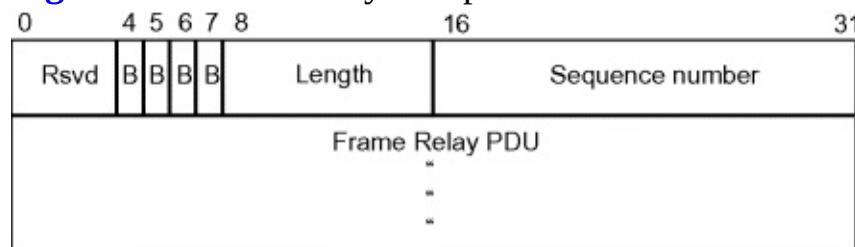
8.1.3 Protocol-Specific Encapsulation

Encapsulation of several Layer-2 protocols has been proposed. The following is a list of the most popular encapsulations.

8.1.3.1 Draft-Martini Specification for Frame Relay

A Frame Relay PDU is transported without the Frame Relay header or the frame check sequence (FCS). [Figure 8.2](#) shows this encapsulation. The control word is required. However, its use is optional, although desirable. The use of the control word means that the ingress and egress label switching routers follow the procedures: if an ingress label switching router chooses not to use the control word, it must set the flags in the control word to 0; if an egress label switching router chooses to ignore the control word, it must set the Frame Relay control bits to 0.

[Figure 8.2](#) Frame Relay encapsulation.



The Backward Explicit Congestion Notification (BECN), Forward Explicit Congestion Notification (FECN), Discard Eligibility (DE), and Command/Response (C/R) bits are carried across the network in the control word.

The edge routers that implement Draft-Martini specification may, when either adding or removing the encapsulation, change the BECN and/or FECN bits from 0 to 1 in order to reflect the network congestion known to the edge routers, and the D/E bit from 0 to 1 to reflect marking from edge policing of the Frame Relay Committed Information Rate. The BECN, FECN, and DE bits should not be changed from 1 to 0.

8.1.3.2 Asynchronous Transfer Mode

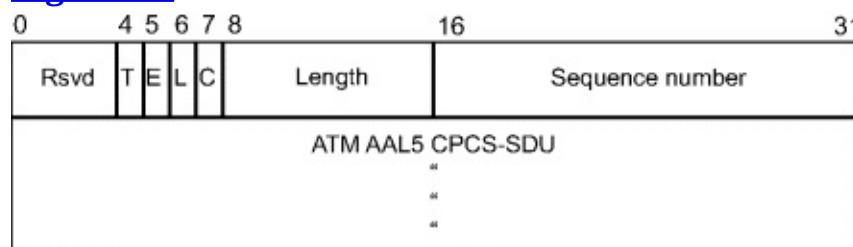
Two encapsulations are supported for ATM transport: one for ATM Adaption Layer 5 (AAL5) and another for ATM cells. These are summarized as follows:

- The AAL5 Common Part Convergence Sublayer—Service Data Unit (CPCS-SDU) encapsulation consists of the required control word and the AAL5 CPCS-SDU.
- The ATM cell encapsulation consists of an optional control word, a 4-byte ATM cell header, and the ATM cell payload.

8.1.3.3 Draft-Martini Specification for ATM AAL5 CPCS-SDU Mode

In ATM AAL5 mode, the ingress router is required to reassemble AAL5 CPCS-SDUs from the incoming VC and transport each CPCS-SDU as a single packet ([Figure 8.3](#)). No AAL5 trailer is transported. The Explicit Forward Congestion Indication (EFCI) and Cell Loss Priority (CLP) bits are carried across the network in the control word.

Figure 8.3 Header format for AAL5 CPCS-SDU mode.



The edge routers that implement Draft-Martini specification may, when either adding or removing the encapsulation, change the EFCI bit from 0 to 1 in order to reflect congestion in the network that is known to the edge routers, and the CLP bit from 0 to 1 to reflect marking from edge policing of the ATM Sustained Cell Rate. The EFCI and CLP bits must not be changed from 1 to 0.

ATM cells are transported individually without a segmentation and reassembly (SAR) process.

The ATM cell encapsulation consists of an optional control word and one or more ATM cells, each of which consists of a 4-byte ATM cell header and the 48-byte ATM cell payload.

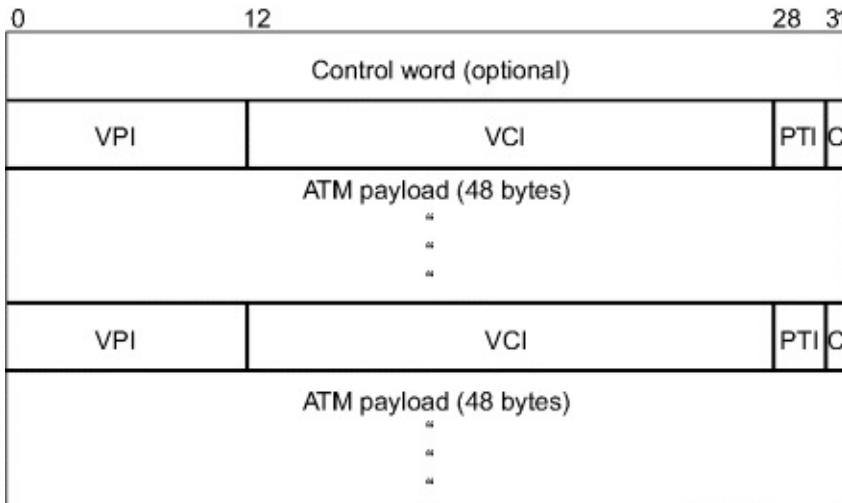
The length of each frame without the encapsulation header is a multiple of 52 bytes. The maximum number of ATM cells that can be fitted in a frame in this fashion is limited only by the network maximum transmission unit (MTU) and by the ability of the egress router to process them.

The ingress router must not send more cells than the egress router is willing to receive. The number of cells that the egress router is willing to receive may either be configured in the ingress router or signaled. The number of cells encapsulated in a particular frame can be inferred by the frame length.

The control word is optional. In most cases, if the control word is used, then the Flag bits in the control word are not used. The EFCI and CLP bits (of an ATM cell) are carried across the network in the ATM cell header. The edge routers that implement Draft-Martini specification may, when either adding or removing the encapsulation, change the EFCI bit from 0 to 1 in order to reflect congestion in the network that is known to the edge router, and the CLP bit from 0 to 1 to reflect marking from edge policing of the ATM sustained cell rate. The EFCI and CLP bits should not be changed from 1 to 0.

[Figure 8.4](#) shows an illustration of a Draft-Martini encapsulation of two ATM cells. Details on the Virtual Path Identifier (VPI), Virtual Circuit Identifier (VCI), Payload Type Identifier (PTI), and CLP fields are discussed as follows:

[Figure 8.4](#) ATM encapsulation.



- **VPI:** The ingress router must copy the VPI field from the incoming cell into this field. For particular emulated VCs, the egress router may generate a new VPI and ignore the VPI contained in this field.
- **VCI:** The ingress router must copy the VCI field from the incoming ATM cell header into this field. For special emulated VCs, the egress router may generate a new VCI.
- **PTI and CLP (C bit):** The PTI and CLP fields are the PTI and CLP fields of the incoming ATM cells. The cell headers of the cells within the packet are the ATM headers (without the Header Error Correction (HEC) field) of the incoming cell.

8.1.3.4 Draft-Martini Specification for Ethernet VLAN

For an Ethernet 802.1q VLAN, the entire Ethernet frame without the preamble or FCS is transported as a single packet.

The control word is optional. If the control word is used, then the flag bit in the control word is not used; it must be set to 0 when transmitting and must be ignored upon receipt. The 4-byte VLAN tag is transported as it is and may be overwritten by the egress router. The ingress router may consider the User Priority field (see IEEE802.3ac) of the VLAN tag header when determining the value to be placed in the Quality of Service field of the encapsulating protocol (e.g., the EXP fields of the MPLS label stack).

The egress router may consider the Quality of Service field of the encapsulating protocol when queuing the packet for egress. Ethernet packets containing hardware-level cyclic redundancy check (CRC) errors, framing

errors, or runt packets must be discarded on input.

8.1.3.5 Draft-Martini Specification for Ethernet

For simple Ethernet port to port transport, the entire Ethernet frame without the preamble or FCS is transported as a single packet.

The control word is optional. If the control word is used, then the flag bits in the control word are not used. The flag bits must be set to 0 when transmitting, and they must be ignored upon receipt. Ethernet packets with hardware-level CRC errors, framing errors, and runt packets must be discarded on input.

8.1.3.6 Draft-Martini Specification for Point-to-Point Protocol

Point-to-Point Protocol (PPP) mode provides point-to-point transport of PPP-encapsulated traffic [4,5]. The PPP PDU is transported in its entirety including the protocol field (whether compressed using PFC or not), but excluding any media-specific framing information such as the High-Level Data Link Control (HDLC) address and control fields or Frame Check Sequence (FCS). Since media-specific framing is not carried, the following options will not operate correctly if the PPP peers attempt to negotiate them:

- Frame Check Sequence alternatives
- Address-and-Control-Field-Compression (ACFC)
- Asynchronous-Control-Character-Map (ACCM)

It should be noted that VC LSP Interface MTU negotiation as specified in RFC4906 is not affected by PPP Maximum Receive Unit (MRU) advertisements. Thus, if a PPP peer sends a PDU with a length in excess of that negotiated for the VC LSP, that PDU will be discarded by the ingress router. The control word is optional. If the control word is used, then the flag bits in the control word are not used; they must be set to 0 when transmitting and must be ignored upon receipt.

8.2 Layer-2 Tunneling Protocol

The Layer-2 Tunneling Protocol (L2TP) provides a dynamic mechanism for tunneling Layer-2 *circuits* across a packet-oriented data network (e.g., over

IP) [6]. L2TP, as originally defined in RFC 2661 [4], is a standard method for tunneling PPP sessions.

PPP defines an encapsulation mechanism for transporting multiprotocol packets across Layer-2 point-to-point links. Typically, a user obtains a Layer-2 connection to a network access server (NAS) using one of a number of techniques (e.g., plain old telephone service [POTS], Integrated Services Digital Network [ISDN], Asymmetrical Digital Subscriber Line [ADSL]) and then runs PPP over that connection. In such a configuration, the L2 termination point and PPP session endpoint reside on the same physical device (i.e., the NAS).

L2TP extends the PPP model by allowing Layer-2 and PPP endpoints to reside on different devices interconnected by a packet-switched network. With L2TP, a user has a Layer-2 connection to an access concentrator (e.g., modem bank, ADSL Digital Subscriber Line Access Multiplexer [DSLAM]), and the concentrator tunnels individual PPP frames to the NAS. This allows the actual processing of PPP packets to be divorced from the termination of the Layer-2 circuit.

A benefit of such a separation is that instead of requiring that the L2 connection terminate at the NAS, which may involve a long-distance toll charge, the connection can terminate at a local circuit concentrator, which extends the logical PPP session over a shared infrastructure such as a frame relay circuit or the Internet. From the user's perspective, there is no functional difference between the L2 circuit terminating directly in an NAS or using L2TP.

8.2.1 Layer-2 Tunneling Protocol Version 3

L2TPv3 is an IETF standard related to L2TP that can be used as an alternative protocol to MPLS for encapsulation of multiprotocol Layer-2 communications traffic over IP networks. Like L2TP, L2TPv3 provides a *pseudowire* service, but version 3 is scaled to fit carrier requirements. The relationship between L2TPv3 and MPLS is similar to that between IP and ATM. In other words, L2TPv3 is a simplified version of MPLS with many of the same benefits achieved with a fraction of the effort. However, some technical features considered to be less important in the market are lost. In the case of L2TPv3, the teletraffic engineering features that are lost are considered to be important in MPLS.

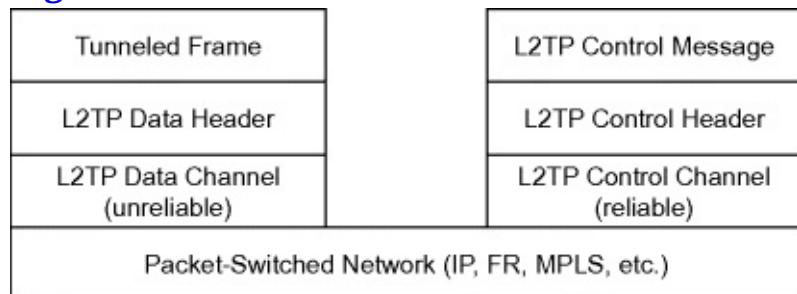
- Separation of all PPP-related attribute value pairs (AVPs), which are required for control connection, including a portion of the L2TP data header specific to the needs of PPP.
- Transition from a 16-bit Session ID and Tunnel ID to a 32-bit Session ID and Control Connection ID.
- Extension of the tunnel authentication mechanism to cover the entire control message rather than just a portion of the message.

The L2TPv3 control message format (described in the following section) definition is borrowed largely from L2TPv2. These control messages are used in conjunction with the associated protocol state machines that govern the dynamic setup, maintenance, and teardown for L2TP sessions. The data message format for tunneling data packets may be utilized with or without the L2TP control channel, either via manual configuration or via other signaling methods to preconfigure or distribute L2TP session information.

L2TP is comprised of two types of messages: control messages (control packets) and data messages (data packets). Control messages are used in the establishment, maintenance, and clearing of control connections and sessions. These messages utilize a reliable control channel within L2TP to guarantee delivery. Data messages are used to encapsulate the L2 traffic being carried over the L2TP session. Unlike control messages, data messages are not retransmitted when packet loss occurs.

[Figure 8.5](#) depicts the relationship of control messages and data messages over the L2TP control and data channels. Data messages are passed over an unreliable data channel, encapsulated by an L2TP header, and sent over a packet-switched network (PSN) such as IP, UDP, Frame Relay, ATM, or MPLS. Control messages are sent over a reliable L2TP control channel that operates over the same PSN.

Figure 8.5 L2TPv3 structure.



The necessary setup for tunneling a session with L2TP consists of two

steps:

1. Establish the control connection.
2. Establish the session triggered by an incoming call or an outgoing call.

An L2TP session is established before L2TP can begin to forward session frames. Multiple sessions may be bound to a single control connection, and multiple control connections may exist between the same two LCCEs.

8.2.1.1 Control Message Types

Connection setup, management, and tear up use the control messages listed in [Table 8.1](#).

Table 8.1 Control Message Types.

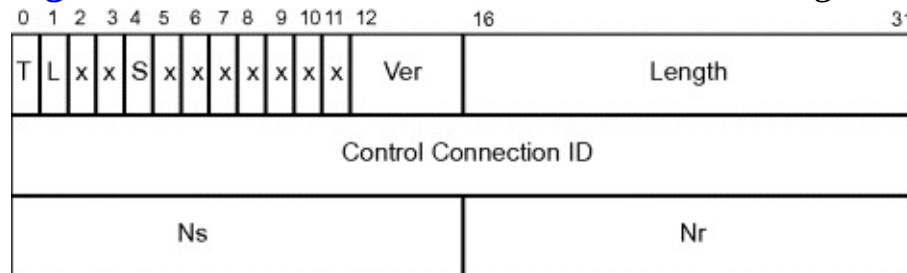
Control Connection Management		
0	(Reserved)	
1	Start control connection request	(SCCRQ)
2	Start control connection reply	(SCCRP)
3	Start control connection connected	(SCCCN)
4	Stop control connection notification	(StopCCN)
5	(Reserved)	
6	Hello	(HELLO)
20	Explicit acknowledgment	(ACK)
Call Management		
7	Outgoing call request	(OCRQ)
8	Outgoing call reply	(OCRP)
9	Outgoing call connected	(OCCN)
10	Incoming call request	(ICRQ)
11	Incoming call reply	(ICRP)
12	Incoming call connected	(ICCN)
13	(Reserved)	
14	Call disconnect notify	(CDN)
Error Reporting		
15	WAN error notify	(WEN)
Link Status Change Reporting		
16	Set link info	(SLI)

8.2.1.2 L2TP Control Message Header

The L2TP control message header provides information for the reliable transport of messages that govern the establishment, maintenance, and

teardown of L2TP sessions. By default, control messages are sent over the underlying media in-band with L2TP data messages. The L2TP control message header is formatted as shown in [Figure 8.6](#).

Figure 8.6 Header format of an L2TP control message.

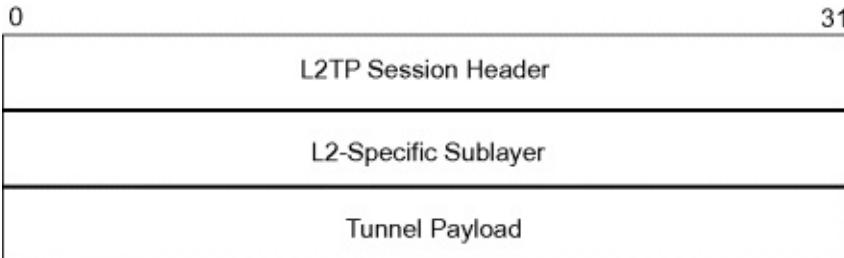


The L and S bits indicate that the length field and sequence numbers are present (if both are set to 1). The x bits are reserved for future extensions. All reserved bits must be set to 0 on outgoing messages and ignored on incoming messages. The Ver field indicates the version of the L2TP control, which is set to 3. The Length field indicates the total length of the message in octets, which is always calculated from the start of the control message header itself (beginning with the T bit). The Control Connection ID field contains the identifier for the control connection. L2TP control connections are named by identifiers that have local significance only. Nonzero Control Connection IDs are selected and exchanged as Assigned Control Connection ID AVPs during the creation of a control connection. Ns indicates the sequence number for this control message beginning at zero and incrementing by one (*modulo* 2^{16}) for each message sent. Nr indicates the sequence number of the next control message that is expected. Thus, Nr is set to the Ns of the last in-order message received plus one (*modulo* 2^{16}).

8.2.1.3 L2TP Data Message

In general, an L2TP data message consists of a (1) Session Header, (2) an optional L2-Specific Sublayer, and (3) the Tunnel Payload, as depicted in [Figure 8.7](#).

Figure 8.7 L2TP data message.



The L2TP Session Header is specific to the encapsulating PSN over which the L2TP traffic is delivered. The Session Header must provide (1) a method of distinguishing traffic among multiple L2TP data sessions and (2) a method of distinguishing data messages from control messages. Each type of encapsulating PSN defines its own session header by clearly identifying the format of the header and necessary parameters to set up the session. There are two session headers: one for transport over UDP and one for transport over IP. The L2-Specific Sublayer is an intermediary layer between the L2TP session header and the start of the tunneled frame. It contains control fields that are used to facilitate the tunneling of each frame (e.g., sequence numbers or flags). The Data Message Header is followed by the Tunnel Payload, including any necessary Layer-2 framing as defined in the payload specific for each protocol.

8.2.2 Pseudowire Emulation Edge to Edge

Pseudowire emulation provides an emulation of Layer-2 point-to-point connection-oriented service over a PSN [7]. The pseudowire emulates the operation of a *transparent wire* carrying a service. The service being carried over the wire may be ATM, Frame Relay, Ethernet, low-rate time-division multiplexing (TDM), or synchronous optical networking (SONET), while the packet network may be MPLS, IPv4 or IPv6, or L2TPv3.

In 2001, the IETF set up the Pseudowire Emulation Edge-to-edge (PWE3) working group. The working group was chartered to develop an architecture for service provider edge-to-edge pseudowires and service-specific documents detailing the encapsulation techniques.

PWE3 is a mechanism that emulates the essential attributes of a telecommunications service (such as a T1 leased line or Frame Relay) over a PSN. PWE3 is intended to provide only the minimum necessary functionality to emulate the wire with the required degree of faithfulness for the given service definition.

8.2.2.1 Functionality and Objective of PWE3

PWs provide the following functions with the objective to emulate the behavior and characteristics of the native service:

- 1.** Encapsulation of service-specific PDUs or circuit data arriving at the PE-bound port (logical or physical).
- 2.** Carriage of the encapsulated data across a PSN tunnel.
- 3.** Establishment of the PW, including the exchange and/or distribution of the PW identifiers used by the PSN tunnel endpoints.
- 4.** Managing the signaling, timing, order, or other aspects of the service at the boundaries of the PW.
- 5.** Service-specific status and alarm management.

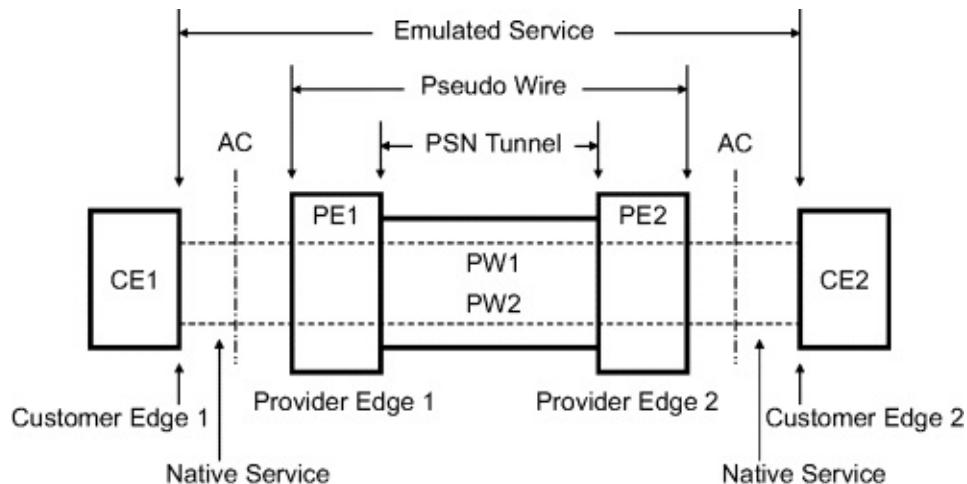
8.2.2.2 PWE3 Applicability

The PSN carrying a PW will subject payload packets to loss, delay, delay variation, and reordering. During a network transient there may be a sustained period of impaired service. The applicability of PWE3 to a particular service depends on the sensitivity of that service (or the CE implementation) to these effects, and on the ability of the adaptation layer to mask them. Some services, such as IP over Frame Relay over PWE3, may prove quite resilient to IP and MPLS PSN characteristics. Other services, such as the interconnection of private branch exchange (PBX) systems via PWE3, will require more careful consideration of the PSN and adaptation layer characteristics. In some instances, traffic engineering of the underlying PSN will be required, and in some cases the constraints may make the required service guarantees impossible to provide.

8.2.2.3 PWE3 Network Reference Model

The network reference model is presented in [Figure 8.8](#). The elements of this model are defined as follows:

Figure 8.8 Reference model of PWE3.



Attachment Circuit (AC)

The physical or virtual circuit attaching a CE to a PE. An attachment circuit may be, for example, a Frame Relay data link connection identifier (DLCI), an asynchronous transfer mode (ATM) virtual path identifier (VPI)/virtual channel identifier (VCI), an Ethernet port, a virtual local area network (VLAN), a PPP connection on a physical interface, a PPP session from an L2TP tunnel, or an MPLS LSP. If both physical and virtual ACs are of the same technology (e.g., both ATM, both Ethernet, both Frame Relay), the PW is said to provide *homogeneous transport*; otherwise, it is said to provide *heterogeneous transport*.

CE-Bound

The traffic direction in which PW-PDUs are received on a PW via the PSN, processed, and then sent to the destination CE.

CE Signaling

Messages sent and received by the CE's control plane. It may be desirable or even necessary for the PE to participate in or to monitor this signaling in order to emulate the service effectively.

Control Word (CW)

A four-octet header used in some encapsulations to carry per-packet information when the PSN is MPLS.

Customer Edge (CE)

A device where one end of a service originates and/or terminates. The CE is not aware that it is using an emulated service rather than a native service.

Forwarder (FWRD)

A PE subsystem that selects the PW to use in order to transmit a payload received on an AC.

Fragmentation

The action of dividing a single PDU into multiple PDUs before transmission with the intent of the original PDU being reassembled elsewhere in the network. Packets may undergo fragmentation if they are larger than the MTU of the network they will traverse.

Maximum Transmission

The packet size (excluding data link header) unit (MTU) that an interface can transmit without needing to fragment.

Native Service Processing (NSP)

Processing of the data received by the PE from the CE before presentation to the PW for transmission across the core, or processing of the data received from a PW by a PE before it is output on the AC. NSP functionality is defined by standards bodies other than the IETF, such as the International Telecommunications Union Telecommunication Standardization Sector (ITU-T), the American National Standards Institute (ANSI), or Asynchronous Transfer Mode Forum (ATMF).

PE-Bound

The traffic direction in which information from a CE is adapted to a PW, and PW-PDUs are sent into the PSN.

PE/PW Maintenance

Used by the PEs to set up, maintain, and tear down the PW. It may be coupled with CE Signaling in order to manage the PW effectively.

Provider Edge (PE)

A device that provides PWE3 to a CE.

Pseudo Wire (PW)

A mechanism that carries the essential elements of an emulated service from one PE to one or more other PEs over a PSN.

Pseudo Wire Emulation Edge to Edge (PWE3)

A mechanism that emulates the essential attributes of service (such as a T1 leased Edge (PWE3) line or Frame Relay) over a PSN.

Pseudo Wire PDU

A PDU sent on the PW that contains all of (PW-PDU) the data and control information necessary to emulate the desired service.

PSN Tunnel

A tunnel across a PSN, inside which one or more PWs can be carried.

PSN Tunnel Signaling.

Used to set up, maintain, and tear down the underlying PSN tunnel.

PW Demultiplexer

Data-plane method of identifying a PW terminating at a PE.

Time Division Multiplexing (TDM)

Frequently used to refer to the synchronous bit streams at rates defined by G.702.

Tunnel

A method of transparently carrying information over a network.

References

1. Martini, L., El-Aawar, N., Heron, G., Vlachos, D.S., Tappan, D., Rosen, E.C., Vogelsang, S., et al., “Encapsulation Methods for Transport of Layer 2 Frames Over MPLS,” IETF RFC 4905, Nov. 2000.
2. Martini, L., Rosen, E., El-Aawar, N., Smith, T. and Heron, G., “Pseudowire Setup and Maintenance Using the Label Distribution Protocol

(LDP)," IETF RFC 4447, Apr. 2006.

3. Martini, L., Rosen, E. and El-Aawar, N., "Transport of Layer 2 Frames Over MPLS," IETF RFC 4906, June 2007.
4. Townsley, W., Valencia, A., Rubens, A., Pall, G., Zorn, G. and Palter, B., "Layer Two Tunneling Protocol (L2TP)," IETF RFC 2661, Aug. 1999.
5. Simpson, W., "The Point-to-Point Protocol (PPP)," IETF RFC 1661, July 1994.
6. Luo, W., "Layer 2 Virtual Private Network (L2VPN) Extensions for Layer 2 Tunneling Protocol (L2TP)," IETF RFC 4667, Sep. 2006.
7. Bryant, S. and Pate, P., "Pseudo Wire Emulation Edge-to-Edge (PWE3) Architecture," IETF RFC 3985, March 2005.

Chapter 9

Virtual Private Wired Service

Layer-2 virtual private network over transport can provide several features such as redundancy against failures and controlled connectivity through packet-switching or Generalized Multiprotocol Label Switching (GMPLS). Multiple Layer-2 access protocols can be virtually emulated. This chapter introduces point-to-point virtual connectivity and virtual broadcast access connectivity, such as virtual local area networks (LANs).

9.1 Types of Private Wire Services

Several types of private wire services can be implemented over an Internet Protocol (IP) packet-switched network (PSN) or GMPLS network. These are:

- Virtual private LAN service (VPLS): A Layer-2 service that emulates an Ethernet virtual LAN (Ethernet VLAN) across an IP or an MPLS-enabled IP PSN.
- Virtual private wire service (VPWS): A Layer-2 service that provides point-to-point connectivity for a variety of link layers, including Frame Relay, Asynchronous Transfer Mode (ATM), Ethernet, and PPP, across an IP or MPLS-enabled IP PSN.
- Virtual private multicast service (VPMS): A Layer-2 service that provides point-to-multipoint connectivity for a variety of link layers, including Frame Relay, ATM, Ethernet, and PPP, across an IP or MPLS-enabled IP PSN.
- IP-only L2VPN: A point-to-point or point-to-multipoint IP-only service over an IP or MPLS-enabled PSN. This service is similar to VPWS because it supports a variety of link-layer protocols on attachment circuits, including Frame Relay, ATM, Ethernet, and PPP.

These are described in more detail in the remainder of this chapter.

9.1.1 Layer-2 Virtual Private Services: Wide Area Networks and Local Area Networks

Layer-2 virtual private network, as the name indicates, is a Layer-2 service that emulates an Ethernet VLAN across an IP or MPLS-enabled IP PSN [1].

A virtual private LAN service is a way to provide Ethernet-based multipoint-to-multipoint communication over IP/MPLS networks. It allows geographically dispersed networks, LANS and WANS, to share an Ethernet broadcast domain by connecting sites through pseudo wires. This connectivity would make the networks look as if they were connected using a LAN. The technologies that can be used as pseudo wire include Ethernet over MPLS [2], Layer-2 Tunneling Protocol version 3 (L2TPv3), or Generic Routing Encapsulation (GRE).

The functions of VPLS include the following:

- VPLS emulates a LAN over an MPLS network.
- Sets up MPLS tunnel between every pair of provider edge devices (PEs).
- Sets up Ethernet pseudowires (PWs) inside tunnels for each VPN [3] instance.
- Makes customer edge (CE) devices appear to be connected by a single LAN.
- PE must know where to send the Ethernet frames as an Ethernet bridge does.

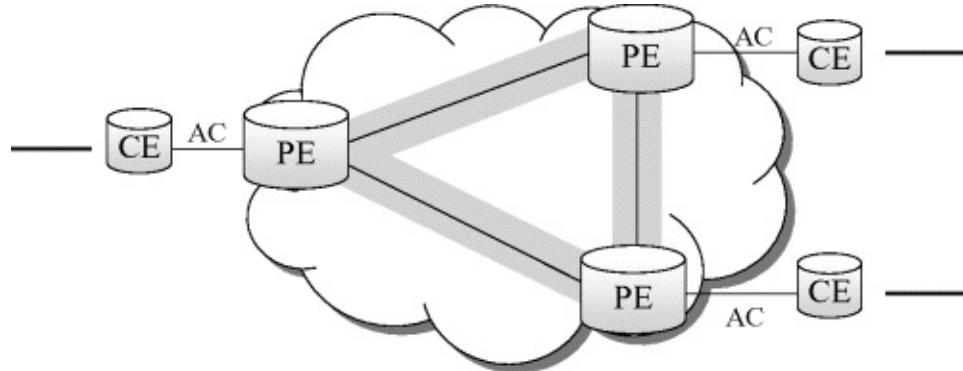
A VPLS module performs the following functions:

- Signaling establishes PWs between PEs per VPLS.
- Autodiscovery locates PEs participating in VPLS instance.
- Obtains frame from bridge, encapsulates Ethernet frames, and injects packet into PW.
- Retrieves packet from PW, removes PW encapsulation, and forwards Ethernet frame to bridge.

[Figure 9.1](#) shows an example of network topology with VPLS, where multiple CEs are connected through a PSN, but they are in a VLAN. The different LAN functions must then be provided by VPLS. The implementation of VPLS is described in two different approaches, mainly derived by the need to perform site (address) discovery and routing, where the selection of the approach can be based on different infrastructure. These

are found in RFC4761 [4] and RFC4762 [1]. These two approaches are not compatible with each other and their adoption is mutually exclusive. This chapter aims to describe the mechanism in RFC4762.

Figure 9.1 Network topology in VPLS.



Unlike Border Gateway Protocol (BGP) VPNs [5], reachability and address information is not advertised and distributed via a control plane. Reachability is obtained by standard learning bridge functions in the data plane. This means that a large number of addresses may be learned automatically. For performance and security reasons, it may be necessary to remove LAN addresses, which are called media access control (MAC) addresses in the remainder of this chapter. Removing or unlearning a MAC address can be achieved by using a Label Distribution Protocol (LDP) Address Withdraw message that contains a new list. Its format is shown in [Figure 9.2](#).

Figure 9.2 Format of Address Withdraw message [1] used in VPLS to remove MAC addresses.

0	1	2	16	31			
U	F		Length	Type			
MAC Address 1							
MAC Address 1		MAC Address 2					
MAC Address 2							
...							
MAC Address <i>n</i>							
MAC Address <i>n</i>							

The fields of the list for the LDP Address Withdraw message are as follows:

- *U bit*: Unknown bit. This bit must be set to 1 to be active. If the MAC address format is not understood, then the list of addresses is ignored.
- *F bit*: Forward bit. This bit must be set to 0. Since the LDP mechanism used here is targeted, the list must not be forwarded.
- *Type*: Type field. This field must be set to 0x0404. This identifies the type-length-value (TLV) as MAC List TLV.
- *Length*: Length field. This field specifies the total length in octets of the MAC addresses in the TLV. The length must be a multiple of 6.
- *MAC Address*: The MAC address(es) being removed. The MAC Address Withdraw message contains an FEC TLV (to identify the VPLS affected), a MAC Address TLV, and optional parameters. No optional parameters have been defined for the MAC Address Withdraw signaling.

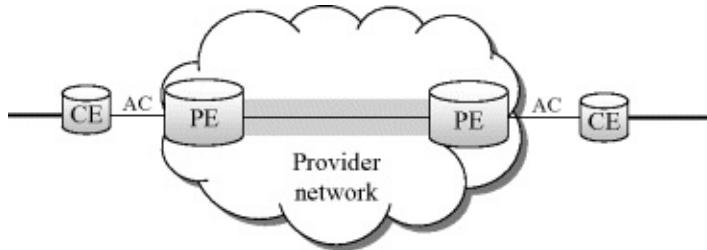
PE routers are assumed to have the capability to establish transport tunnels. Tunnels are set up between PEs to aggregate traffic, and PWs are signaled to demultiplex encapsulated Ethernet frames from multiple VPLS instances that traverse the transport tunnels.

9.1.2 Virtual Private Wire Service

Virtual private wire service is a Layer-2 service [6] that provides point-to-point connectivity for a variety of data link layers (and Layer-3 integration), including Frame Relay, ATM, Ethernet, and PPP, across an IP or MPLS-enabled IP PSN.

A VPWS L2VPN employs Layer-2 services over MPLS to build a topology of point-to-point connections that connect end-customer sites in a VPN. These L2VPNs provide an alternative to private networks that have been provisioned by means of dedicated leased lines or by means of Layer-2 virtual circuits that employ ATM or Frame Relay. The service provisioned with these L2VPNs is known as VPWS ([Figure 9.3](#)).

Figure 9.3 Example of a VPWS service.



The functions of VPWS include the following:

- Runs as a Layer-2 point-to-point service.
- Emulates a wire supporting the Ethernet physical layer.
- Sets up MPLS tunnel between PEs.
- Sets up Ethernet PW inside tunnel.
- Makes CEs appear to be connected by a single Layer-2 circuit.
- Can also make VPWS for ATM or Frame Relay.

Existing protocols that implement VPWS are private (Cisco), and are known as AToM and L2TPv3.

9.1.3 Virtual Private Multicast Service

Today, many kinds of IP multicast services are becoming available. Customers would often like to operate their multicast applications to remote sites over their Layer-2 VPN service, and particular over VPLS. Virtual private multicast service is a Layer-2 service that provides point-to-multipoint connectivity for a variety of link layers, including Frame Relay, ATM, Ethernet, and PPP, across an IP or MPLS-enabled IP PSN [7].

As discussed in the preceding section, VPLS has a shortcoming regarding multicast scalability due to the replication mechanisms intrinsic to the original architecture.

9.1.3.1 Multicast Scalability

In VPLS, replication occurs at an ingress PE (in the hierarchical VPLS [H-VPLS] case, at N-PE) when a CE sends (1) broadcast, (2) multicast, or (3) unknown destination unicast. There are two well-known issues with this approach.

- *Replication to nonmember site:* In cases (1) and (3), the upstream PE has to transmit packets to all of the downstream PEs that belong to the common VPLS instance. The number of members cannot be decreased, so this is basically an inevitable situation for most VPLS

deployments. In case (2) the issue is multicast traffic sent to sites with no members. Usually this is caused when the upstream PE does not maintain downstream membership information. The upstream PE simply floods frames to all downstream PEs, and the downstream PEs forward them to directly connected CEs; however, those CEs might not be members of any multicast group.

- *Replication of PWs on shared physical path:* In VPLS, a virtual switch instance (VSI) associated with each VPLS instance behaves as a logical emulated bridge that can transport Ethernet across the PSN backbone using PWs. In principle, PWs are designed for unicast traffic. In cases (1), (2), and (3), Ethernet frames are replicated on one or more PWs that belong to that VSI. This replication is often inefficient in terms of bandwidth usage if those PWs are traversing shared physical links in the backbone.

9.1.3.2 Multicast Packet Types

Ethernet multicast is used for conveying Layer-3 multicast data. When IP multicast is encapsulated by an Ethernet frame, the IP multicast group address is mapped to the Ethernet destination MAC address. In IPv4, the mapping uses the lower 23 bits of the (32-bit) IPv4 multicast address and places them as the lower 23 bits of a destination MAC address with the fixed header of 01-00-5E in hex. Since this mapping is ambiguous (i.e., there is a multiplicity of one Ethernet address to 32 IPv4 addresses). MAC-based forwarding is not ideal for IP multicast because some hosts might possibly receive packets they are not interested in, which is inefficient in traffic delivery and has an impact on security.

Ethernet multicast is also used for Layer-2 control frames. For example, Bridge Protocol Data Unit (BPDU) for IEEE 802.1D Spanning Trees uses a multicast destination MAC address (01-80-C2-00-00-00). Also, some of the IEEE 802.1ag Connectivity Fault Management (CFM) messages use a multicast destination MAC address that is dependent on their message type and application. From the perspective of IP multicast, however, it is necessary in VPLS to flood such control frames to all participating CEs, without requiring any membership controls.

9.1.4 IP-Only Layer-2 Virtual Private Network

This is a point-to-point or point-to-multipoint IP-only service over an IP or MPLS-enabled PSN. It is similar to VPWS in that it supports a variety of link-layer protocols on the ACs, including Frame Relay, ATM, Ethernet, and PPP.

The interconnected systems may be LAN switches. Furthermore, using IP hosts or IP routers, certain simplifications to the operation of the VPLS can be possible. This simplified type of VPLS has been called an IP-only LAN service (IPLS) in recent IETF drafts. In an IPLS, as in a VPLS, LAN interfaces are run in promiscuous mode, and frames are forwarded based on their destination MAC addresses. However, the maintenance of the MAC forwarding tables is done via signaling rather than via the MAC address learning procedures specified in IEEE 802.1D. IPLS is an example of a Layer-2 and Layer-3 VPN, as interlayer crossing of protocols is used.

Functions of IP-only L2VPN include the following:

- IP-only L2VPNs are different from both VPLS and VPWS because unicast Layer-2 frames containing IP data packets, either IPv4 or IPv6, are de-encapsulated leaving only the IP data packet to be transmitted over the PSN.
- An IP-only L2VPN service also differs from L3VPN service since no routing protocol operates between the PE and CE. Furthermore, connectivity from CE to CE is provided via an emulated Layer-2 service over the PSN, which results in the CEs appearing to be directly attached to each other at Layer 2. The WG will address two specific types of IP-only L2VPN:
 1. Those with ACs that use the same Layer-2 framing at all attachment points in the same L2VPN.
 2. Those with ACs that use different Layer-2 framing at various attachment points in the same L2VPN.

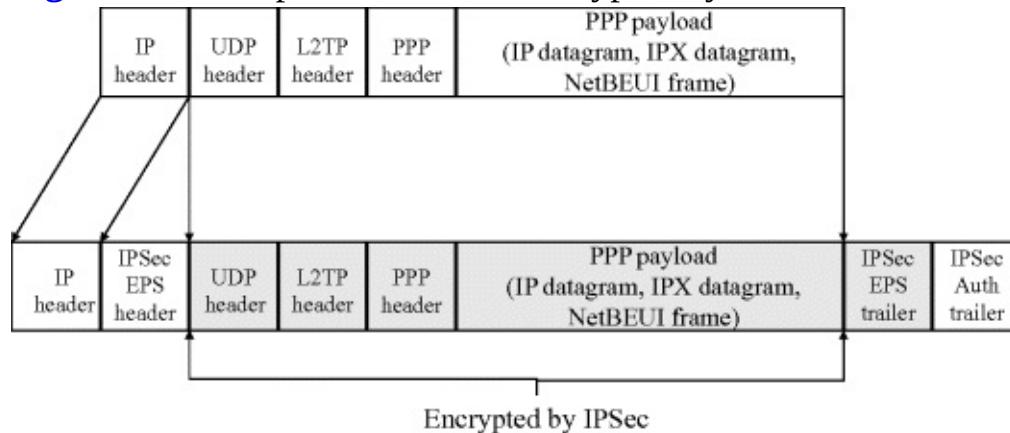
Regarding the second type, interworking between link layers is strictly beyond the scope of what is minimally necessary to ensure IP packets are transported from an AC of one type, across the IP or MPLS-enabled IP PSN, and to an AC of another type in as transparent a manner as possible to the CEs on both sides of the service.

9.1.5 Internet Protocol Security

IP Security (IPSec) Protocol has the objective to protect data transmitted through different networks (with different administrations) from eavesdropping and modification. Both communicating parties have to be authenticated (i.e., the identity of each party must be ensured to be the intended communicating party) and once communication starts, data has to be kept private from any other users sharing the same infrastructure networks. IPSec has two operation modes: tunnel mode, in which it provides its own tunnels, and transport mode, in which it provides encryption and authentication on tunnels created some other way (or on real network links). IPSec is the probable long-term direction for tunnels and secure data transmission in general due to its (intended) interoperability and evolution toward an Internet standard. However, that interoperability is so far still hit or miss—mostly miss.

IPSec communication has access controls in addition to encryption and authentication. Whereas a normal network connection will transmit anything it's asked to transmit, an IPSec tunnel will only transmit what its configuration specifies ([Figure 9.4](#)). This makes it considerably more complex to use than other types of tunnels.

Figure 9.4 Example of the fields encrypted by IPSec.



9.1.5.1 Authentication Header

Authentication Header (AH) is a member of the IPSec Protocol suite ([Figure 9.5](#)). AH guarantees connectionless integrity and data origin authentication of IP packets.

Figure 9.5 Format of the Authentication Header in IPSec.

Authentication Header Format			
0	8	16	31
Next Header	Payload Len	Reserved	
Security Parameter Index (SPI)			
Sequence Number			
Integrity Check Value (ICV)			

IPSec/Point-to-Point Tunneling Protocol (PPTP) tunnels use the same Microsoft Point-to-Point Encryption (MPPE)/PPTP tunnel protocol as Microsoft Point-to-Point Encryption (MPPE)/PPTP tunnels but with IPSec encryption. Windows 2000 includes support for PPTP. IPSec/Layer-2 Tunneling Protocol tunnels use L2TP to establish the tunnel and then run IPSec encryption on it. L2TP is very similar to PPTP but has a multi vendor origin. Windows 2000 includes L2TP, but older Windows versions do not. An open-source L2TP implementation is available on the net for Linux and BSD UNIX, is simple to build, and requires no kernel modification. However, Linux and BSD do need IPSec built into the kernel. L2TP/IPSec isn't a tunnel type, but it is a different way of using L2TP and IPSec together that has become common and is described in RFC 3193. The use of RFC3193 is analogous to using a conduit to run wires through a hazardous area; The Internet is the hazard zone, an ordinary IPSec tunnel is the conduit, and L2TP tunnels are the otherwise unprotected wires. IPSec/Generic Routing Encapsulation (GRE) tunnels layer IPSec directly onto plain GRE tunnels as mentioned under GRE and IP/IP tunnels.

9.2 Generic Routing Encapsulation

Generic Routing Encapsulation [8, 9, 10] is a tunneling protocol developed by Cisco Systems that can encapsulate a wide variety of network layer protocol packet types inside IP tunnels, creating a virtual point-to-point link to various brands of routers at remote points over an IP internetwork ([Figure 9.6](#)).

Figure 9.6 Header structure of a GRE packet.

0	1	2	3	4	5	8	13	16	31		
C	R	K	S	s	Recur	Flags	Version	Protocol Type			
Checksum (optional)					Offset (optional)						
Key (optional)											
Sequence Number (optional)											
Routing (optional)											

The functions of GRE are as follows:

- Provides low overhead tunneling (often between two private networks).
- Does not provide encryption.
- Used to encapsulate an arbitrary layer protocol over another arbitrary layer protocol: delivery header + GRE header + payload packet.
- IPv4 is generally the delivery mechanism for GRE with any arbitrary protocol nested inside (e.g., IP protocol type 47: GRE packets using IPv4 headers).

9.3 Layer-2 Tunneling Protocol

In computer networking, Layer-2 Tunneling Protocol (L2TP) is a tunneling protocol used to support virtual private networks (VPNs). It does not provide any encryption or confidentiality by itself; it relies on an encryption protocol that it passes within the tunnel to provide privacy.

The entire L2TP packet, including payload and L2TP header, is sent within a UDP datagram. It is common to carry Point-to-Point Protocol (PPP) sessions within an L2TP tunnel. L2TP does not provide confidentiality or strong authentication by itself. IPSec is often used to secure L2TP packets by providing confidentiality, authentication, and integrity. The combination of these two protocols is generally known as L2TP/IPSec. The two endpoints of an L2TP tunnel are called the L2TP Access Concentrator (LAC) and the LNS (L2TP Network Server). The LAC is the initiator of the tunnel, while the LNS is the server that waits for new tunnels. Once a tunnel is established, the network traffic between the peers is bidirectional. To be useful for networking, higher-level protocols are then run through the L2TP tunnel. To

facilitate this, an L2TP session (or call) is established within the tunnel for each higher-level protocol such as PPP. Either the LAC or LNS may initiate sessions. The traffic for each session is isolated by L2TP, so it is possible to set up multiple virtual networks across a single tunnel. MTU should be considered when implementing L2TP. The packets exchanged within an L2TP tunnel are categorized as either control packets or data packets. L2TP provides reliability features for the control packets, but no reliability for data packets. Reliability, if desired, must be provided by the nested protocols running within each session of the L2TP tunnel.

An L2TP packet consists of the parts shown in [Figure 9.7](#).

Figure 9.7 Format of an L2TP packet.

0	16	31
Flags and Version Info	Length (optional)	
Tunnel ID	Session ID	
Ns(optional)	Nr(optional)	
Offset Size (optional)	Offset Pad (optional).....	
Payload Data		

9.4 Layer-3 Virtual Private Network 2547bis, Virtual Router

A Layer-3 VPN (L3VPN) interconnects sets of hosts and routers based on Layer-3 (e.g., IP) addresses. The specifications of L3VPN are based on standard operations of VPNs, and in particular, how to manage objects to configure and/or monitor Multiprotocol Label Switching (MPLS) L3VPNs on an MPLS label switching router (LSR) supporting this feature.

The method to provide a VPN in an MPLS network uses a *peer model*, in which the customers' edge routers (CE routers) send their routes to the service providers' edge routers (PE routers). The Border Gateway Protocol (BGP) is then used by the service provider to exchange the routes of a particular VPN among the PE routers that are attached to that VPN. Routes from different VPNs remain distinct and separate, even if two VPNs have an overlapping address space. The PE routers distribute to the CE routers in a particular VPN the routes from other CE routers in that VPN. The CE routers

do not peer with each other, hence there is no overlay visible to the VPN's routing. The term *IP* in IP VPN is used to indicate that the PE receives IP datagrams from the CE, examines their IP headers, and routes them accordingly.

Each route within a VPN is assigned an MPLS label. When BGP distributes a VPN route, it also distributes an MPLS label for that route. Before a customer data packet travels across the service providers backbone, it is encapsulated with the MPLS label that corresponds, in the customers VPN, to the route that is the best match to the packet's destination address. This MPLS packet is further encapsulated (e.g., with another MPLS label or with an IP or GRE tunnel header) so that it gets tunneled across the backbone to the proper PE router. Thus, the backbone core routers do not need to know the VPN routes [5].

The functions of BGP MPLS VPNs (2547bis) are as follows:

- Presently the most popular provider-managed VPN.
- Originally specified in RFC 2547 and updated in (draft) RFC 4364.
- Transports IPv4 and IPv6 traffic in MPLS tunnels.
- Uses BGP for route distribution because service providers commonly use BGP for routing.

Objectives are listed as follows:

- Customer router (CE) has an IP peering connection with PE/edge router in MPLS network
 - IP routing/forwarding across PE-CE link.
- MPLS VPN network responsible for distributing routing information to remote VPN sites
 - MPLS VPN part of customer IP routing domain.
- MPLS VPNs enable full-mesh, hub-and-spoke, and hybrid connectivity among connected CE sites.
- MPLS VPN service in networks only requires VPN configuration at edge/PE nodes
 - Connectivity in core automatically established via BGP signaling.

MPLS L3VPN technology components include the following:

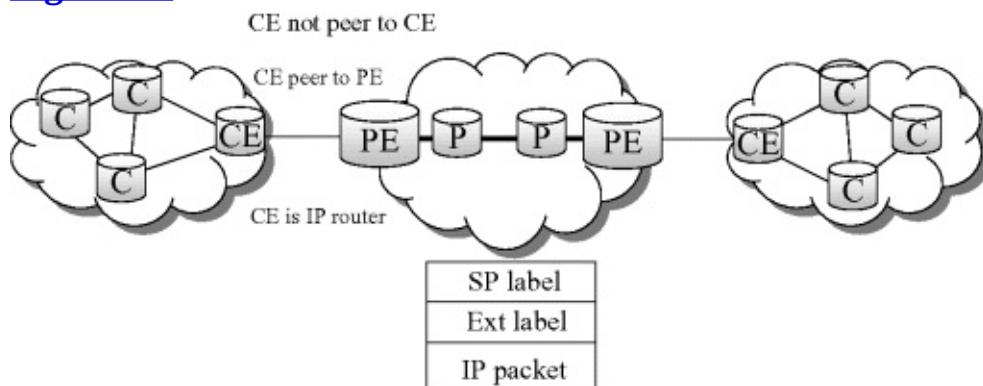
- PE-CE link
 - Can be any type of Layer-2 connection (e.g., Frame Relay,

Ethernet).

- CE configured to route IP traffic to and from adjacent PE router.
- Variety of routing options (e.g., static routes, eBGP, Open Shortest Path First [OSPF], Intermediate System to Intermediate System [IS-IS]).
- MPLS L3VPN Control Plane
 - Separation of customer routing via VPN routing table.
 - In PE router: customer interfaces (I/Fs) connected to virtual routing table.
 - Between PE routers: customer routes exchanged via BGP.
- MPLS L3VPN Forwarding Plane
 - Separation of customer VPN traffic via additional VPN label.
 - VPN label used by receiving PE to identify VPN routing table.

A virtual router (VR), as shown in [Figure 9.8](#), is a PE-based VPN approach in which the PE router maintains a complete logical router for each VPN that it supports. Each logical router maintains a unique forwarding table and executes a unique instance of the routing protocol.

Figure 9.8 Model of a virtual router.



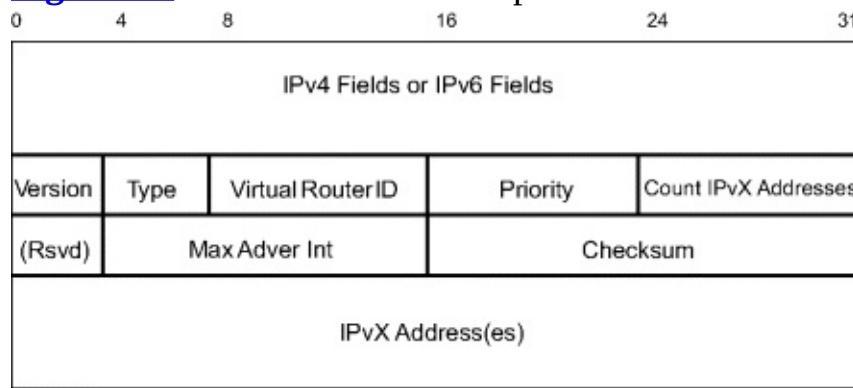
9.4.1 Virtual Router Redundancy Protocol

Virtual Router Redundancy Protocol (VRRP) (RFC 5798) specifies an election protocol that dynamically assigns responsibility for a virtual router to one of the VRRP routers on a LAN. VRRP is designed to eliminate the single point of failure inherent in the static default routed environment. The VRRP router controlling the IPv4 or IPv6 address(es) associated with a virtual router is called the master, and it forwards packets sent to these IPv4 or IPv6

addresses. VRRP master routers are configured with virtual IPv4 or IPv6 addresses, and VRRP backup routers infer the address family of the virtual addresses being carried based on the transport protocol. Within a VRRP router, the virtual routers in each of the IPv4 and IPv6 address families are a domain unto themselves and do not overlap. The election process provides dynamic failover in the forwarding responsibility should the Master become unavailable. For IPv4, the advantage gained from using VRRP is a higher-availability default path without requiring configuration of dynamic routing or router discovery protocols on every end-host. For IPv6, the advantage gained from using VRRP for IPv6 is a quicker switchover to backup routers than can be obtained with standard IPv6 Neighbor Discovery mechanisms.

The purpose of a VRRP packet is to communicate to all VRRP routers the priority and the state of the master router associated with the virtual router ID (VRID). When VRRP is protecting an IPv4 address, VRRP packets are sent encapsulated in IPv4 packets. They are sent to the IPv4 multicast address assigned to VRRP. When VRRP is protecting an IPv6 address, VRRP packets are sent encapsulated in IPv6 packets. They are sent to the IPv6 multicast address assigned to VRRP. The format of the VRRP information packet is shown in [Figure 9.9](#).

Figure 9.9 Format of the VRRP packet.



IPv4 field descriptions are explained as follows:

- *Source Address*: This is the primary IPv4 address of the interface the packet is being sent from.
- *Destination Address*: The IPv4 multicast address as assigned by the IANA for VRRP is 224.0.0.18. This is a link-local scope multicast address. Routers must not forward a datagram with this destination address regardless of its time-to-live (TTL).
- *Time-to-Live (TTL)*: The TTL must be set to 255. A VRRP router

receiving a packet with the TTL not equal to 255 must discard the packet.

- *Protocol*: The IPv4 protocol number assigned by the IANA for VRRP is 112 (decimal).

IPv6 field descriptions are explained as follows:

- *Source Address*: This is the IPv6 link-local address of the interface the packet is being sent from.
- *Destination Address*: The IPv6 multicast address assigned by the IANA for VRRP is FF02:0:0:0:0:0:12. This is a link-local scope multicast address. Routers must not forward a datagram with this destination address regardless of its hop limit.
- *Hop Limit*: The hop limit must be set to 255. A VRRP router receiving a packet with the hop limit not equal to 255 must discard the packet.
- *Next Header*: The IPv6 Next Header protocol assigned by the IANA for VRRP is 112 (decimal).

VRRP field descriptions are explained as follows:

- *Version*: The version field specifies the VRRP protocol version of this packet. The current version is 3, and so is the value of this field.
- *Type*: The type field specifies the type of the VRRP packet. The only packet type defined in this version of the protocol is 1 advertisement. A packet with unknown type must be discarded.
- *Virtual Rtr ID (VRID)*: The Virtual Rtr ID field identifies the virtual router for which the packet is reporting status.
- *Priority*: The priority field specifies the sending VRRP routers priority for the virtual router. Higher values equal higher priority. This field is an 8-bit unsigned integer field. The priority value for the VRRP router that owns the IPvX address (X is either version 4 or 6) associated with the virtual router must be 255 (decimal). VRRP routers backing up a virtual router must use priority values between 1-254 (decimal). The default priority value for VRRP routers backing up a virtual router is 100 (decimal). The priority value zero (0) has special meaning, indicating that the current master router has stopped participating in VRRP. This is used to trigger backup routers to quickly transition to master without having to wait for the current master to time out.

- *Count IPvX Addr*: This is the number of either IPv4 addresses or IPv6 addresses contained in this VRRP advertisement. The minimum value is 1.
- *Rsvd*: This field must be set to zero on transmission and ignored on reception.
- *Maximum Advertisement Interval (Max Adver Int)*: This is a 12-bit field that indicates the time interval (in milliseconds) between advertisements. The default is 1 second.
- *Checksum*: The checksum field is used to detect data corruption in the VRRP message. The checksum is the 16-bit ones complement of the ones complement sum of the entire VRRP message starting with the version field and a pseudo-header. The next header field in the pseudo-header should be set to 112 (decimal) for VRRP. For computing the checksum, the checksum field is set to zero.
- *IPvX Address(es)*: This refers to one or more IPvX addresses associated with the virtual router. The number of addresses included is specified in the *Count IP Addr* field. These fields are used for troubleshooting misconfigured routers. If more than one address is sent, it is recommended that all routers be configured to send these addresses in the same order to make it easier to do this comparison. For IPv4 addresses, this refers to one or more IPv4 addresses that are backed up by the virtual router. For IPv6, the first address must be the IPv6 link-local address associated with the virtual router. This field contains either one or more IPv4 addresses or one or more IPv6 addresses (i.e., IPv4 and IPv6 must not both be carried in one IPvX Address field).

References

1. Lasserre, M. and Kompella, V., “Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling,” IETF RFC 4762, Jan. 2007.
2. Martini, L., Rosen, E., El-Aawar, N. and Heron, G., “Encapsulation Methods for Transport of Ethernet over MPLS Networks,” IETF RFC 4448, Apr. 2006.
3. Andersson, L. and Madsen, T., “Provider Provisioned Virtual Private

- Network (VPN) Terminology,” IETF RFC 4026, Mar. 2005.
- 4.** Kompella, K. and Rekhter, Y., “Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling,” IETF RFC 4761, Jan. 2007.
 - 5.** Rosen, E. and Rekhter, Y., “BGP/MPLS IP Virtual Private Networks (VPNs),” IETF RFC 4364, Feb. 2006.
 - 6.** Andersson, L. and Rosen, E. (Editors), “Framework for Layer 2 Virtual Private Networks (L2VPNs),” IETF RFC 4664, Sep. 2006.
 - 7.** Kamite, Y., Wada, Y., Serbest, Y., Morin, T. and Fang, L., “Requirements for Multicast Support in Virtual Private LAN Services,” IETF RFC 5501, Mar. 2009.
 - 8.** Hanks, S., Li, T., Farinacci, D. and Traina, P., “Generic Routing Encapsulation over IPv4 Networks (INFORMATIONAL),” IETF RFC 1702, Oct. 1994.
 - 9.** Farinacci, D., Li, T., Hanks, S., Meyer, D. and Traina, P., “Generic Routing Encapsulation (GRE) (PROPOSED STANDARD -Updated by RFC 2890),” IETF RFC 2784, Mar. 2000.
 - 10.** Dommety, G., “Key and Sequence Number Extensions to GRE (PROPOSED STANDARD),” IETF RFC 2890, Sep. 2000.

Chapter 10

IP and Optical Networking

The bandwidth explosion ushered in by the popularity of the Internet has spurred the recent acceleration in the development and deployment of equipment supporting packet-based broadband services. This, coupled with the widespread deployment of wavelength-division multiplexing (WDM) – based optical transport systems in the core network to satisfy the corresponding increase in capacity demand, has led network designers to reconsider traditional approaches to provisioning, traffic engineering, and network restoration.

Provisioning: In today's network for high bandwidth services, time is the critical resource that service providers optimize. Time constraints are particularly acute for provisioning end-to-end high-speed broadband services, where provisioning times are often measured in weeks or months and the opportunity costs of delayed revenue recognition and unserved customers are huge. Rapid technologic advances in low-cost, high-capacity optical transport systems have only exacerbated the end-to-end provisioning challenge as carriers install a wide range of point solutions to meet explosive new service demands. There is an emerging consensus that rapid end-to-end provisioning of broadband services over optical transport networks is essential.

Traffic Engineering: Rapid growth and increasing requirements for service quality, reliability, and efficiency have made traffic engineering an essential consideration in the design and operation of Internet Protocol (IP)/optical backbone networks. Recent developments in Generalized Multiprotocol Label Switching (GMPLS) and differentiated services have opened up possibilities to address some of the limitations of the traditional IP/optical networks.

Network Restoration: Existing architectures for network restoration were developed based on the ubiquitous circuit-switched/time-division multiplexing (TDM) network paradigm. As services increasingly migrate from circuit switched/TDM to packet networks, restoration architectures must also evolve to meet the service requirements of this new IP/optical

network paradigm.

The above features are the core of next-generation networks and the key to future success as optical transport and IP services converge. The Internet Engineering Task Force (IETF) and Optical Interworking Forum (OIF) define control plane models aimed at reducing not only provisioning times but also overall network robustness and reliability with network restoration techniques.¹³⁹

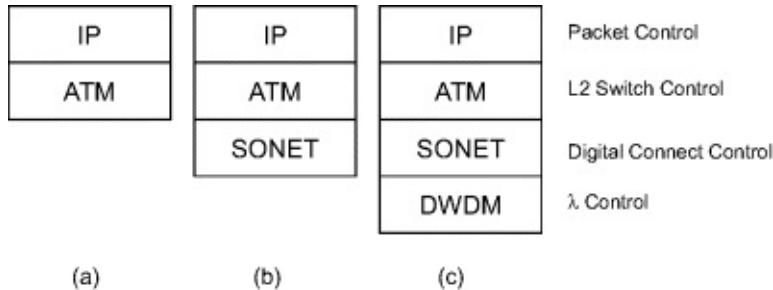
This chapter is organized as follows. The multilayer network evolution is introduced in Section 10.2, followed by a discussion on the limitations in current legacy networks in Section 10.2. Automated provisioning in IP/optical networks is described in Section 10.3. Section 10.4 presents the proposed control plane techniques in the industry, followed by key design requirements for next-generation multilayer IP/optical networks in Section 10.5. Section 10.7 compares the proposed control plane models with the key design requirements from Section 10.6.

10.1 IP/Optical Network Evolution

10.1.1 Where Networking is Today

For years now, the datacom and the telecom networks have existed in different worlds. Having different objectives and customer bases, each discipline has formed its own language, procedures, and standards. Placing data on a telecom network was a challenging and often difficult task. Placing datacom traffic onto a voice network required encapsulating several layers. In [Figure 10.1a](#), we see data traffic stacked on top of an Asynchronous Transfer Mode (ATM) layer. In order to send this traffic on a synchronous optical networking (SONET) network, it was restacked, as shown in [Figure 10.1b](#). Finally, to place this traffic on an optical dense WDM (DWDM) network, it was stacked again, as shown in [Figure 10.1c](#).

Figure 10.1 Layer structures.



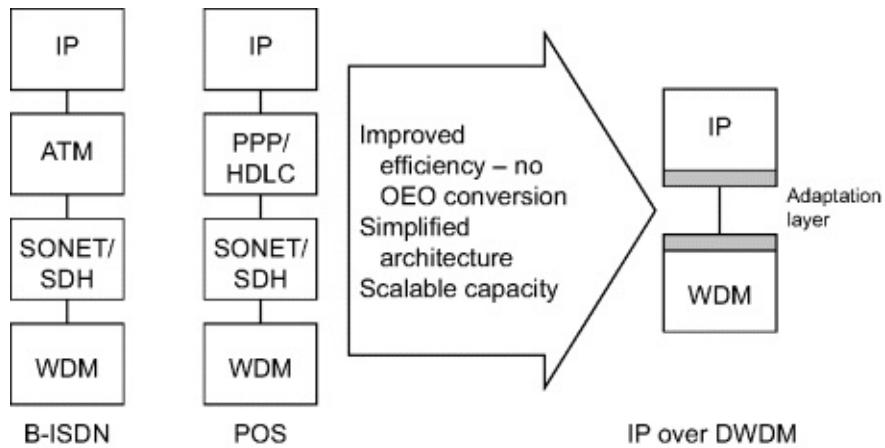
Notice how each layer has its own management and control. This method of passing data onto a telecom network is inefficient and costly. Interfacing between layers requires manual provisioning; each layer is managed separately by different types of service providers. Reducing the number of interface layers promises to reduce overall operational cost and improve packet efficiency. Common control plane concepts, which are defined by the OIF and IETF, promise to fulfill the aspiration of one interface and one centralized automatic control.

As the telecom world marches toward its goal of an all-optical network, data packets may need to cross several different types of networks before being carried by an optical network. These network types include packet-switched networks, Layer-2-switched networks, Lambda-switched networks, and fiber-switched networks.

10.1.2 Where Networking is Going

As shown in [Figure 10.2](#), one approach for sending IP traffic on WDM networks is using a multilayered architecture comprised of IP over ATM over SONET over WDM. This architecture has four management layers. One can also use the packet over SONET (POS) approach, doing away with the ATM layer, by putting IP, Point-to-Point Protocol (PPP), and High-Level Data Link Control (HDLC) into SONET framing. This architecture has three management layers. However, such multilayered architectures come with increased complexity and management costs for service providers. As the network migrates, we will find that layers of this stack will begin to disappear, as shown in [Figure 10.2](#).

Figure 10.2 Evolution of multilayer networks.



One of the main goals of the integration of IP/optical architecture is to make optical channel provisioning driven by IP data paths and traffic engineering mechanisms. This will require tight cooperation of routing and resource management protocols at the two (IP and optical) layers. The multilayered protocol architecture, without tight cooperation, can complicate the timely flow of the possibly large amount of topologic and resource information. Another problem is with respect to survivability. There are various proposals stating that the optical layer itself should provide restoration/and protection capabilities of some form. This will require careful coordination with the mechanisms of the higher layers such as SONET automatic protection switching (APS) and IP rerouting strategies. Hold-off timers have been proposed to inhibit the backup mechanisms of higher layers.

Problems can also arise from the high level of multiplexing. The optical fiber links contain a large number of higher-layer flows such as SONET/synchronous digital hierarchy (SDH), IP flows, or ATM virtual channels (VCs). Since these have their own mechanisms, a flooding of alarm messages can take place. Hence a much closer IP/WDM integration is required.

Over the last 10 years the telecom industry has witnessed a dramatic change with the rising popularity of new applications such as peer-to-peer and real-time multimedia over IP. The high dynamism of traffic patterns will require networks to adapt to new requirements of service providers: “bandwidth on demand, not to forecast.” This paradigm shift has created the need to migrate today's complex overlay network to a new, highly scalable architecture that is simple yet enables rapid service provisioning, dynamic bandwidth management, and flexible service creation. One viable solution for this challenge is to unify all the different services under a common control

plane and automate network resources using OIF optical user network interface (O-UNI) and IETF (GMPLS) techniques.

Specified by OIF, O-UNI is a signaling suite that uses Resource Reservation Protocol (RSVP) signaling to create, modify or delete a light path or SONET/SDH circuit. O-UNI is a client-server approach, where the O-UNI client (in this case, a router in the packet layer is an O-UNI client) requests a light path across optical transport network (OTN) by using an RSVP path request message to the O-UNI network side (Optical cross-connect (OXC) in the OTN acts as an O-UNI network-side server).

Specified by IETF, GMPLS is a new protocol suite that uses advanced network signaling and routing mechanisms to automate set up for end-to-end connections for all types of network traffic (TDM, packet/cell, wavebands and wavelengths). Carrier metro and core networks, given their need to support a diverse range of services using a variety of transport technologies, are an ideal candidate for GMPLS-based control and provisioning. A GMPLS-based network can offer many capabilities that include the following:

- Network resource discovery and routing control
- Dynamic provisioning and traffic engineering for end-to-end connections
- Bandwidth on demand for just-time service creation
- New quality of service (QoS)-defined value-added services
- New granulations in service restoration and protection

10.2 Challenges in Legacy Traditional IP/Optical Networks

10.2.1 Proprietary Network Management Systems

Carriers have large investments in proprietary, centralized, network management systems (NMSs). These NMSs have a view of the network elements (NEs) under their control, including physical topology, fiber routes, connections, the available capacity of network links, switching and termination capabilities of the nodes and interfaces, and the protection properties of the link. In this environment, route determination is a

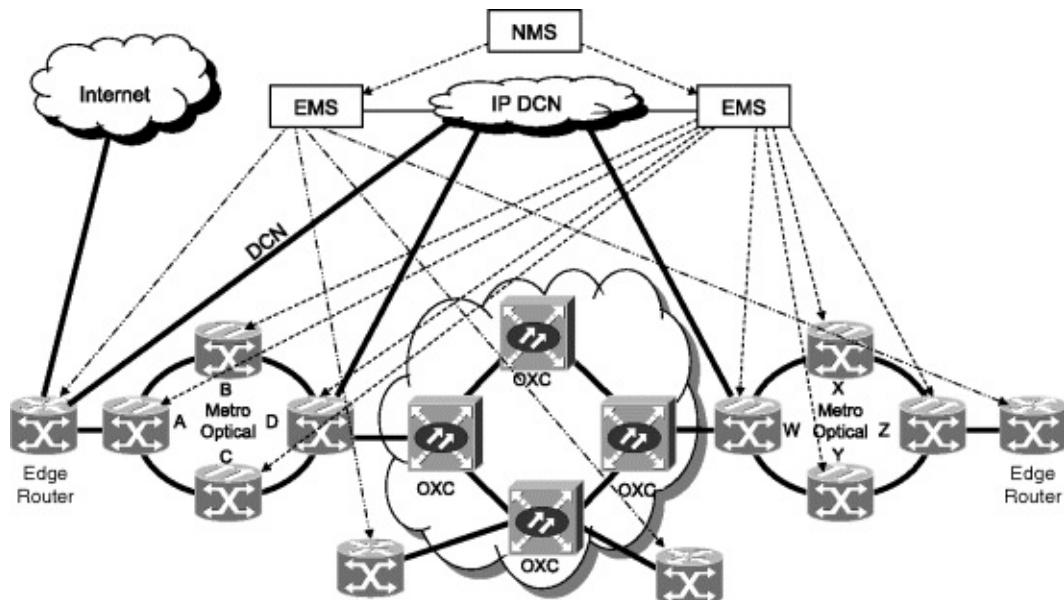
centralized, manual process that considers network efficiency, reliability, and service differentiation. This information is populated manually in the database, so it is subject to human error. These systems were developed to manage a static network and use batch processes and off-line processing. Because of the investment and the extent of NMS connectivity, the cost of change will be enormous. Thus, it will take much more than an incremental improvement in features or costs to prompt carriers to switch to control plane architecture.

Carriers typically develop their NMSs, and element management systems (EMSs) are usually proprietary to equipment vendors. The human operator uses the NMS to issue commands to the NEs involved in a circuit via the equipment vendor's EMS. Standard communication protocols such as Transaction Language 1 (TL1) and Common Object Request Broker Architecture (CORBA), and a centralized management plane that uses Telecommunication Management Network (TMN) or Simple Network Management Protocol (SNMP), permit an exchange of data between the EMSs and NMSs. Very little multivendor interoperability has been achieved for this system and the NMS must be customized for every new system deployed. In addition, the different transport layers (optical, SONET/SDH, IP end-user) are managed separately. As a result, an end-to-end circuit in a multivendor environment involves multiple management systems that don't readily communicate with each other.

10.2.2 Complexity of Provisioning in Legacy IP/Optical Networks

Today's network configurations using EMSs and NMS are static using, as shown in [Figure 10.3](#). Establishing a connection is a process that involves many steps and can take several months. The process is initiated by a series of administrative tasks to request the service. Data entry can be performed by the carrier or, in a few cases, directly by the client. The request is then transferred to planning organization to determine the route and equipment used for the circuit. Simulations may be run to determine the effect of added circuits on the network. Protection routers and facilities are determined according to the service requirements.

[**Figure 10.3**](#) Network provisioning in legacy traditional IP/optical networks.



NMS: Network management system
EMS: Element management system
DCN: Data communication network

The circuit may need to cross multiple domains, which are administrative boundaries within a carrier and between carriers. Each crossing needs to be coordinated. The end-user may need local access facilities, which would be separately planned and installed. This installation could include equipment on the customer's premises and local access wiring and cabling.

Once the planning process is complete and routes and equipment have been selected, the circuits are provisioned using vendor proprietary EMSs and NMSs as described in Section 10.2.1. With equipment that supports point-and-click provisioning, this part of the process can take just a few minutes if capacity is available and card slots are equipped. For example, a network provider builds a network with optical switches and proprietary vendor software that allows it to provision services quickly. For networks built on equipment that requires manual provisioning, this part of the process can take weeks because vehicles may need to be dispatched to nodes across the country. As a result, carriers treat circuit turn up as a batch process, installing circuit packs and turning up and testing groups of circuits to save cost. Finally, the carrier performs end-to-end testing of the circuit before turning it over to the client.

10.3 Automated Provisioning in IP/Optical

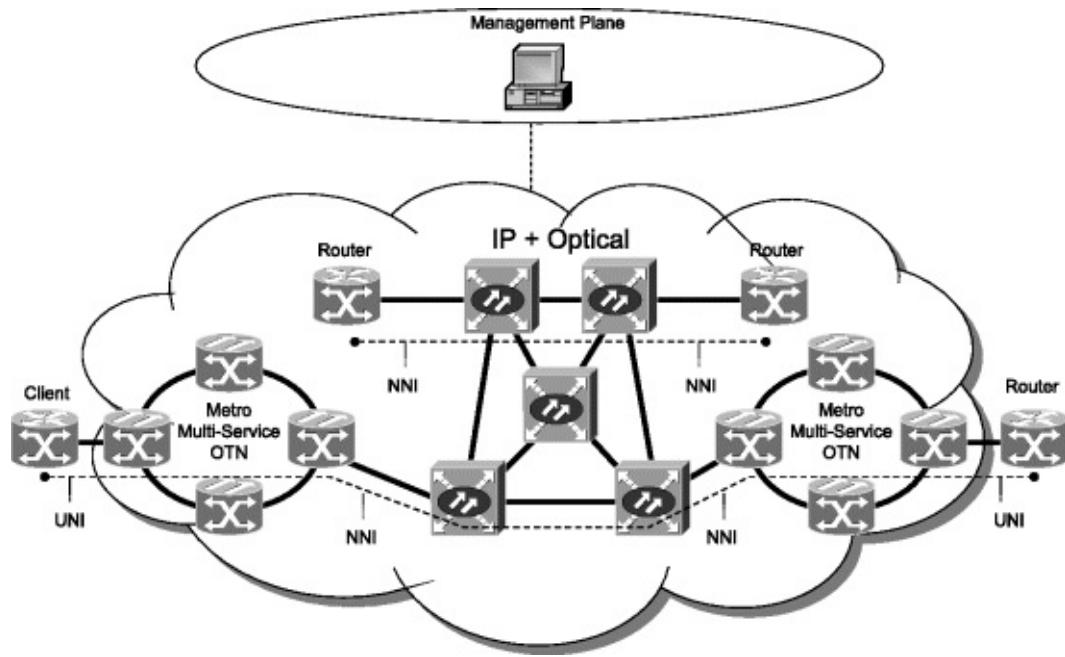
Networks

With networks becoming increasingly difficult to operate, control plane provides the ability to automate many of the network functions that are directly related to the operational complexities. These functions include end-to-end provisioning of services, network resource discovery, bandwidth assignment and service creation. Traffic engineering parameters relating to SONET protection support, available bandwidth, route diversity, and quality of service are distributed throughout the network allowing every node in the network to have full visibility and configuration status of every other node. Ultimately this leads to increased optical network intelligence. Therefore, as service providers introduce new network elements into their networks, add or remove facilities, or turn up new circuits, the control plane will automatically distribute and update the network with the new information. At the time of publication, many of these upgrades and updates are being performed manually and are operationally intensive.

The complexity of current overlay architectures means the provisioning of connections often requires a substantial amount of coordination among operations staff located throughout the network. Capacity is assessed, optimal connection and restoration paths are determined, and the connection must be fully tested once it is established. Automated provisioning using control plane, on the other hand, uses advanced routing (Open Shortest Path First [OSPF], Intermediate System to Intermediate System [IS-IS]) and the RSVP Traffic Engineering (RSVP-TE) signaling protocol to build intelligence into the network such that it is sufficiently self-discovered to dynamically advertise availability or lack of resources throughout the network. With this capability, multi-hop connections with optimal routes and backup paths can be established in a single provisioning step.

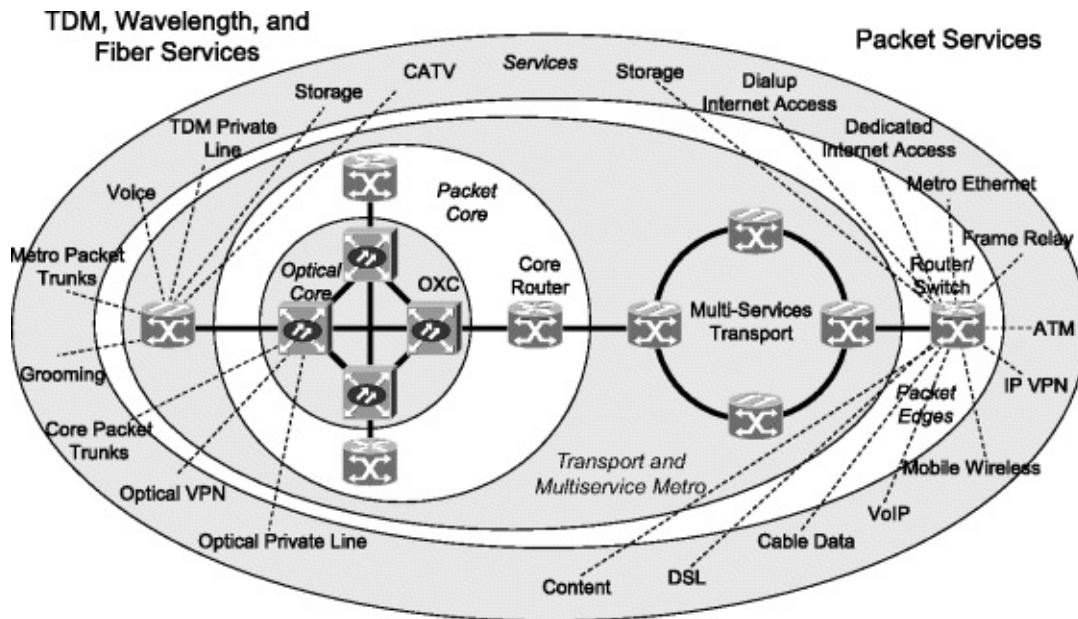
Basically, control plane allows service providers to shift their focus to the creation and acceleration of new services and minimize the large investment currently being made to support operations. [Figure 10.4](#) shows a control plane –based new generation IP/optical network. Using a simple, NMS-based, point-and-click provisioning mechanism, the intelligent optical network will first signal a request for Label Switched Paths (LSPs) and, upon confirmation, dynamically set up the requested path.

[Figure 10.4](#) Control plane–based automated provisioning.



The goal of control plane is to reduce provisioning times from months and days down to minutes and seconds. Certainly, the full potential of control plane protocol is realized when considering an end-to-end network, extending from the access segment to the long-haul core. In this scenario a network element in the access network can signal and set up a path across the metro to the backbone network, which in turn will request a path from the long-haul element and vice versa. With this autonomous architecture, service providers not only increase the service agility of their own networks, they can also solve multivendor interconnection problems when connecting different network segments or to other carrier networks. For example, a metro local exchange carrier (LEC) would rely on a long distance interexchange carrier (IXC) partner for long-distance services as shown in [Figure 10.5](#).

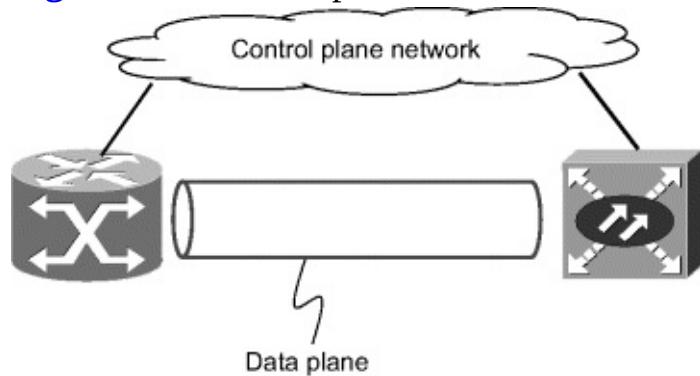
Figure 10.5 End-to-end control plane–based provisioning.



10.4 Control Plane Models for IP/Optical Networking

In optical networks and interworking networks between IP and optical networks, a control plane is logically separated, as shown in [Figure 10.6](#). In only IP networks, control packets are transmitted through the same interface as one through which data are transmitted. However, interfaces of optical network elements in the data plane may not be capable of handing packets. To control optical network elements, interfaces that process control packets are needed. The control plane can be constructed by using IP networks.

[Figure 10.6](#) Control plane.

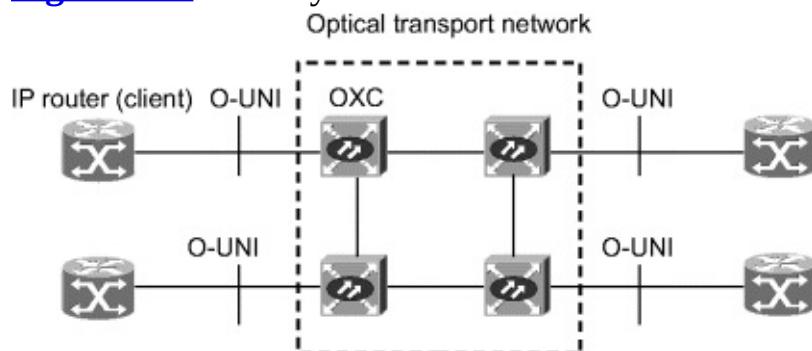


10.4.1 Optical Internetworking Forum's Optical

User Network Interface: Overlay Model

In an overlay network architecture defined by OIF using an O-UNI [1], there is a clear boundary between the client and the optical transport OTN, as shown in [Figure 10.7](#). Routing and topologic information does not cross this boundary in the sense that each layer is running its own instance of a routing protocol (e.g., OSPF or entirely different routing protocols). Network clients request network services (e.g., connections across a well-defined interface). This interface is generally called the optical user network interface. The client side of the O-UNI is known as UNI-C (e.g., IP/packet domain), while the term UNI-N has been used to identify the network side (optical/WDM) of the O-UNI. One or more signaling channels exist between the UNI-C and UNI-N. The UNI control channel must support the transport of RSVP signaling protocols.

[Figure 10.7](#) Overlay model.



The four actions described as follows are provided at the O-UNI.

- *Connection Create.* Creates an end-to-end connection across the OTN with specified connection attributes such as bandwidth and diversity.
- *Connection Delete.* Deletes an already existing connection.
- *Connection Modify.* Modifies one or more of the connection attributes of the existing connections.
- *Status Enquiry.* Allows the client to enquire about the status of a connection or a group of connections.

10.4.2 Internet Engineering Task Force's Generalized Multiprotocol Label Switching: Peer Model

GMPLS has evolved from MPLS, the original IETF standard intended to enhance the forwarding performance and traffic engineering intelligence of packet-based (ATM, IP) networks [2]. In an MPLS network, a label, which is attached to the top of each packet, is uniquely defined for each link. The series of assigned labels makes up an LSP tunnel. The MPLS lies between the data link layer and the network layer. GMPLS extends the MPLS label concept for other layers, as shown in [Figure 10.8](#). Time slots, wavelengths, and fibers are also considered labels, so that GMPLS is not only packet-switch capable (PSC), but also TDM-switch capable fiber-switch capable (FSC) and lambda-switch capable (LSC). Therefore unlike MPLS, which is supported mainly by routers and Layer-2 switches, GMPLS can be supported by a variety of optical platforms including SONET add/drop multiplexers (ADMs), optical cross-connects (OXCs), and WDM systems. This will allow an entire infrastructure, extending from the access network to the core network, to utilize a common control plane. The GMPLS signaling and routing protocols were developed by extending the MPLS protocols [3, 4, 5]. [Table 10.1](#) summarizes the characteristics of GMPLS.

Figure 10.8 GMPLS labels.

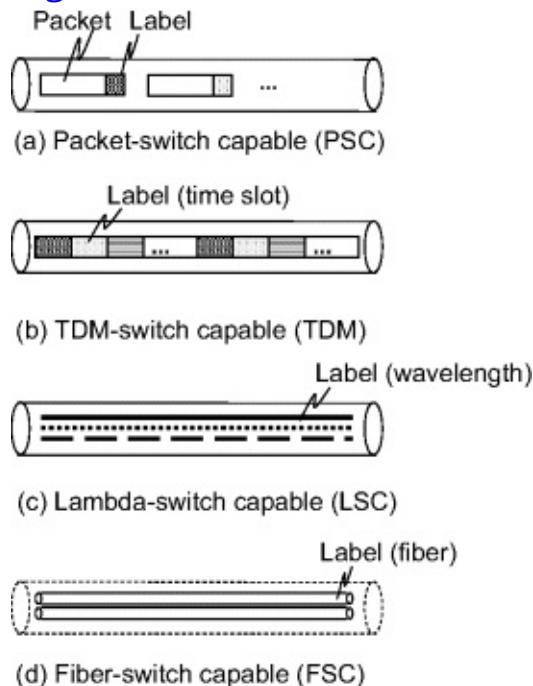


Table 10.1 GMPLS Characteristics.

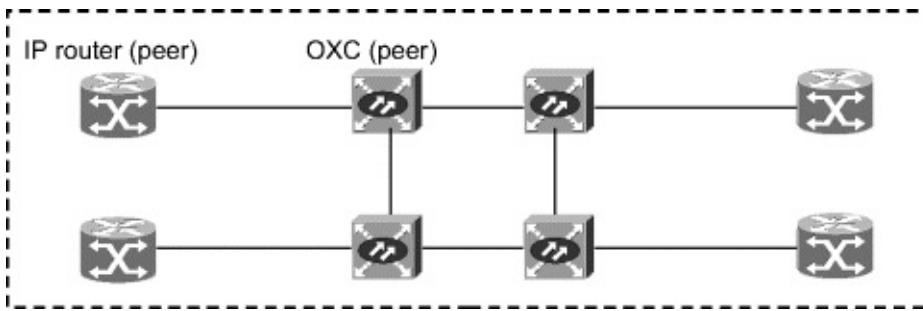
Switching Domain	Traffic Type	Forwarding Scheme	Network Element(s)	Nomenclature
Packet; cell	IP; ATM	Label as Shim Header; VCC	IP router; ATM switch	Packet-switch capable (PSC)
Time	SONET/SDH	Time slot in repeating cycle	Digital cross-connect system (DCS); add/drop multiplexers (ADM)	TDM-switch capable (TDM)
Wavelength	Transparent	Lambda	Optical cross-connect (OXC); dense wave division multiplexer (DWDM)	Lambda-switch capable (LSC)
Physical space	Transparent	Fiber; line	Optical cross-connect (OXC)	Fiber-switch capable (FSC)

Regarding connection set up, while MPLS requires a Label Switched Path (LSP) between two-end point devices, GMPLS extends this concept beyond simple point-to-point connections. In a GMPLS network it is possible to find and provision end-to-end paths that traverse different networks.

GMPLS also newly introduces a Link Management Protocol (LMP) [6] to manage and maintain the health of the control and data planes between two neighboring nodes. LMP maintains control channel connectivity, verifies the physical connectivity of the data links, correlates the link property information, and localizes link failures.

Under the peer model, as shown in [Figure 10.9](#), the IP control plane acts as a peer of the optical transport network control plane. This implies that a single instance of the control plane is deployed over the IP and optical domains. When there is a single optical network involved and the IP and optical domains belong to the same entity, then a common IGP (e.g., OSPF or IS-IS) with appropriate extensions can be used to distribute topology information over the integrated IP/optical network. In the case of OSPF, opaque LSAs can be used to advertise topology state information. In the case of IS-IS, extended type, length, and value (TLV) will have to be defined to propagate topology state information. Many of these extensions are occurring within the context of GMPLS.

[**Figure 10.9**](#) Peer model.



When an optical interworking network with multiple optical networks is involved (e.g., spanning different administrative domains), a single instance of an intradomain routing protocol is not attractive or even realistic. In this case, interdomain routing and signaling protocols are needed. In either case, a tacit assumption is that a common addressing scheme will be used for the optical and IP networks. A common address space can be trivially realized by using IP addresses in both IP and optical domains. Thus, the optical network elements become IP-addressable entities.

10.5 Next-Generation MultiLayer Network Design Requirements

Much has been written about the merits of an end-to-end control plane for fast connection setup, restoration, and similar functions. However, adapting the GMPLS-based control plane to the existing IP/optical networks has been a complex and challenging task. For successful introduction of GMPLS technology, transition and coexistence of the GMPLS control plane (for optical or WDM layer) with the MPLS control plane (for packet or IP layer) in multilayer IP/optical networks is a key factor.

The following network design principles are key for control plane model adoption in multilayer networks.

- *Administrative separation:* To accommodate the administrative separation between the IP and optical layers, allowing the management of these domains to remain separate and autonomous as per current service provider organizational structure
- *Service virtualization:* Service virtualization, or separation, among IP/MPLS service networks, leading to effective utilization of optical/DWDM network resources
- *Multilayer coordination:* Tighter coordination between the IP and

- optical layers to simplify operations, by dynamic path computation and resource optimization, leading to enablement of future applications such as combined multilayer protection and restoration
- *MPLS and GMPLS interworking:* Success of GMPLS technology depends on the transition and coexistence of GMPLS in the optical layer with existing traditional IP/MPLS service networks in packet layer, along with common path computation across packet and optical domains.

Administrative separation allows service providers to maintain their current operational boundaries or “business as usual,” while optimizing the network hardware. The multilayer coordination requirement takes the next step, as it is important to optimize network resource utilization globally (i.e., taking into account both packet and optical layers), rather than optimizing resource utilization at each layer independently. This supports better network efficiency and scalability.

10.6 Benefits and Challenges in IP/Optical Networking

One of the biggest concerns in networking today is about how to deploy GMPLS technology and realize its benefits while still respecting the administrative boundaries between the IP (packet) and optical (WDM) domains. Control plane models, OIF O-UNI-based overlay, and IETF GMPLS-based peer models are being discussed in the literature as next-generation-networks. The key difference among these models is how much and what kind of network information can be shared between IP and optical domains. The peer model is suitable, while the optical (WDM) and IP (packet) domains are operated by a single entity. Many service providers have traditionally built their own networks, where optical and IP service networks belong to different management and ownership. The overlay model is suitable for such a scenario, but does not offer the benefits of the peer model approach for efficient resource utilization, optimal routing, and protection and restoration between IP and optical layers. A new model, known as the border model, is suitable in this scenario, where optical and IP domains administrated by different entities would like to maintain a separation between IP and optical layers and, at the same time, get the

benefits of the peer model approach.

The overlay and peer models do not satisfy the key design requirements mentioned in Section 10.5. An overlay model, defined by OIF and based on O-UNI, limits the capability between optical and higher (packet)-layer domains for automatic signaling based on RSVP and some management functions. It works well when the automatic signaling function is first introduced into the legacy optical network and is well-suited for scenarios where the user side (packet layer) and network side (optical layer) of O-UNI are under different administrative domains. However, it does not offer any multilayer coordination function between packet (IP) and WDM (optical) layers, as there is no routing protocol on the interface between the user (packet layer) and network (optical layer) side of O-UNI that can be used for sharing routing topology information.

A peer model fully exchanges topology and resource information between multiple (optical and packet) layers within the network, which can make GMPLS difficult to introduce in existing networks that are administered by different organizations with different policies. The peer model is suitable only when both optical and packet layers are under a single administrative domain with a GMPLS control plane. The peer model does not isolate fault in the optical domain from the packet domain, and vice versa. The peer model also assumes that the entire packet and optical network upgrade with GMPLS, which is not an economically beneficial task.

References

1. The Optical Internetworking Forum, “User Network Interface (UNI) 1.0 Signaling Specification –Implementation Agreement,” OIF-UNI-01.0.
2. Mannie, E. (Editor), “Generalized Multi-Protocol Label Switching Architecture,” IETF RFC 3945, Oct. 2004.
3. Berger, L. (Editor), “Generalized Multi-Protocol Label Switching (GMPLS) Signaling Functional Description,” IETF RFC 3471, Jan. 2003.
4. Berger, L. (Editor), “Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions,” IEFT RFC 3473, Jan. 2003.
5. Kompella, K. and Rekhter, Y.Y. (Editors), “OSPF Extensions in Support of

Generalized Multi-Protocol Label Switching (GMPLS)," IETF RFC 4203, Oct. 2005.

6. Lang, J. (Editor), "Link Management Protocol (LMP)," IETF RFC 2404, Oct. 2005.

Chapter 11

IP Version 6

Demand for Internet connectivity is growing quickly. At present, 1.5 billion users access the Internet World Stats [1], with tremendous growth in both the developed world and, especially, in the developing world. This growth is expected to continue. In particular the increasingly popular use of data services in mobile telephony networks has the potential to triple the number of Internet users within the foreseeable future [2, 3]. Machine-to-machine communications are expected to further contribute to the growth [4]. These developments are causing a rapid increase in the number of devices that should be addressable on the Internet.

Unfortunately, the growing demand for addresses is exceedingly difficult to meet with Internet Protocol (IP) version 4 (IPv4). Projections show that by the year of 2012 the pool of unallocated global IPv4 addresses will be exhausted [5]. Given the ceasing supply of IPv4 addresses for new devices, what can be done to enable future growth of the Internet? There are two options:

1. Share existing IPv4 addresses across multiple devices.
2. Introduce additional IP addresses.

Sharing IPv4 addresses has been pursued for a long time by means of IPv4 address translation [6], also known as network address translation. IPv4 would have already run out of addresses if IPv4 address translators had not become ubiquitous. Introducing additional IP addresses, is the main objective of IP version 6 (IPv6). Unfortunately, even though the deployment of IPv6 will in the long term be indispensable to meeting the growing demand for Internet connectivity, deployment as yet has been slow [7, 8, 9].

This chapter describes the main differences between IPv4 and IPv6, and it investigates the reasons for the lagging uptake of IPv6. The chapter explains why, contrary to widespread belief, IPv6's large address space alone is insufficient to drive early IPv6 deployment. The initial need for backwards compatibility with IPv4 will require every device with an IPv6 address to also be reachable at an IPv4 address, thus effectively limiting the number of

devices addressable on the Internet to what is already possible today. The chapter further shows that early IPv6 deployment will mostly be driven by administrative benefits. IPv6's large address space will become a driver at a later stage, when the amount of deployed IPv6 renders backwards compatibility with IPv4 dispensable.

11.1 Addresses in IP Version 6

IPv6 addresses are 128-bit numbers that identify a single network interface or a group of interfaces. In this regard, IP addresses are very similar to link-layer addresses. However, since IP addresses are used for communications beyond one link, they need to be technology independent and more sophisticated in structure so as to support a node in forwarding a packet toward a destination node. [Figure 11.1](#) shows an example of basic IP address structure. The leading bits of the IP address, 64 bits in the example, form the IP address prefix. This specifies the type of IP address and, for some types, also encodes the location of the IP address owner in Internet topology. The remaining bits identify the interface or the group of interfaces to which the IP address belongs. For a given packet that is sent across the Internet, the IP addresses of the source and destination nodes are called IP source address and IP destination address, respectively.

[Figure 11.1](#) Unicast IP address.



[Figure 11.1](#) also demonstrates the textual representation of an IP address. The 128 bits are given in hexadecimal form, with a colon between consecutive blocks of 16 bits. This representation can be simplified by replacing one sequence of zero bits by a double-colon (::). The length of the IP address prefix is denoted as a decimal number that is separated from the IP address by a slash. The prefix itself can then be referred to by setting all nonprefix bits to zero in this notation. This means that the prefix of the IP address in [Figure 11.1](#) is 2001:FDB8:0000:5AFE::/64.

11.1.1 Unicast IP Addresses

Applications that communicate with a single remote node, such as Web browsing, electronic mail, or conventional Internet telephony, need to address exactly one network interface. This is the purpose of a unicast IP address. A unicast IP address identifies the link to which it belongs, thereby localizing the IP address owner in Internet topology, and it specifies a particular interface attached to that link.

Link identification is the purpose of the IP address prefix, which in the case of a unicast IP address is 64 bits long. Each link on the Internet is assigned one or multiple subnet prefixes, which are globally unique numbers of 64 bits length. Nodes that attach to a particular link reuse these subnet prefixes as prefixes for their unicast IP addresses. The remaining 64 bits in a unicast IP address form an interface identifier, which is unique within the scope of a link and thus distinguishes the interface from others on the same link. The IP address shown in [Figure 11.1](#) is a unicast IP address.

For administrative purposes, the scope of a unicast IP address can be limited to the link to which the respective interface attaches. Such a link-local IP address can only be used for packet exchange between nodes on the same link. The subnet prefix of a link-local unicast IP address is set to the prefix FE80:0000:0000:0000::/64, and it does not bear any localization semantics. Link-local IP addresses are primarily used during autoconfiguration when a node gains IP connectivity. To differentiate IP addresses with global scope from link-local IP addresses, the former are also referred to as global IP addresses.

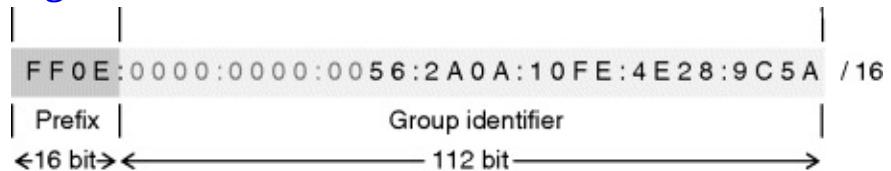
As with link-layer addresses, IP addresses include a universal/local bit and an individual/group bit, with the distinction that the meaning of the former is reversed to match the aforementioned subnet prefix for link-local unicast IP addresses. However, these special bits are not explicitly highlighted in [Figure 11.1](#) due to the more simple, hexadecimal format used.

11.1.2 Multicast IP Addresses

A multicast IP address identifies a group of network interfaces of typically different nodes. Since the interfaces may attach to multiple links, there is no single subnet prefix that a multicast IP address could use. The interface group is instead solely identified by a 112-bit group identifier, which is appended to a 16-bit prefix to form a multicast IP address. The semantics of the universal/local and individual/group bits are the same for unicast and

multicast IP addresses. [Figure 11.2](#) displays an example of a multicast IP address.

[Figure 11.2](#) Multicast IP address.

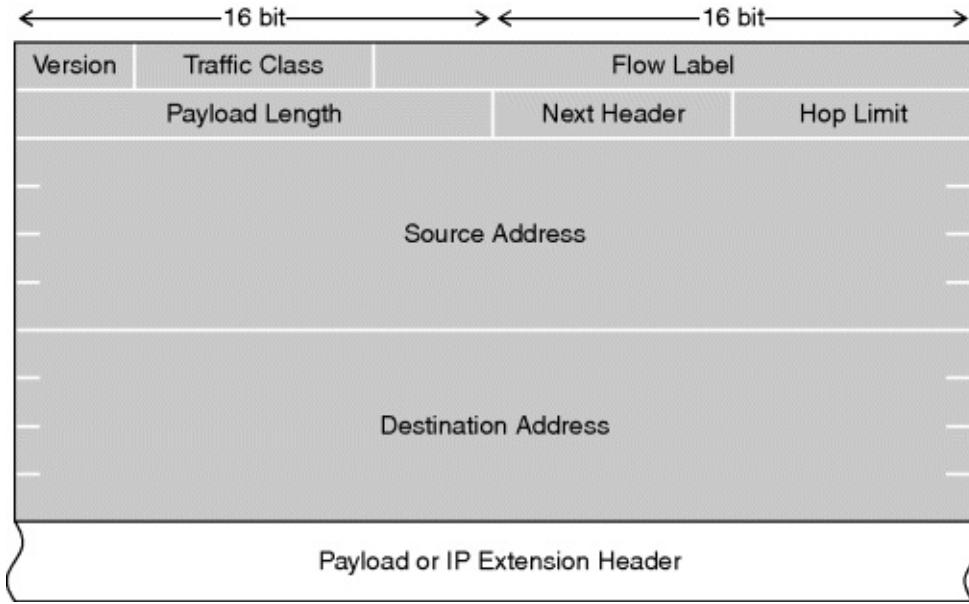


Certain autoconfiguration tasks require a node to send a packet to a neighbor for which it does not know the link-layer address. In such cases, the packet can be sent to a so-called solicited-node multicast IP address. This special multicast IP address is derived from the unicast IP address of interest, and it addresses all nodes on a link that use an IP address with a particular pattern in the last 24 bits. In this case, the packet is sent to a multicast link-layer address, so the sender does not require knowledge of the recipient's actual unicast link-layer address. Given a unicast IP address, the corresponding solicited-node multicast IP address is formed by taking the lower 24 bits of the unicast IP address and prepending to this the prefix FF02:0:0:0:1:FF00::/64.

11.2 IP Packet Headers

For a packet to be forwarded across the Internet, routers must be able to determine the packet's destination node. Moreover, when the destination node eventually receives the packet, it should be able to identify the source node that originated the packet. Packets carry this and other related information in an IP header, a 40-byte data structure depicted in [Figure 11.3](#), that is prepended to the packet's payload generated at the source node's transport layer. Routing is primarily based on the IP destination address in an IP header's Destination Address field, and a Source Address field holds the packet's IP source address.

[Figure 11.3](#) Format of IPv6 header.



The remaining fields in the IP header either define how routers should handle the packet or facilitate packet parsing. By means of the Hop Limit field, the source node of the packet can define how often the packet may be forwarded at most. A router decrements a nonzero value in the Hop Limit field before forwarding the packet or drops the packet altogether if the value is already zero. The source node or a router may further use the Flow Label and Traffic Class fields to classify the packet for special quality-of-service treatment. The value in the Next Header field specifies the type of payload that follows the header, and the Payload Length field indicates the length of that payload. The Version field contains a constant “6,” indicating that the header format conforms to IPv6.

Some protocols, including mobility protocols, require packets to carry IP layer information or directives that go beyond what fits in an IP header. Such information can be transported in the following optional IP extension headers that are inserted between the IP header itself and the payload [10].

- Various auxiliary IP layer information can be included in a Destination Options extension header. This header carries one or more options to be processed by a destination node.
- A packet that is supposed to take a particular path to the destination node may include a Routing extension header to specify a sequence of intermediate IP destination addresses that the packet should traverse. These IP destination addresses may belong to different nodes, but as will be explained later on in this chapter, a mobility

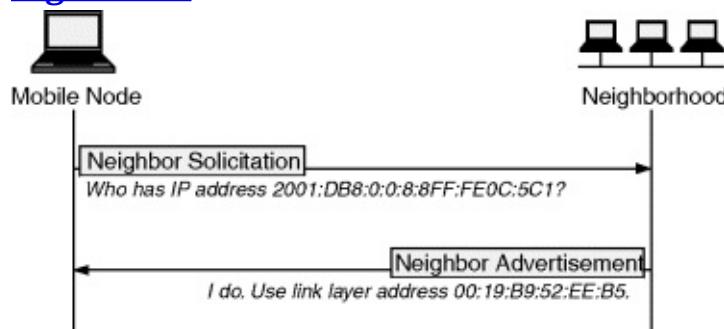
protocol may also use a Routing extension header to forward a packet virtually between different IP addresses of the same node.

- Packets can be authenticated, integrity protected, and encrypted at the IP layer through the IP Security protocol (IPsec) [11]. Depending on the IPsec mode, a protected packet includes either an *Authentication extension header* or an *Encapsulating Security Payload extension header*.
- Fragmented packets include a Fragmentation extension header to aid reassembly at the destination node.
- A packet that requires special handling on each router along its path may carry a Hop-by-Hop Options extension header including one or more options that each router should process.

11.3 IP Address Resolution

IP address resolution is accomplished by the Neighbor Discovery Protocol with the message exchange illustrated in [Figure 11.4](#). A resolving node generates a Neighbor Solicitation message that asks the owner of a specific IP address to return its link-layer address. This message is sent to the solicited-node multicast IP address that corresponds to the IP address of interest. The node that uses the IP address of interest returns its link-layer address in a Neighbor Advertisement message. Resolved IP addresses are stored in a neighbor cache along with the respective link-layer address to avoid repeated resolution of the same IP address.

[Figure 11.4](#) IP address resolution.



11.4 IP Version 6 Deployment: Drivers and

Impediments

The two options to make more devices addressable on the Internet—address sharing with IPv4 address translation and introducing new addresses with IPv6—have contrasting properties regarding ease of deployment and sustainability. IPv4 address translation makes address sharing facile. It is simply a matter of installing an address translator on the border of the network and renumbering the network internally using nonglobal address space. The address translator then maps the nonglobal address space onto one or a small set of global IPv4 addresses. Except for the address translator, all changes are configuration-only; hardware or software upgrades are not required.

Deploying IPv6 takes considerably more effort because it affects the network, the host operating systems, and the applications. Regarding the network, not only routers need to change, but also network-based applications such as mobility support, load balancing, firewalling, and intrusion detection. The challenge here involves multiple stakeholders that are in general independent and not coordinated. The complexity of deploying IPv6 explains the continuing reluctance to deploy IPv6 at a large scale. Of the sets of administratively contiguous networks on the Internet, so-called *autonomous systems*, the fraction that has deployed IPv6 has grown insignificantly from only 2.4% in 2004 to 4.4% in 2009 [7, 8].

On the other hand, deploying IPv6 is significantly more sustainable than IPv4 address sharing. This is due to three important technical drawbacks in address sharing:

- *Limited host reachability*: Address sharing makes it more difficult to contact a device that shares its address because the address is then not unique.
- *Reduced network robustness*: Address sharing creates a new single point of failure in the network because address translators create state that communication sessions depend on.
- *No support for address referrals*: Address translation does not work well with applications that use address referrals because the translation of an address invalidates referrals to the address.

These three issues are inherent in IPv4 address sharing and will be aggravated as IPv4 addresses are shared more and more aggressively. The issues can, on the long term, only be avoided by introducing additional

addresses, that is, through the deployment of IPv6. Nonetheless, as will be shown in the following, address sharing will still be necessary on the short term. It is needed for interworking between early IPv6 adopters and the legacy IPv4 Internet.

11.4.1 Need for Backwards Compatibility

Given the involvement of multiple uncoordinated stakeholders, the transition to IPv6 cannot happen instantaneously. Therefore it is essential to enable individual stakeholders to deploy IPv6 independently of each other. This requires interworking between stakeholders who have already adopted IPv6 and stakeholders who have not yet done so. There are two ways of enabling such interworking:

- *Backwards compatibility for IPv6*: IPv6 adopters use methods that allow them to connect to and become reachable from the legacy IPv4 Internet.
- *Forward compatibility for IPv4*: IPv4 networks deploy methods that allow them to connect to and become reachable from IPv6-enabled networks.

Which of these approaches is feasible depends on whether the impetus for interworking is on the side of early IPv6 adopters or the legacy IPv4 Internet. This impetus will shift over time. Initially, the impetus will naturally be high on the side of early IPv6 adopters. Services and peers will at this point almost exclusively be IPv4-only, so IPv6 adopters will have to ensure they can communicate via IPv4. For the same reason, initial incentives to invest in interworking will be low on the side of the legacy IPv4 Internet. Why would the legacy IPv4 Internet want to communicate via IPv6 if everything they need is also available via IPv4?

As IPv6 deployment progresses, more and more services and peers will become IPv6 capable. The interworking impetus on the side of IPv6 adopters will therefore shrink. At some point, a sufficient amount of deployed IPv6 will make it feasible for services and peers to be IPv6 only. And as there are more and more IPv6-only services and peers, the impetus for interworking will grow on the side of the legacy IPv4 Internet, so that they can reach the IPv6 Internet with its new services and potential peers.

Consequently, the interworking methods that will be needed first are those for deployment by IPv6 adopters, which make IPv6 backwards compatible

with IPv4. Unfortunately, exactly those interworking methods limit IPv6's main benefit, the high number of devices additionally addressable on the Internet. The reason is that backwards compatibility of IPv6 with IPv4 requires every device with an IPv6 address to also be reachable via an IPv4 address. So the number of devices addressable continues to be limited by what is possible with IPv4.

To corroborate why IPv4 reachability is a prerequisite for IPv6 backwards compatibility, let's take a closer look at the three main backwards compatibility methods for IPv6:

- *Dual stack*: Dual stack [12] makes IPv6-adopting networks, host operating systems, and applications capable of supporting both IP versions. The goal is to provide an end-to-end path for at least one of the two IP versions. Accordingly, dual-stack hosts must be given addresses from both IPv6 and IPv4.
- *Translation*: Translation between IPv4 and IPv6 [6] enables a network, including its hosts and their applications, to be IPv6 only. It requires an IPv4–IPv6 address translator on the border of the network, to convert IPv4 packets into IPv6 packets, and vice versa, when necessary. Hosts in an IPv6-only network are directly assigned only an IPv6 address, but they must still be reachable via an IPv4 address that is assigned to the Internet-facing side of the IPv4–IPv6 address translator.
- *Tunneling*: Tunneling [13] enables hosts to communicate via paths that are partly IPv4 only and partly IPv6 only. If the hosts use IPv4, their packets are carried over IPv6-only networks via IPv4-in-IPv6 tunnels. If they use IPv6, their packets are carried over IPv4-only networks via IPv6-in-IPv4 tunnels. Either the hosts or the tunnel endpoints must each be assigned an IPv4 address.

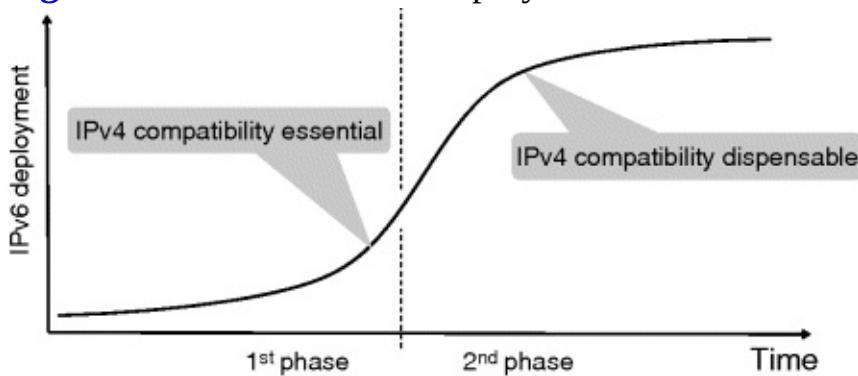
Consequently, either method to make IPv6 backwards compatible with IPv4 limits the benefit of IPv6 deployment in terms of making more devices addressable on the Internet because all methods require IPv6-enabled hosts to be reachable via an IPv4 address. This IPv4 address may be directly assigned to the host, as in dual stack, or it may be assigned to the Internet-facing side of a translator. Therefore the number of devices addressable will continue to be limited by IPv4 as long as IPv6 must be backwards compatible. This, in turn, means that IPv4 address sharing will continue to be necessary in addition to IPv6.

11.4.2 Initial Deployment Drivers

The initial need for backwards compatibility with IPv4 calls for two stages of IPv6 deployment ([Figure 11.5](#)):

- *Backwards compatibility essential during first stage:* During an initial IPv6 deployment stage, backwards compatibility with IPv4 will be essential. Each host will have to be reachable via an IPv4 address, and IPv4 address sharing will be necessary to make this happen. Unfortunately, this also means that deploying IPv6 will initially have little benefit in terms of increasing the number of devices addressable on the Internet.
- *Backwards compatibility dispensable during second stage:* During a later stage of IPv6 deployment, however, backwards compatibility with IPv4 will no longer be needed. The amount of deployed IPv6 will be sufficient, so devices that have an IPv6 address will no longer have to be reachable at an IPv4 address at the same time. Then, IPv6 will unfold its true potential in terms of making more devices addressable. These advantages are going to accelerate IPv6 deployment in the second phase. The second IPv6 deployment phase will finally slow down for saturation reasons.

[Figure 11.5](#) Phases of IPv6 deployment.



The observation that IPv6 deployment will happen in two phases, and that the large address space of IPv6 will be a real driver only for the second IPv6 deployment phase, raises an important question about the incremental deployability of IPv6: What will drive the first phase of IPv6 deployment? If reachability via IPv4, and hence IPv4 address sharing, will be required anyway, then why would anyone adopt IPv6 early on?

Fortunately, IPv6 has benefits in addition to a large address space. These

additional benefits are administrative in nature. In general, they are secondary to the large address space, but they will be immediately noticeable by early IPv6 adopters. The main administrative advantages of IPv6 are as follows:

- *Simplified network numbering*: IPv6 makes it easier to renumber large networks internally. The larger address space of IPv6 makes it possible to introduce generous grace periods, during which a new address space coexists with the old address space.
- *Autonomous host configuration*: IPv6 does not require infrastructure for address assignment, router and neighbor discovery, and configuration of various other network access parameters. IPv4 requires infrastructure for these functions.
- *Low address space fragmentation*: In IPv4, the address shortage has led to the policy of allocating address blocks that are not much larger than the actual need. As a consequence, an allocated IPv4 address block typically does not allow for much network growth, and a network that is growing may require additional address blocks. These address blocks are in general not contiguous and must be maintained separately. This implies an administrative cost. This cost can often be avoided in IPv6; the large IPv6 address space enables more generous address allocation, which reduces the frequency by which new address blocks must be obtained.
- *Less infrastructure for mobility support*: Mobility support in IPv6 requires less infrastructure than in IPv4, and is hence less expensive to provide. In IPv4, every access router must be upgraded with mobility support [14]. This is not needed for mobility support in IPv6 [15].
- *More by-default functionality*: IPv6 devices come with more functionality by default, and are therefore more versatile to use. One feature that is readily available in IPv6 devices, but not necessarily in IPv4 devices, is IP security. Another such feature is multicast support [16].

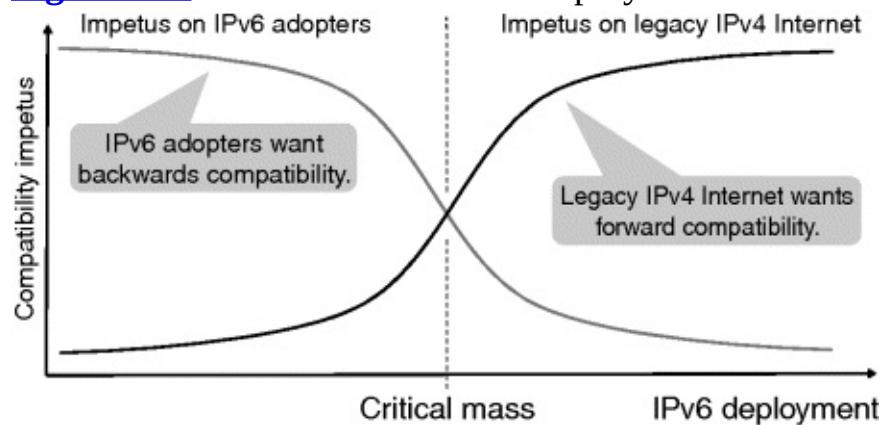
These administrative benefits will drive the first stage of IPv6 deployment. Whether the second stage of IPv6 deployment will be reached depends on whether the administrative benefits prove to be sufficient to yield the amount of IPv6 deployment that is necessary to facilitate IPv6-only deployments. Only then will IPv6's primary benefit, the large new address space, take effect.

11.4.3 Reaching a Critical Mass

Whereas the complexity of deploying IPv6 compared to sharing IPv4 addresses explains the past widespread reluctance to deploy IPv6 at a large scale, the lack of sustainability of IPv4 address sharing, due to fundamental technical drawbacks, implies that deployment of IPv6 is the only viable solution in the long term. Nonetheless, IPv4 address sharing will be necessary during the initial stages of IPv6 deployment due to the need for backwards compatibility with IPv4. Each device with an IPv6 address will initially also have to be reachable at an IPv4 address, and this can happen only with aggressive sharing of IPv4 addresses.

Therefore, the extra cost for deploying IPv6 compared with IPv4 address sharing will not pay off right away with an ability to address more devices on the Internet. Initially, the number of devices addressable on the Internet will still be limited by what is possible today with IPv4 and IPv4 address sharing. Early deployment of IPv6 will be largely driven by administrative benefits. The large new address space of IPv6 will become a deployment driver at a later stage, when the amount of deployed IPv6 has reached a critical mass, rendering backwards compatibility with IPv4 dispensable. [Figure 11.6](#) illustrates the point when IPv6 deployment reaches its critical mass.

Figure 11.6 Critical mass of IPv6 deployment.



What amount of IPv6 deployment is necessary to make backwards compatibility with IPv4 dispensable? This question is hard, if not impossible, to predict because it depends on assessments of individual network operators and service providers. First, it depends on the assessed value of the initial, administrative benefits of IPv6. This will determine the speed of early IPv6

deployment. Second, it depends on the assessment of how much IPv6 deployment there should be to obviate need for IPv4 backwards compatibility.

References

- 1.** Miniwatts Marketing Group, Internet World Stats, “World Internet Users and Population Statistics,” <http://www.internetworldstats.com/stats.htm>, Oct. 2010.
- 2.** Kapko, M., “5.2B Mobile Subscribers By 2011, 1.7B On Mobile Web And 2.1B Mobile Purchases By 2013,” <http://www.moconews.net/>, Aug. 2008.
- 3.** International Telecommunication Union, “Measuring the Information Society –The ICT Development Index,” 2009.
- 4.** International Telecommunication Union, “186 Million Machines Will Be Connected to Mobile Networks in 2012,” <http://www.itu.int/ITU-D/ict/newslog/186+Million+Machines+Will+Be+Connected+To+Mobile+Networks+In+2012.aspx>, May 2008.
- 5.** Huston, G., “IPv4 Address Report,” <http://www.potaroo.net/tools/ipv4>, Jan. 2011.
- 6.** Srisuresh, P. and Egevang, K.B., “Traditional IP Network Address Translator (Traditional NAT),” IETF RFC 3022, Jan. 2001.
- 7.** Huston, G., “IPv6 Deployment –Just Where Are We?” http://www.circleid.com/posts/ipv6_deployment_where_are_we/, Mar. 2008.
- 8.** BGP Mon, “Global IPv6 Deployment Statistics,” <http://bgpmon.net/blog/?p=166>, Apr. 2009.
- 9.** Arbor Networks, “Tracking the IPv6 Migration,” Aug. 2008.
- 10.** Deering, S.E. and Hinden, R.M., “Internet Protocol, Version 6 (IPv6) – Specification,” IETF RFC 2460, Dec. 1998.
- 11.** Kent, S. and Seo, K., “Security Architecture for the Internet Protocol,” IETF RFC 4301, Dec. 2005.
- 12.** Stewart, L. and Healy, J., “Tuning and Testing the FreeBSD 6 TCP Stack,” Technical Report 070717B, Centre for Advanced Internet Architectures, Faculty of Information and Communication Technologies,

Swinburne University, Melbourne, Australia, Jul. 2007.

13. Montenegro, G. (Editor), “Reverse Tunneling for Mobile IP, Revised,” IETF RFC 3024, Jan. 2001.

14. Perkins, C. (Editor), “IP Mobility Support for IPv4,” IETF RFC 3344, Aug. 2002.

15. Johnson, D., Perkins, C.E. and Arkko, J., “Mobility Support in IPv6,” IETF RFC 3775, June 2004.

16. Loughney, J. (Editor), “IPv6 Node Requirements,” IETF RFC 4294, Apr. 2006.

Chapter 12

IP Traffic Engineering

Implementing the most appropriate set of routes can increase the network resource utilization rate and network throughput of Internet Protocol (IP) networks. Since it optimizes the assignment of traffic resources, additional traffic can be supported. It also suppresses network congestion and increases robustness in the face of traffic demand fluctuations, most of which are difficult to predict. One useful approach to enhancing routing performance is to minimize the maximum link utilization rate, also called the network congestion ratio, of all network links. Minimizing the network congestion ratio leads to an increase in admissible traffic. This chapter describes several routing schemes to maximize network utilization for various traffic-demand models, where the Multiprotocol Label Switching (MPLS) and Open Shortest Path First (OSPF) technologies are considered.

12.1 Models of Traffic Demands

When the routes between source and destination nodes are designed to utilize network resources, traffic demands for each source and destination pair need to be predicted. This section introduces three traffic-demand models: a pipe model, a hose model, and an intermediate model.

The traffic demand d_{pq} between source node p and destination node q is denoted as d_{pq} . The traffic matrix is expressed by $T = \{d_{pq}\}$, whose dimension is $N \times N$, where N is the number of nodes in the network. If each element of the traffic matrix d_{pq} is specified, the traffic model is called the *pipe model*. In other words, the traffic matrix is known [1, 2, 3]. However, it is difficult for network operators to measure and predict the actual traffic matrix [4, 5, 6, 7].

There are several studies on estimating the traffic matrix [8, 9, 10, 11]. To estimate the traffic matrix, network operators do not need to measure each traffic demand d_{pq} . Instead, they only have to measure the traffic loads on

each link in the network. Using the measured link loads, the traffic matrix can be estimated with a certain level of accuracy, but some errors are inevitable. In addition, traffic demands often fluctuate. Therefore, in the pipe model, d_{pq} must include some margin to absorb estimation errors of the traffic matrix and the traffic fluctuations. This lowers the network resource utilization rate.

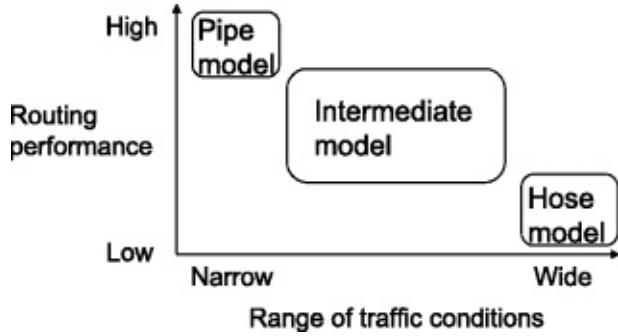
It is, however, easy for network operators to specify the traffic as just the total outgoing/incoming traffic from/to node p and node q . The total outgoing traffic from node p is represented as $\sum_q d_{pq} \leq \alpha_p$, where α_p is the maximum rate of traffic that node p can send into the network. The total incoming traffic to node q is represented as $\sum_p d_{pq} \leq \beta_q$, where β_q is the maximum rate of traffic that node q can receive from the network. The traffic model that is bounded by α_p and β_q is called the *hose model* [1, 2, 3]. In the hose model, the traffic demand between each source-destination pair does not need to be specified. Therefore, it is beneficial for network operators to specify a set of traffic conditions in the hose model, especially when the network is large. The hose model handles well highly variable traffic conditions. Routing that is robust to changing and uncertain traffic demands is called *oblivious routing* [7, 12, 13, 14, 15].

The hose model has a weakness in that its routing performance is much lower than that of the pipe model. This is because all possible sets of $T = \{d_{pq}\}$ must be considered in the hose model, while the exact $T = \{d_{pq}\}$ is known in the pipe model. Therefore, it is desirable for network operators to narrow the range of traffic conditions specified by the hose model so as to enhance its routing performance.

Network operators are able to impose additional bounds on the hose model from their operational experience and past traffic data as described below. First, the range of errors of the traffic matrix can be estimated. It is reported that traffic estimations based on link traffic measurements are in error by 20% or more [8]. Second, traffic fluctuations from the network operational history can be predicted. Third, the network operators know the trends of traffic demands. For example, traffic demands among large cities are much larger than those of small cities. Thus, the additional bounds are expressed by $\delta_{pq} \leq d_{pq} \leq \gamma_{pq}$ for each pair of source node p and destination q . γ_{pq} and δ_{pq} are the upper and lower bounds of d_{pq} , respectively. This model, the hose

model with additional bounds, is called the *intermediate model* [16, 17]. When the range of d_{pq} is not exactly determined, a large range of d_{pq} is set (i.e., a conservative approach) to avoid any congestion. This may still be effective because α_p and β_q also bound the traffic demands. The intermediate model offers better routing performance than the hose model, by narrowing the range of traffic conditions specified by the hose model. The features of the pipe, hose, and intermediate models are schematically shown in [Figure 12.1](#).

Figure 12.1 Features of pipe, hose, and intermediate models.



12.2 Optimal Routing with Multiprotocol Label Switching

12.2.1 Overview

Wang and colleagues [18] present a general routing problem as a linear programming (LP) formulation, where traffic demands are assumed to be explicitly routed and flexibly split among source and destination nodes. This operation is performed by the MPLS Traffic Engineering (TE) technology [19], [20]. They assume the exact traffic demand d_{pq} between source node p and destination node q is known. This is the pipe model. Chu and colleagues formulate the general routing problem as an LP problem in the hose model [4, 5]. Oki and colleagues formulate the routing problem as an LP problem in the intermediate model [16, 17] and clarify how much the optimal-routing performances offered by the pipe and hose models differ, and how much the intermediate model improves the performance compared with the hose model.

12.2.2 Applicability of Optimal Routing

12.2.2.1 IP Backbone Network

Optimal routing based on a traffic model is applied to IP backbone networks. They provide high-bandwidth connectivity across wide geographical areas. An IP backbone network consists of a set of high-performance nodes and high-speed links. Nodes, which are known as points of presence (POPs), connect regional and metro networks to the backbone networks. Nodes in the IP backbone network are considered to provide nonblocking features.

To avoid network congestion, the maximum link utilization rate in the network, as determined by the pipe, hose, or intermediate model, must be provisioned to be minimized. The traffic conditions are determined by measurements and/or traffic demands estimated by operational experience and past traffic data.

12.2.2.2 Virtual Private Network Services

Network operators provide customers with virtual private network (VPN) services. Optimal routing is also essential in VPN services. It is common to enter into a service-level agreement (SLA) with VPN customers that specifies a VPN service model.

Conventional VPN service models are the pipe and hose models. For VPN customers, the hose model is more flexible than the pipe model. In the hose model, they only have to specify the traffic as the total outgoing/incoming traffic from/to provider edge nodes, while in the pipe model they need to specify their traffic matrix. If the customers can specify the traffic matrix, the pipe model is more cost effective because it provides better routing performance than the hose model.

The intermediate model can be added as one of the VPN service models. When a VPN service is requested by a customer, the network operator performs *service admission control* to determine whether the service request can be accepted by computing optimal routing. A customer declares its traffic conditions as specified by the pipe, hose, or intermediate model. The network operator computes optimal routing based on the available bandwidths in the network. The rerouting of existing traffic demands may be considered. From the customers' point of view, the intermediate model is more (less) flexible than the pipe model (hose model). However, a network operator can provide

a VPN service more cost-effectively with the intermediate model than with the hose model because the intermediate model offers superior routing performance.

12.2.3 Network Model

The terminologies defined here are used in this chapter unless otherwise stated. The network is represented as directed graph $G(V, E)$, where V is the set of vertexes (nodes) and E is the set of links. Let $Q \subseteq V$ be the set of edge nodes through which traffic is admitted into the network. A link from node $i \in V$ to node $j \in V$ is denoted as $(i, j) \in E$. c_{ij} is the capacity of $(i, j) \in E$. The traffic demands of $T = \{d_{pq}\}$ are specified by the intermediate model. x_{ij}^{pq} , where $0 \leq x_{ij}^{pq} \leq 1$ is the portion of the traffic from node $p \in Q$ to node $q \in Q$ routed through $(i, j) \in E$. $(i, j) \in E$. $x_{ij}^{pq} > 0$ means (i, j) is a link on one of the routes for d_{pq} . $d_{pq} \cdot x_{ij}^{pq} = 0$ means that (i, j) is not a link on any route for d_{pq} . In the case of $0 < x_{ij}^{pq} < 1$, d_{pq} is split over multiple routes.

The network congestion ratio, which refers to the maximum value of all link utilization rates in the network, is denoted as r . Minimizing r means that admissible traffic is maximized. The admissible traffic volume is accepted up to the current traffic volume multiplied by $1/r$.

12.2.4 Optimal Routing Formulations with Three Models

12.2.4.1 Pipe Model

An optimal routing formulation with the pipe model to minimize the network congestion ratio is presented in reference 18. The network congestion ratio is obtained by solving an LP problem as follows.

$$\begin{aligned} & \min r \\ \text{(12.1a)} \quad & \text{s.t. } \sum_{j:(i,j) \in E} x_{ij}^{pq} - \sum_{j:(j,i) \in E} x_{ji}^{pq} = 0 \end{aligned}$$

$$p, q \in Q, i \neq p, i \neq q$$

$$(12.1b) \quad \sum_{j:(i,j) \in E} x_{ij}^{pq} - \sum_{j:(j,i) \in E} x_{ji}^{pq} = 1$$

$$(12.1c) \quad p, q \in Q, i = p$$

$$(12.1d) \quad \sum_{p,q \in Q} d_{pq} x_{ij}^{pq} \leq c_{ij} \cdot r \quad (i, j) \in E$$

$$(12.1e) \quad 0 \leq x_{ij}^{pq} \leq 1 \quad p, q \in Q, (i, j) \in E$$

$$(12.1f) \quad 0 \leq r \leq 1$$

The decision variables are r and x_{ij}^{pq} , and the given parameters are d_{pq} and c_{ij} . The objective function in [Equation 12.1a](#) minimizes the network congestion ratio. [Equations 12.1b](#) and [12.1c](#) are constraints for flow conservation. [Equation 12.1b](#) states that the traffic flow incoming to node i must be the traffic outgoing from node i if node i is neither a source or destination node for the flow. [Equation 12.1c](#) states that the total traffic flow ratio outgoing from node $i (= p)$, which is a source node, is 1. As flow conservation is satisfied at a destination if [Equations 12.1b](#) and [12.1c](#) are satisfied, the condition does not need to be included as a constraint. [Equation 12.1d](#) indicates that the sum of the fractions of traffic demands transmitted over (i, j) is equal to or less than the network congestion ratio times the total capacity c_{ij} for all links.

12.2.4.2 Hose Model

The optimal routing problem with the hose model is formulated by [Equations 12.1a](#) through [12.1f](#). The decision variables are r and x_{ij}^{pq} , and the given parameters are d_{pq} and c_{ij} . However, d_{pq} is bounded by the hose model as follows.

$$(12.2a) \quad \sum_{q \in Q} d_{pq} \leq \alpha_p \quad p \in Q$$

$$(12.2b) \quad \sum_{p \in Q} d_{pq} \leq \beta_q \quad q \in Q$$

Although [Equations 12.1a](#) through [12.1f](#) can be expressed as an LP problem, its solution is not easy to obtain. Constraint [12.1d](#) lists every valid combination in $T = \{d_{pq}\}$ constrained by [Equations 12.2a](#) and [12.2b](#). Chu and colleagues converted the hose-model formulation into a regular LP problem that can be easily solved [4] by using the following dual theorem.

$$\min r$$

$$(12.3a) \quad s.t. \quad \sum_{j:(i,j) \in E} x_{ij}^{pq} - \sum_{j:(j,i) \in E} x_{ji}^{pq} = 0 \\ p, q \in Q, i \neq p, i \neq q$$

$$(12.3b) \quad \sum_{j:(i,j) \in E} x_{ij}^{pq} - \sum_{j:(j,i) \in E} x_{ji}^{pq} = 1 \\ p, q \in Q, i = p$$

$$(12.3c) \quad \sum_{p \in Q} \alpha_p \pi_{ij}(p) + \sum_{p \in Q} \beta_p \lambda_{ij}(p) \leq c_{ij} \cdot r$$

$$(12.3d) \quad (i, j) \in E$$

$$x_{ij}^{pq} \leq \pi_{ij}(p) + \lambda_{ij}(q) \quad p, q \in Q, (i, j) \in E$$

$$(12.3e) \quad \pi_{ij}(p), \lambda_{ij}(p) \geq 0$$

$$(12.3f) \quad p, q \in Q, (i, j) \in E$$

$$(12.3g) \quad 0 \leq x_{ij}^{pq} \leq 1 \quad p, q \in Q, (i, j) \in E$$

$$(12.3h) \quad 0 \leq r \leq 1$$

The decision variables are r , x_{ij}^{pq} , $\pi_{ij}(p)$, $\lambda_{ij}(p)$, and the given parameters are c_{ij} , α_p , β_q . [Equation 12.1d](#) and [Equations 12.2a](#) and [12.2b](#) are replaced by [Equations 12.3d](#) through [12.3f](#). By introducing the variables of $\pi_{ij}(p)$, $\lambda_{ij}(p)$, [Equations 12.2a](#) and [12.2b](#) are incorporated in this optimization problem. [Equations 12.3a](#) through [12.3h](#) represent a regular LP problem and can be

solved optimally with a standard LP solver.

12.2.4.3 Intermediate Model

The optimal routing problem with the intermediate model is formulated by [Equations 12.1a](#) through [12.1f](#). The decision variables are r and x_{ij}^{pq} , and the given parameters are d_{pq} and c_{ij} . However, d_{pq} is bounded by the intermediate model as follows.

$$(12.4a) \quad \sum_{q \in Q} d_{pq} \leq \alpha_p \quad p \in Q$$

$$(12.4b) \quad \sum_{p \in Q} d_{pq} \leq \beta_q \quad q \in Q$$

$$(12.4c) \quad \delta_{pq} \leq d_{pq} \leq \gamma_{pq} \quad p, q \in Q$$

As is the case of the hose model, although [Equations 12.1a](#) through [12.1f](#) can be expressed as an LP problem, it cannot be easily solved as a regular LP problem. Constraint [12.1d](#) lists every valid combination in $T = \{d_{pq}\}$ bounded by [Equations 12.4a](#) through [12.4c](#). It is impossible to repeatedly solve the LP problems for all possible sets of $T = \{d_{pq}\}$. This is because d_{pq} takes a real value and the number of valid combinations of d_{pq} is infinite. This problem is solved by extending Chu's property [5] in the hose model to the intermediate model. Oki and colleagues converted the intermediate-model formulation into a regular LP problem that can be easily solved in the following [17].

$$(12.5a) \quad \begin{aligned} & \min r \\ & \text{s.t. } \sum_{j:(i,j) \in E} x_{ij}^{pq} - \sum_{j:(j,i) \in E} x_{ji}^{pq} = 0 \\ & \quad p, q \in Q, i \neq p, i \neq q \end{aligned}$$

$$(12.5b) \quad \sum_{j:(i,j) \in E} x_{ij}^{pq} - \sum_{j:(j,i) \in E} x_{ji}^{pq} = 1$$

$$(12.5c) \quad \begin{aligned} & \sum_{p \in Q} \alpha_p \pi_{ij}(p) + \sum_{p \in Q} \beta_p \lambda_{ij}(p) \\ & \quad + \sum_{p,q \in Q} [\gamma_{pq} \eta_{ij}(p, q) - \delta_{pq} \theta_{ij}(p, q)] \leq c_{ij} \cdot r \end{aligned}$$

$$(i, j) \in E$$

$$\text{(12.5d)} \quad x_{ij}^{pq} \leq \pi_{ij}(p) + \lambda_{ij}(q) + \eta_{ij}(p, q) - \theta_{ij}(p, q) \\ p, q \in Q, (i, j) \in E$$

$$\text{(12.5e)} \quad \pi_{ij}(p), \lambda_{ij}(p), \eta_{ij}(p, q), \theta_{ij}(p, q) \geq 0$$

$$\text{(12.5f)} \quad p, q \in Q, (i, j) \in E$$

$$\text{(12.5g)} \quad 0 \leq x_{ij}^{pq} \leq 1 \quad p, q \in Q, (i, j) \in E$$

$$\text{(12.5h)} \quad 0 \leq r \leq 1$$

The decision variables are r , x_{ij}^{pq} , $\pi_{ij}(p)$, $\lambda_{ij}(p)$, $\eta_{ij}(p, q)$, and $\theta_{ij}(p, q)$, and the given parameters are c_{ij} , α_p , β_q , δ_{pq} , and γ_{pq} . [Equation 12.1d](#) and [Equations 12.4a](#) through [12.4c](#) are replaced by [Equations 12.5d](#) and [12.5f](#). By introducing the variables of $\pi_{ij}(p)$, $\lambda_{ij}(p)$, $\eta_{ij}(p, q)$, and $\theta_{ij}(p, q)$, [Equations 12.4a](#) through [12.4c](#) are incorporated in this optimization problem. [Equations 12.5a](#) through [12.5h](#) represent a regular LP problem and can be solved optimally with a standard LP solver.

12.3 Link-Weight Optimization with Open Shortest Path First

12.3.1 Overview

Open Shortest Path First [21] is widely used as a link-state-based Interior Gateway Protocol (IGP) for IP networks. In OSPF, all packets are transmitted over the shortest paths as determined by weights associated with each link in the network. As link states, including weights, are distributed from each originating node to all nodes in the same network, each node is able to compute routes to destination nodes in the network. Determining the optimal routing based on shortest-path routing means determining the optimal link weights.

A simple weight setting policy is to make a link weight inversely proportional to its capacity [22]. This policy, which is implemented in commercial routers, makes it easy for network providers to configure routers to avoid network congestion. Traffic tends to avoid links with small capacity and instead use those with large capacity. However, as this policy does not

take traffic demand or network topology into consideration, it fails to achieve optimal routing performance.

Optimization algorithms that compute a set of optimal link weights in OSPF-based networks were addressed in [23, 24, 25, 26, 27] under the condition that the network topology and traffic matrix are given. The general objective is to distribute traffic flows evenly across available network resources in order to avoid network congestion and quality of service degradation. A straight-forward approach is to compute the network congestion ratios for every possible set of link weights to find the optimal set that minimizes the network congestion ratio. However, the computation time complexity of this approach is $O(x^L)$, where x is the higher limit of weights and L is the number of links in the network. To reduce the computation time complexity, Fortz and colleagues presented a heuristic algorithm based on tabu search [23, 24]. Buriol and colleagues presented a genetic algorithm with a local improvement procedure [25]. A fast heuristic algorithm was also developed by Reichert and Magedanz [26]. These optimization algorithms yield nearly optimal sets of link weights in a practical manner.

Chu and colleagues presented an optimization scheme for a set of link weights with the hose model [4, 5].

12.3.2 Examples of Routing Control with Link Weights

Consider a network model, as shown in [Figure 12.2](#). Assume that all the link capacities in the network are 10. Let w_{ij} be the link weight of $(i, j) \in E$. Consider the traffic demands in Case 1 in [Table 12.1](#). To minimize the network congestion ratio, which is the maximum link utilization in the network, the optimum link weights and link utilizations for each link are determined in [Table 12.2](#). Traffic from node a to node d is routed on $a - b - c - d$, while traffic from node e to node d is routed on $e - d$. As a result, the network congestion ratio is 0.5.

[Figure 12.2](#) Network model.

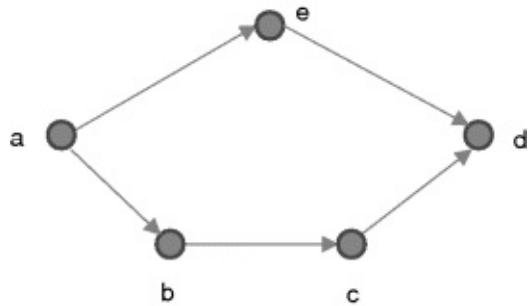


Table 12.1 Traffic Demand (Case 1)

Source	Destination	Traffic Demand
a	d	4
e	d	5

Table 12.2 Optimum Link Weights (Case 1)

Link	Weight	Utilization
(a, b)	1	0.4
(a, e)	2	0.0
(b, c)	1	0.4
(c, d)	1	0.4
(e, d)	2	0.5

Next, consider that traffic demands change from Case 1 to Case 2. In Case 2, the traffic demand from node c to node d , whose volume is 5, is added (Table 12.3). If the link weights determined in Table 12.1 are used for Case 2, the network congestion ratio becomes 0.9, which is the link utilization of (c, d) . These link weights are not optimum for Case 2. In OSPF, if there are multiple routes that have the same cost, each of which is a sum of link weights on the route, traffic can be *equally* distributed on the multiple routes. It is called equal-cost multipath (ECMP) routing. To optimize the link weights for Case 2, ECMP should be considered. The optimum link weights are shown in Table 12.4. The traffic with 4 from node a to node d is split into two routes of $a - b - c - d$ and $a - e - d$, whose path costs are equal. As a result, traffic loads for each link are balanced over the network by updating the link weights, and the network congestion ratio becomes 0.7.

Table 12.3 Traffic Demand (Case 2)

Source	Destination	Traffic Demand
a	d	4
c	d	5
e	d	5

Table 12.4 Optimum Link Weights (Case 2)

Link	Weight	Utilization
(a, b)	1	0.2
(a, e)	2	0.2
(b, c)	1	0.2
(c, d)	2	0.7
(e, d)	2	0.7

12.3.3 Link-Weight Setting Against Network Failure

When the network topology and traffic matrix are given, start-time optimization (SO) can determine the optimal set of link weights once at the beginning of network operation. Unfortunately, SO is weak against network failure, for example, a link failure. As link failure will trigger the rerouting of active paths, the SO-generated weight set is no longer optimal. This causes unexpected network congestion.

The weakness of SO can be overcome by computing a new optimal set of link weights whenever the topology changes. This approach, called run-time optimization (RO), provides the best routing performance after each link failure, but it makes the network unstable. When link weights are changed, the updated information is broadcast across the network. As routers learn the updated link weights, they recompute their shortest paths to update their routing tables. Meanwhile, IP packets may arrive out of order and the performances of Transport Control Protocol (TCP) connections are degraded [23, 24]. Obviously, the more often link weights are changed, the more the network becomes chaotic. This is because packets are sent back and forth between routers to achieve the very divergent processes of updating the routing table and calculating the shortest paths based on the new link weights. If the network is large, the degree of network instability is more pronounced when link failure occurs. Therefore, it seems reasonable to target the one-time calculation of link weights that can handle any link failure.

Oki and colleagues presented preventive start-time optimization (PSO), a scheme that determines, at the start time, a suitable set of link weights that can handle any link failure scenario preventively. This paper describes detail analysis of PSO by extending our previous work in [28]. The set of link weights determined by PSO minimizes the worst-case network congestion

ratio for all possible link failure scenarios.

The network is represented as directed graph $G(V, E)$, where V is the set of nodes and E is the set of links. $v \in V$, where $v = 1, 2, \dots, N$, indicates an individual node and e , where $e = 1, 2, \dots, L$, indicates a bidirectional individual link. N is the number of nodes and L is the number of links in the network. We consider only single link failure in this work, as the probability of multiple link failure at the same time is much less than that of single link failure. F is the set of link failure indices l , where $l = 0, 1, 2, \dots, L$ and $F = E \cup \{0\}$. The number of elements in F is $|F| = L + 1$. $l = 0$ indicates no link failure and $l(\neq 0)$ indicates the failure of link $e = l \in E$. G_l denotes G that has no link $e = l(\neq 0)$ because of link failure, where $G_0 = G$ as $l = 0$ indicates no failure.

$W = \{w_e\}$ is the $L \times 1$ link weight matrix of network G , where w_e is the weight of link e . $r(W, l)$ is a function that returns the congestion ratio for G_l according to OSPF-based shortest-path routing using the link weights in W . $R(W)$ refers to the worst-case congestion ratio in W among all link failure scenarios $l \in F$. $R(W)$ is defined by

$$(12.6) \quad R(W) = \max_{l \in F} r(W, l).$$

$W_l^* = \{w_1, w_2, \dots, w_{l-1}, w_{l+1}, \dots, w_L\}$ is an $(L - 1) \times 1$ link weight matrix that is optimized for network G_l . $(L - 1) \times 1$ is an $L \times 1$ link weight matrix, where w_l^k is obtained by the optimization for G_k ($k \neq l \in F$) and the other link weights are set by using W_l^* . W_l^k is represented by,

$$(12.7) \quad W_l^k = W_l^* \cup \{w_l^k\}.$$

W_l , which is an $L \times 1$ link weight matrix, refers to W_l^k defined in [Equation 12.7](#) that minimizes $R(W_l^k)$ defined in [Equation 12.6](#) over $k \in F$. W_l is defined by

$$(12.8) \quad W_l = \arg \min_{k(\neq l) \in F} R(W_l^k).$$

Our target is to find the most appropriate set of link weights, W_{PSO} , for network G that minimizes $R(W_l)$ defined in [Equation 12.6](#) over link failure index $l \in F$. W_{PSO} is defined by

$$(12.9) \quad W_{PSO} = \arg \min_{l \in F} R(W_l).$$

The network congestion ratio achieved by using W_{PSO} is $R(W_{PSO})$; it

represents the upper bound of congestion for any link failure scenario in the network.

12.4 Extended Shortest-Path-Based Routing Schemes

Legacy networks mainly employ shortest-path-based routing protocols such as OSPF and Intermediate System to Intermediate System (IS-IS). This means that already deployed IP routers in the legacy networks need to be upgraded, which significantly increases capital expenditures. Another problem with MPLS-TE is its scalability in terms of network size. Network operators need to configure and manage LSP tunnels between all edge nodes to form a mesh-like logical topology. The number of tunnels increases in proportion to N^2 . Therefore, for network operators that have not adopted MPLS in their network, it is desirable that an existing IP routing protocol still in use should be utilized.

In OSPF, all packets are transmitted over shortest paths based on link weights associated with each link in the network. As link states, including weights, are distributed from each originating node to all nodes in the same network, each node is able to compute routes to destination nodes in the network. When traffic demands change optimum link weights are recalculated and network operators configure the updated link weights. According to the updated weights, IP routes are changed. Changing routes frequently may cause network instability, which leads to packet loss and the formation of loops.

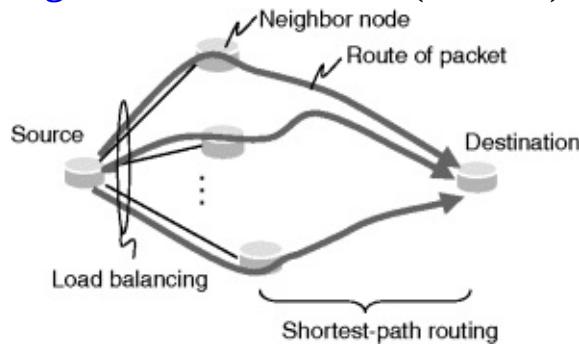
This section introduce three shortest-path-based routing schemes that extends the routing based on OSPF.

12.4.1 Smart–Open Shortest Path First

Mishra and colleagues introduced Smart-OSPF (S-OSPF) [MishraGlobecom07]29. In S-OSPF, source nodes distribute traffic to the neighbor nodes, as shown in [Figure 12.3](#), with optimum ratios that are obtained by solving an LP problem. After the traffic reaches the neighbor nodes, it is routed according the OSPF protocol. As a result, traffic is more load-balanced over the network compared with regular OSPF. Oki and

colleagues formulated and solved the problem to obtain the optimum ratios for traffic distribution for each source node with the hose model [30], while Mishra and colleagues did this with the pipe model [29].

Figure 12.3 Smart-OSPF (S-OSPF).



To implement S-OSPF, the IP forwarding table at the source node must be changed, unlike regular OSPF. The IP forwarding table that is created by regular OSPF is modified so that traffic can be distributed to the neighbor nodes with the computed distribution ratios. [Figure 12.4](#) shows examples of IP forwarding tables for classical OSPF and S-OSPF. S-OSPF has two types of IP forwarding tables. One is used for traffic entering the network at the node, where traffic is distributed to the neighbor nodes. The other, which is the same as that of classical OSPF, is used to handle transit traffic at the node.

Figure 12.4 Examples of IP forwarding tables with OSPF and S-OSPF.

Prefix	Next hop
130.60. 225.0/24	B
123.45.0.0/16	A
123.45.65.0/24	C
:	:

(a) Classical OSPF

Prefix	Next hop	Distribution ratio
130.60. 225.0/24	A	0.4
	B	0.5
	C	0.1
123.45.0.0/16	B	0.7
	C	0.3
123.45.65.0/24	A	0.2
	B	0.3
	C	0.5
:	:	:

For traffic entering network

Prefix	Next hop
130.60. 225.0/24	B
123.45.0.0/16	A
123.45.65.0/24	C
:	:

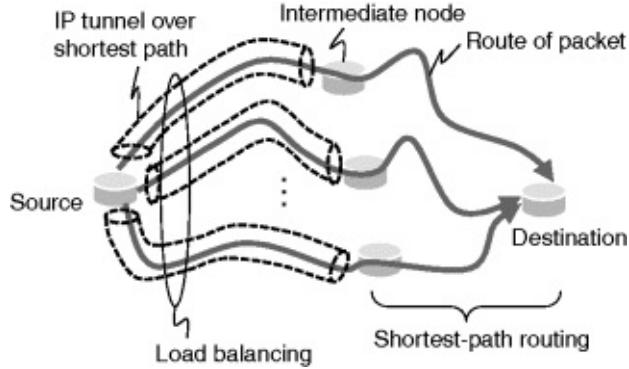
For transit traffic

(b) S-OSPF

12.4.2 Two-Phase Routing

Two-phase routing (TPR) over shortest paths, presented by Antic and colleagues [31, 32]. In [31], all the network nodes are assumed to generate equal loads. TPR was developed for the hose model. TPR performs load balancing and each flow is routed according to the shortest-path-based OSPF protocol, in two stages across intermediate nodes, as shown in [Figure 12.5](#). It is shown that TPR effectively relaxes network congestion in [31, 32].

[Figure 12.5](#) Two-phase routing over shortest paths.



In TPR, packets at the source node are not directly routed to the destination node over the shortest path. TPR consists of two phases as follows. In the first phase, packets are directed to intermediate nodes and pass through IP tunnels, such as IP-in-IP and Generic Routing Encapsulation (GRE) tunnels. The tunnels are made by encapsulating the original IP packets with additional IP headers that indicate the intermediate node. An encapsulated IP packet in an IP tunnel is also routed over the shortest path (to the intermediate node). In the second phase, an encapsulated IP packet is de-encapsulated to yield the original IP packet at the intermediate node, and then sent to the destination node over the shortest path. For a large network, the number of possible routes is high because there are many intermediate nodes that can be used as the target of the distributed IP tunnels. The protocol requires the configuration of IP tunnels between all edge nodes and intermediate nodes in the network. As is true in the case of MPLS-TE, the number of tunnels increases in proportion to N^2 , which weakens scalability from the network operation point of view.

12.4.3 Fine Two-Phase Routing

Oki and colleagues presented an IP load-balanced routing scheme based on two-phase routing over shortest paths for the pipe model [33, 34], [34, 35] and the hose model [35, 36]; it is an extended version of the original shortest-path-based TPR [31]. It is called fine two-phase routing (F-TPR). F-TPR more finely distributes traffic from a source node to intermediate nodes than the original TPR. F-TPR determines the distribution ratio to an intermediate node for each source-destination pair independently, while the distribution ratio to an intermediate node are commonly determined among all the source-destination pairs. F-TPR performs load balancing and each flow is routed according to the OSPF protocol, which is an existing IP routing protocol, in

two stages across intermediate nodes.

For the hose model, since F-TPR solves a nonlinear programming problem [35], it is applied only to networks with fewer than 10 nodes so as to yield practical levels of computation complexity. In the hose model, it is observed that F-TPR outperforms TPR at the cost of higher implementation complexity for the forwarding functions.

F-TPR is applied to the hose and pipe models [33, 34, 35, 36]. To determine an optimum set of k_m^{pq} for the pipe model, an LP formulation is derived. For the hose model, F-TPR solves a nonlinear programming problem [35] to determine the distribution ratios. F-TPR greatly reduces the network congestion ratio compared with TPR. In addition, F-TPR provides comparable routing performance to MPLS-TE in the pipe model [34], [36].

12.4.4 Features of Routing Schemes

[Table 12.5](#) summarizes the features of five routing schemes: MPLS-TE, F-TPR, TRP, S-OSPF, and OSPF.

[Table 12.5](#) Comparison of Routing Schemes.

	MPLS Routing	F-TPR over Shortest Paths	TRP over Shortest Paths	S-OSPF	OSPF (Regular)
Tunnel Routing	LSP tunnel Explicit routing	IP tunnel Combination of two shortest paths	IP tunnel Combination of two shortest paths	Not required Combination of source distribution and shortest path from neighbor node	Not required Shortest path
Additional functional requirements over original OSPF	Significant	Medium or more	Small	NA	
Number of possible distributed routes	Not limited	Equal to the number of possible intermediate nodes	Equal to the number of possible intermediate nodes	Equal to the number of neighbor nodes	One or multiple with ECMP

References

1. Juttner, A., Szabo, I. and Szentesi, A., “On Bandwidth Efficiency of the Hose Resource Management Model in Virtual Private Networks,” IEEE Infocom 2003, pp. 386-395, Mar./Apr. 2003.
2. Duffield, N.G., Goyal, P., Greenberg, A., Mishra, P., Ramakrishnan, K.K. and van der Merwe, J.E., “Resource Management with Hoses: Point-to-Cloud Services for Virtual Private Networks,” IEEE/ACM Trans. on Networking, vol. 10, no. 5, pp. 679-692, Oct. 2002.
3. Kumar, A., Rastogi, R., Silberschatz, A. and Yener, B., “Algorithms for Provisioning Virtual Private Networks in the Hose Model,” the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, pp. 135-146, 2001.
4. Chu, J. and Lea, C., “Optimal Link Weights for Maximizing QoS Traffic,” IEEE ICC 2007, pp. 610-615, 2007.

5. Chu, J. and Lea, C., "Optimal Link Weights for IP-Based Networks Supporting Hose-model VPNs," IEEE/ACM Trans. on Networking, vol. 17, no. 3, pp. 778-788, Mar. 2009.
6. Kodialam, M., Lakshman, T.V., Orlin, J.B. and Sengupta, S., "Pre-Configuring IP-over-Optical Networks to Handle Router Failures and Unpredictable Traffic," IEEE Infocom 2006, Apr. 2006.
7. Kodialam, M., Lakshman, T.V., Orlin, J.B. and Sengupta, S., "Oblivious Routing of Highly Variable Traffic in Service Overlays and IP Backbones," IEEE/ACM Trans. Networking, vol. 17, no. 2, pp. 459-472, April 2009.
8. Medina, A., Taft, N., Salamatian, K., Bhattacharyya, S. and Diot, C., "Traffic Matrix Estimation: Existing Techniques and New Directions," ACM SIGCOMM 2002, pp. 161-174, Aug. 2002.
9. Zhang, Y., Roughan, M., Duffield, N. and Greenberg, A., "Fast Accurate Computation of Large-scale IP Traffic Matrices from Link Loads," ACM SIGMETRICS 2003, pp. 206.217, Jun. 2003.
10. Nucci, A., Cruz, R., Taft, N. and Diot, C., "Design of IGP Link Weight Changes for Estimation of Traffic Matrices," INFOCOM 2004, vol. 4, pp. 2341.2351, Mar. 2004.
11. Ohsita, Y., Miyamura, T., Arakawa, S., Ata, S., Oki, E., Shiromoto, K. and Murata, M., "Gradually Reconfiguring Virtual Network Topologies Based on Estimated Traffic Matrices," INFOCOM 2007, pp. 2511.2515, May 2007.
12. Applegate, D. and Cohen, E., "Making Routing Robust to Changing Traffic Demands: Algorithms and Evaluation," IEE/ACM Trans. Networking, vol. 14, no. 6, pp. 1193-1206, 2006.
13. Towles, B. and Dally, W.J., "Worst-Case Traffic for Oblivious Routing Functions," IEEE Computer Architecture Letters, vol. 1, no.1, 2002.
14. Applegate, D. and Cohen, E., "Making Intra-domain Routing Robust to Changing and Uncertain Traffic Demands: Understanding fundamental Tradeoffs," Proc. of SIGCOMM'03, 2003.
15. Bienkowski, M., Korzeniowski, M. and Räcke, H., "A Practical Algorithm for Constructing Oblivious Routing Schemes," Proc. of SPAA'03, 2003.
16. Oki, E. and Iwaki, A., "Performance Comparisons of Optimal Routing by Pipe, Hose, and Intermediate Models," IEEE Sarnoff 2009, Mar./Apr. 2009.
17. Oki, E. and Iwaki, A., "Performance of Optimal Routing by Pipe, Hose,

- and Intermediate Models," IEICE Trans. Commun., vol. E93-B, no. 5, pp. 1180-1189, May 2010.
- 18.** Wang, Y. and Wang, Z., "Explicit Routing Algorithms for Internet Traffic Engineering," IEEE International Conference on Computer Communications and Networks (ICCCN), 1999.
- 19.** Awdanche, D., Malcolm, J., Agogbua, J., O'Dell, M. and McManus, J., et al., "Requirements for Traffic Engineering Over MPLS," IETF RFC 2702, Sep. 1999.
- 20.** Rosen, E., Viswanathan, A. and Callon, R., "Multiprotocol Label Switching Architecture," RFC 3031, Jan. 2001.
- 21.** Moy, J., "OSPF version 2," IETF RFC 2328, Apr. 1998.
- 22.** Cisco, "Configuring OSPF," Online, <http://www.cisco.com>.
- 23.** Fortz, B. and Thorup, M., "Optimizing OSPF/IS-IS Weights in a Changing World," IEEE Journal on Selected Areas in Communications, vol. 20, no. 4, pp. 756-767, 2002.
- 24.** Fortz, B., Rexford, J. and Thorup, M., "Traffic Engineering with Traditional IP protocols," IEEE Commun. Mag., vol. 40, no. 10, pp. 118-124, 2002.
- 25.** Buriol, L.S., Resende, M.G.C., Ribeiro, C.C. and Thorup, M., "A Hybrid Genetic Algorithm for the Weight Setting Problem in OSPF-IS-IS Routing," Networks, vol. 46, no. 1, pp. 36-56, Aug. 2005.
- 26.** Reichert, C. and Magedanz, T., "A fast Heuristic for Genetic Algorithms in Link Weight Optimization," Lecture Notes in Computer Science, vol. 3266, pp.144-153, 2004.
- 27.** Wang, Y. and Ito, M.R., "Dynamics of Load Sensitive Adaptive Routing," IEEE International Conference on Communications (ICC), 2005.
- 28.** Kamrul, I.M. and Oki, E., "PSO: Preventive Start-Time Optimization of OSPF Link Weights to Counter Network Failure," IEEE Commun. Letters, vol. 14, no. 6, 2010.
- 29.** Mishra, A.K. and Sahoo, A., "S-OSPF: A Traffic Engineering Solution for OSPF based on Best Effort Networks," IEEE Globecom 2007, pp. 1845-1849, 2007.
- 30.** Oki, E. and Iwaki, A., "Load-Balanced IP Routing Scheme Based on Shortest Paths in Hose Model," IEEE Trans. Commun., vol. 58, no. 7, pp.

2088 -2096, 2010.

- 31.** Antic, M. and Smiljanic, A., “Oblivious Routing Scheme Using Load Balancing Over Shortest Paths,” IEEE ICC 2008, 2008.
- 32.** Antic, M. and Smiljanic, A., “Routing with Load Balancing: Increasing the Guaranteed Node Traffics,” IEEE Commun. Let., vol. 13, no. 6, pp. 450-452, Jun. 2009.
- 33.** Oki, E. and Iwaki, A., “Fine Two-Phase Routing with Traffic Matrix,” 18th International Conference on Computer Communications and Networks (ICCCN 2009), Aug. 2009.
- 33.** Oki, E., Iwaki, A., Urushidani, S. and Aoki, M., “Fine Two-Phase Routing Over Shortest Paths Without Traffic Splitting,” IEEE ICC 2010, May 2010.
- 34.** Oki, E. and Iwaki, A., “F-TPR: Fine two-phase IP Routing Scheme over Shortest Paths for Hose Model,” IEEE Commun. Letters, vol. 13, no. 4, pp. 277-279, Apr. 2009.
- 35.** Oki, E. and Iwaki, A., “Optimization of IP Load-Balanced Routing for Hose Model,” IEEE 21st International Conference on Tools with Artificial Intelligence, Nov. 2009.

Chapter 13

IP Network Security

The security of networks can be assessed by reviewing the vulnerability of the deployed protocols to exploits. Threats exploit these vulnerabilities, misusing the networks to obtain some benefit or disrupt the service. This chapter discusses some popular threats and how they exploit protocol vulnerabilities. Some counter measures based on algorithms that either work as an application or as a protocol at the network or transport layers are discussed.

13.1 Introduction

A denial-of-service (DoS) attack or distributed denial-of-service (DDoS) attack is an attempt to exhaust the resources of a computer or network to prevent the victim from doing useful work. DoS attacks may involve gaining unauthorized access to network or computing resources [1]. DoS attacks have been mostly considered on servers, where the service provided to the intended users becomes noticeably diminished or unavailable. The victim can be a network server, a client or router, a network link or an entire network, an individual Internet user or a company doing business using the Internet, an Internet service provider (ISP), country, or any combination of or variant on these. One common form of attack involves saturating the target (victim) machine with external communications requests, such that it becomes unable to keep up with responding to legitimate requests, or it responds slow enough to be rendered unavailable. In general terms, DoS attacks are implemented by either forcing the targeted computer(s) to a reset state, exhausting its resources, or obstructing the communication media between the intended users and the victim so that they can no longer communicate adequately.

There are two principal classes of attacks: logic attacks and flooding attacks. Logic attacks exploit existing software flaws to cause remote servers to crash or substantially degrade their performance. An example of a logic DoS attack is the so-called ping of death. Flooding attacks overwhelm the

victim's resources, such as CPU use, memory, or network resources, by sending large numbers of requests. The following discussion focuses solely on flooding attacks.

A flooding attack can be performed in two different ways [2]. For example, a single attacker can send 183 small packets towards a victim at a rate beyond its processing capacity or to the capacities of the access network. Similarly, multiple attackers can issue an attack by leveraging the resources of multiple hosts, where these hosts are not precisely in agreement to perform such use. An attacker, as a master, can compromise a set of Internet hosts (using manual or semiautomated methods) and install a small (attack) daemon on each host, to make the host act as a *zombie* or *bot*. This daemon typically contains both the code for generating a variety of attacks and some basic communications infrastructure to allow remote control of the hosts. With variants of this basic architecture, an attacker orchestrates a coordinated attack through the use of a few or large number of zombies. In DoS attacks, attackers commonly spoof (i.e., fake) the source IP address field to conceal the location of the attacking host(s).

13.2 Detection of Denial-of-Service Attack

Denial of service attack detection aims to distinguish malicious packets (as traffic or requests) from legitimate packets. For example, consider a situation where a large number of clients all request a Web service and, at the same time, a DoS attack maliciously floods this Web service. The attack is usually difficult to identify because determining if a packet belongs to legitimate traffic or to an attack is complex. Therefore, legitimate user activity can be easily taken for a flooding attack, or the flooding attack can be seen as legitimate traffic. For this reason, there is an interest in resolving this hard problem. Because there is no innate Internet mechanism for performing malicious traffic discrimination, the best alternative is to install attack detectors to monitor real-time traffic, rather than relying on static traffic load predictions. DoS attack detection approaches are classified into the following three categories [3].

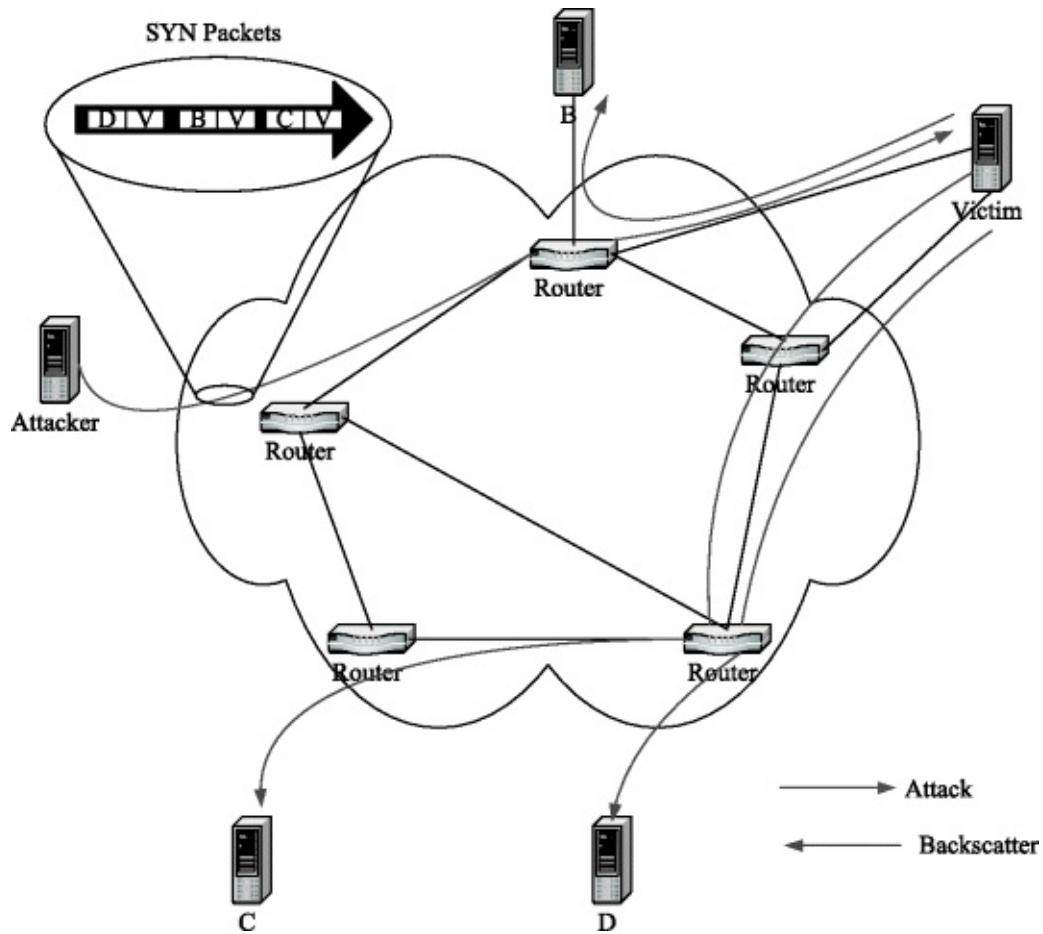
- Activity profiling [2, 4, 5]
- Sequential change-point detection [6, 7]
- Wavelet analysis [8, 9]

Two popular active profiling approaches are presented in this chapter.

13.2.1 Backscatter Analysis

Attackers commonly spoof the source IP address field to conceal the location of the attacking host. Carl and colleagues analyzed unsolicited packets [2]; they reviewed responses to packets created by attackers using the victim's IP address as a source address, and a randomly selected destination address with the objective of forcing a response from the randomly selected destination. This project was called backscatter, as it observes the backscattered traffic in a network. When a spoofed packet arrives at the victim, the victim usually sends what it believes to be an appropriate response to the faked IP address. Because the source address of the host that will reflect packets to the victim are selected at random, the victim's responses are distributed across the entire Internet address space with equal probability. This approach is illustrated in [Figure 13.1](#).

Figure 13.1 Backscatter scheme in action. The attacker sends a series of SYN packets toward the victim V, using a series of randomly spoofed source addresses named C, B, and D. Upon receiving these packets, the victim responds by sending SYN/ACKs to each of the spoofed hosts.



Assuming per-packet random source addresses, reliable delivery, and one response generated for every packet in an attack, the probability of a given host on the Internet to receive at least one unsolicited response from the victim is $\frac{m}{2^{32}}$ during an attack comprising m packets. Similarly, if distinct IP addresses are monitored, the expectation of observing an attack is:

$$(13.1) \quad E(X) \leq \frac{nm}{2^{32}}$$

This work was performed under the assumption that a large sample can be obtained by observing a large enough address range, and this sample may include all such DoS activity on the Internet. Contained in these samples are the identity of the victim, information about the kind of attack, and a timestamp from which the attack duration can be estimated. Moreover, the average arrival rate of unsolicited responses directed at the monitored address range can be used to estimate the actual rate of the attack being directed at the victim,

$$(13.2) \quad R \geq R' \frac{2^{32}}{n}$$

where R' is the measured average inter arrival rate of backscatter from the victim and R is the extrapolated attack rate in packets per second. There are three assumptions in the approach:

- *Address uniformity*: Attackers spoof source addresses at random.
- *Reliable delivery*: Attack traffic is delivered reliably to the victim and backscatter is delivered reliably to the monitor.
- *Backscatter hypothesis*: Unsolicited packets observed by the monitor represent backscatter.

The backscatter analysis approach classifies attacks into two distinct approaches to determine the network under a DoS (or DDoS) attack. In the first approach, a flow-based attack analysis for individual attacks quantifies the number, time scale, and its kind. Here, a flow is defined as a series of consecutive packets sharing the same target IP address and IP protocol. The time space between two adjacent packets can be as long as five minutes. A flow is considered active if it consists of at least 100 attack packets with at least 60 seconds of attack duration. Moreover, each flow should contain packets sent to more than one victim. In the second approach, an event-based attack analysis identifies the intensity of the attack (i.e., number of simultaneous attacks or the attack rates) over short time scales. Here, an event is defined within a one-minute duration.

13.2.2 Multilevel Tree or Online Packet Statistics

Multi level Tree for Online Packet Statistics (MULTOPS) [gil]2 uses disproportional packet rates to and from hosts and subnets as a heuristic scheme to detect (and potentially stop) DoS attacks. It uses a tree-shaped data structure that keeps track of packet rates to and from those subsets of the IP address space. This approach tracks the acquired statistics that display disproportional behavior that is determined to be a DoS attack. In MULTOPS, packets are considered malicious (and, thus, may be dropped) if they are destined for a host or subnet from which too few packets are coming back. This heuristic is based on the assumptions that (1) most Internet traffic consists of packet flows, and (2) during normal operations, the packet rate of a flow from A to B is proportional to the flow rate going in the opposite direction (i.e., from B to A). Thus, during normal operation, the flow rate of

the traffic going in one direction is proportional to the packet rate of traffic going in the opposite direction. This heuristic can be easily validated on the TCP congestion control mechanism.

MULTOPS stores packet rate statistics for flows between hosts (or subnets) A and B using either A's IP address or B's IP address. In this way, MULTOPS can either establish the victim or the source(s) of the attack. Thus, MULTOPS can operate in victim-oriented mode or attacker-oriented mode, respectively. In the victim-oriented mode, MULTOPS attempts to identify the IP address of the victim of an attack. In the attacker-oriented mode, MULTOPS attempts to identify the IP address(es) of the attacker(s). The common objective of these two modes becomes important to drop the attacking packets: either packets going to the victim or packets coming from the attacker are dropped. In both cases, the objective is to stop the attack. In the victim-oriented mode, the IP address of the victim is detected; in the attacker-oriented mode, the IP address of the attacker(s) is detected. In the remainder of this discussion, it is assumed that MULTOPS runs in the victim-oriented mode unless specified otherwise.

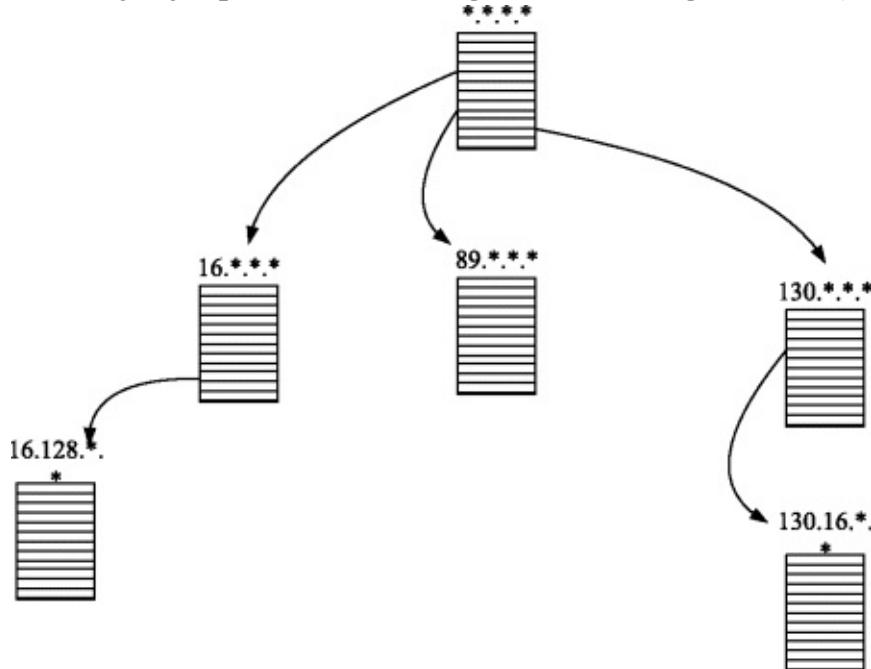
MULTOPS expects two streams of IP packets as input, each connected to a different network interface. The destination addresses of packets going in one direction (*forward packets*) are inspected. In the opposite direction, the source addresses of the (*reverse packets*) are inspected.¹ It presents a query interface that returns an approximation (i.e., ratio) to $R(P)$, where P is the prefix of an IP address.

$$(13.3) \quad R(P) = \frac{\text{Forward packets towards dst IPaddr with } P}{\text{Reverse packets from src IP addr with } P}$$

In the victim-oriented mode, MULTOPS determines a victim's IP address by looking for prefixes for which $R(P)$ are greater than a selected threshold. Dropping packets with destination addresses matching such prefixes might defeat the attack. In the attacker-oriented mode, MULTOPS determines the addresses of attackers by looking for prefixes for which $R(P)$ is smaller than a selected threshold. In a similar way, dropping packets based on source addresses matching such prefixes might defeat the attack. In this way, MULTOPS collects packet rates to and from address prefixes so that, given a certain P , $R(P)$ can be calculated and organized in a 4-level 256-ary tree to conveniently cover the entire IPv4 address space, as shown in [Figure 13.3](#).

Figure 13.3 MULTOPS data structure. (Reprinted with permission from

Thomer M. Gil and Massimiliano Poletto, “MULTOPS: a data-structure for bandwidth attack detection,” In the Proceedings of the 10th USENIX Security Symposium, Washington D.C., August 2001.)

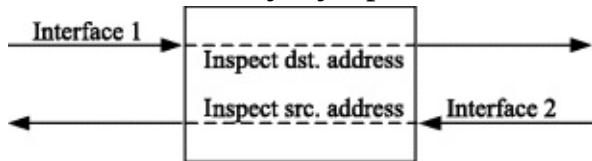


Here, each node in the tree is a table consisting of 256 records, each of which consists of 3 fields: to-rate, from-rate, and 1 pointer (potentially) pointing to a node in the next level of the tree. A table stores all packet rates to and from IP addresses with a common 0-bit, 8-bit, 16-bit, or 24-bit prefix, depending on the level of the tree. Deeper levels of the tree contain packet rates for addresses with a longer prefix. Thus, the root node contains the aggregate packet rates to and from address $0.*.*.*$, $1.*.*.*$, $2.*.*.*$, and so on. The above hierarchical data structure expands or contracts to keep track of more fine-grained packet rates, potentially down to per-IP address packet rates. For example, if the aggregate packet rate to or from subnet $130.17.*.*$ exceeds the $R(P)$ threshold (e.g., R_{max} , a new node is created to keep track of packet rates to and from subnets $130.17.0.*$, $130.17.1.*$, etc). Creating new nodes is called expansion. The reverse (i.e., removing nodes or entire subtrees) is called contraction. Contraction is done when the packet rate from and to a given IP address prefix drops below a certain threshold or when memory is running out.

So, as for the MULTOPS algorithm, each packet (or every n th packet) that is routed causes packet rates in applicable nodes in the tree to be updated; starting in the root, and going down to the deepest available node. Note that,

the first byte of the IP destination address of a forward packet (i.e., packet going through interface 1 in [Figure 13.2](#)) is used as an index in the root node to find the record in which to update the to-rate. For reverse packets (i.e., packet going through interface 2 in [Figure 13.2](#)) the first byte of the IP source address is used as an index in the root node to find the record in which to update the from-rate. The iterative update process works as follows:

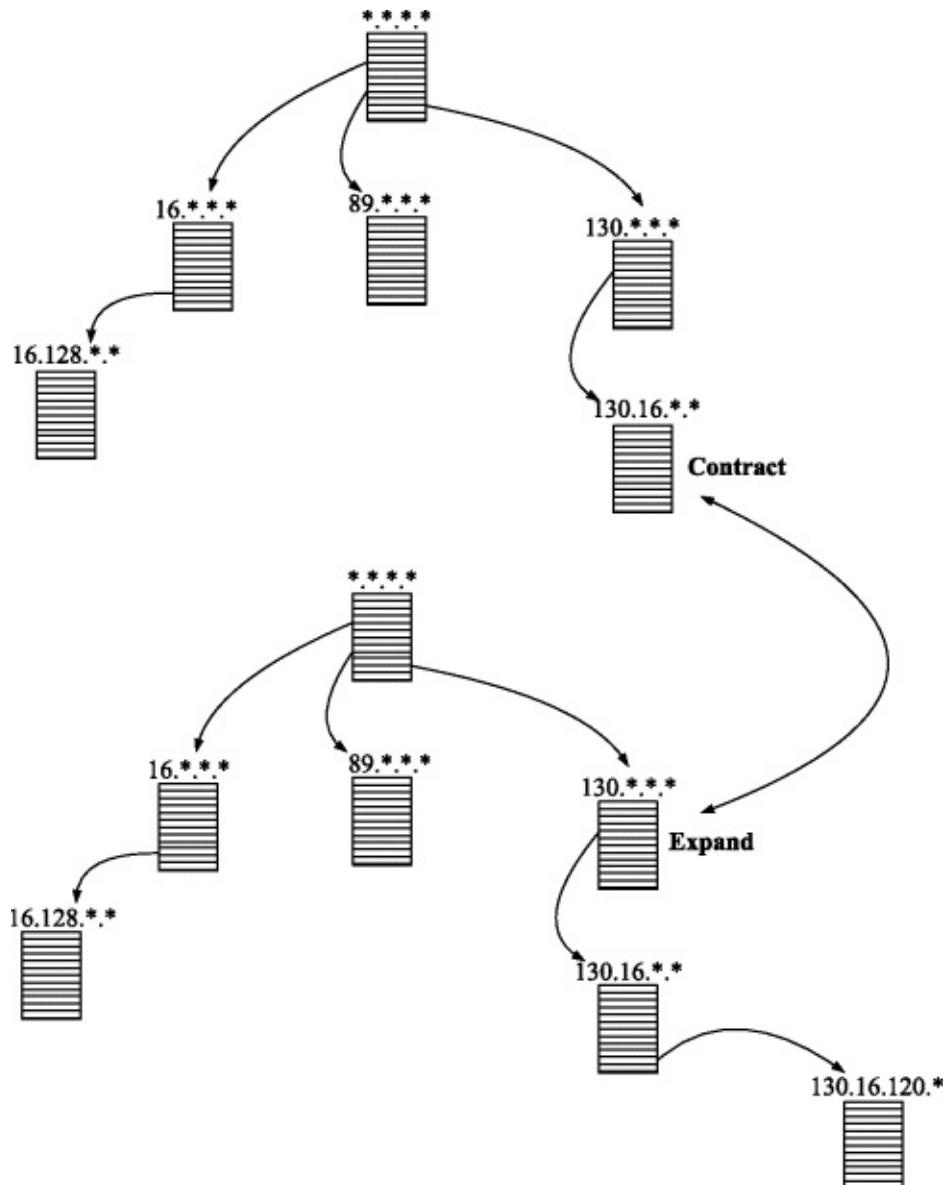
Figure 13.2 Schematic MULTOPS in victim-oriented mode. (Reprinted with permission from Thomer M. Gil and Massimiliano Poletto, “MULTOPS: a data-structure for bandwidth attack detection,” In the Proceedings of the 10th USENIX Security Symposium, Washington D.C., August 2001.)



- Check if the record has a child; the process descends down to the child.
- If there is no child, it is created when either the from-rate or the to-rate exceeds a certain threshold.
- In both case, the process will follow the pointer in the record to the child node. In this child node, the second byte of the IP address is used as an index to find the record and update the packet rates. This process may descend down to the deepest level in the tree where per-IP address packet rates are kept.

[Figure 13.4](#) shows an example of how the MULTOPS data structure expands and contracts using the above-mentioned algorithm. Expansion detects DoS attack activity toward the victim, and the number of nodes denotes the multitude of attack dimension.

Figure 13.4 Expansion and contraction. (Reprinted with permission from Thomer M. Gil and Massimiliano Poletto, “MULTOPS: a data-structure for bandwidth attack detection,” In the Proceedings of the 10th USENIX Security Symposium, Washington D.C., August 2001.)



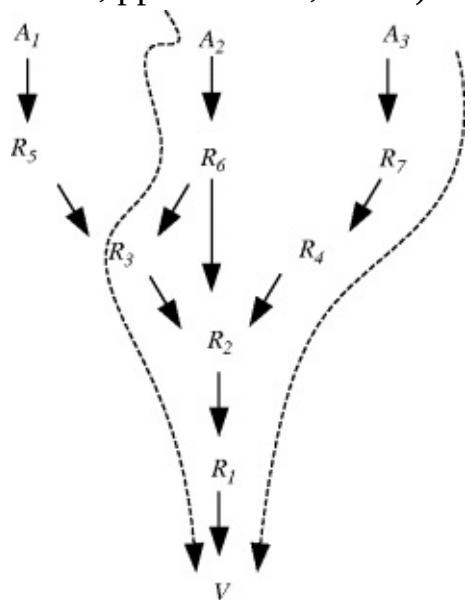
13.3 IP Traceback

As discussed above, a DoS attack is known as the consumption of network resources of a remote host or network by means of flooding malicious packets to deny or degrade network performance of its legitimate users. Malicious use of packets is prevalent in IP networks due to the stateless nature of its IP packet structure (i.e., the source address of an IP packet is not actually required for end-to-end delivery). Therefore, DoS attackers can use a spoofed IP address (i.e., fake source IP address) to generate unwanted traffic flows that impede the service of the targeted remote host to its valid users.

Identification of the source of a DoS (or any other attack) attack by determining the source of its malicious IP packets is called *IP traceback*.

[Figure 13.5](#) shows a segment of the network consisting of a victim node along with attack nodes and intermediate routers. This is a directed acyclic graph (DAG) rooted at V to represent the network as seen from victim V and a distributed DoS attack generated from hosts A_1 and A_3 . In this case, V could be either a single host under attack or an edge device of a network, such as a firewall. A node R_i represents the routers, referred to as upstream routers from V . This is also called *the map of upstream routers from V* .

[Figure 13.5](#) A segment of an IP network depicting a DoS attack path in reference to a remote host (i.e., victim) V . (2001 IEEE. Reprinted with permission from D.X. Song and A. Perrig, “Advanced and authenticated marking schemes for IP traceback,” In Proceedings of IEEE INFOCOM'01, vol. 2, pp. 878–886, 2001.)



For every router R_i , we refer to the set of routers that come immediately before R_i in the graph, as the children of R_i (e.g., R_3 , R_6 and R_4) are R_2 's children. The leaves A_i represent the potential attack origins, or attackers. The attack path from A_i is the ordered list of routers between A_i and V that the attack packet has traversed (e.g., the two dotted lines in the graph indicate two attack paths: $[R_6, R_3, R_2, R_1]$ and $[R_7, R_4, R_2, R_1]$). The distance of R_i from V on a path is the number of routers between R_i and V on the path (e.g.,

the distance of R_6 to V in the path $[R_6, R_3, R_2, R_1]$ is 3). The attack graph is the graph composed of the attack paths. The packets used in DDoS attacks are referred to as attack packets.

A router can be identified as a member of the path. This identification is referred to as a false positive if it is in the reconstructed attack graph but not in the real attack graph. Similarly, a false negative is referred to the identification of a router that is in the actual attack graph but not in the reconstructed attack graph. A solution to the IP traceback problem is called robust if it has a very low rate of false negatives and false positives.

In the IP traceback problem, the goal is to find, at least, a candidate attack path for each attacker that contains the true attack path as a suffix, since determining the complete attack path and the associated attack origin of each is complicated due to several practical limitations in the IP network [8]. For example, $(R_7, R_6, R_3, R_2, R_1)$ in [Figure 13.5](#) is a candidate path for attacker A_i because it contains the true attack (R_6, R_3, R_2, R_1) path as a suffix.

13.3.1 IP Traceback Solutions

There are several approaches that have been introduced to handle the IP traceback problem. They can be categorized into the following major approaches:

- Link testing [7]
- Probabilistic packet marking [8, 10]
- Deterministic packet marking [9, 12]
- Router based approach [13]
- ICMP traceback [14]

Two popular approaches to IP traceback based on probabilistic packet marking are presented in Sections 13.4 and 13.5.

13.4 Edge Sampling Scheme

In the IP packet marking approach, routers (probabilistically) add encoded partial path information into the traffic packets based on their own perspective before forwarding them toward the destination. In edge sampling [8], partial path information is encoded with network edge information that is defined by two IP addresses and the hop count. To achieve this, two static IP

address-sized fields, start field and end field, along with a small static distance field are reserved in each packet to represent the routers (through network edge) along the attack path. The distance refers to the distance from the network edge to the victim.

[Figure 13.6](#) shows the probabilistic approach of edge sampling by routers. When a router decides to mark a packet, it writes the router address into the start field and writes a zero into the distance field. In a router, a distance field with a zero value indicates that the packet was marked by the previous router. In this case, the router writes its address into the end field –thereby representing the edge between itself and the previous router. Finally, if the router doesn't mark the packet, then it always increments the distance field. A marked packet that arrives at the victim host has a distance greater than or equal to the true attack path. In the probabilistic marking scheme, when a sufficient number of packet samples encoding all possible edges of an attack path are received by the victim, it is possible to reconstruct the attach path leading to the attacking sources, as shown in [Figure 13.5](#). The path reconstruction procedure performed by the victim is depicted in [Figure 13.7](#).

Figure 13.6 Probabilistic edge sampling algorithm.

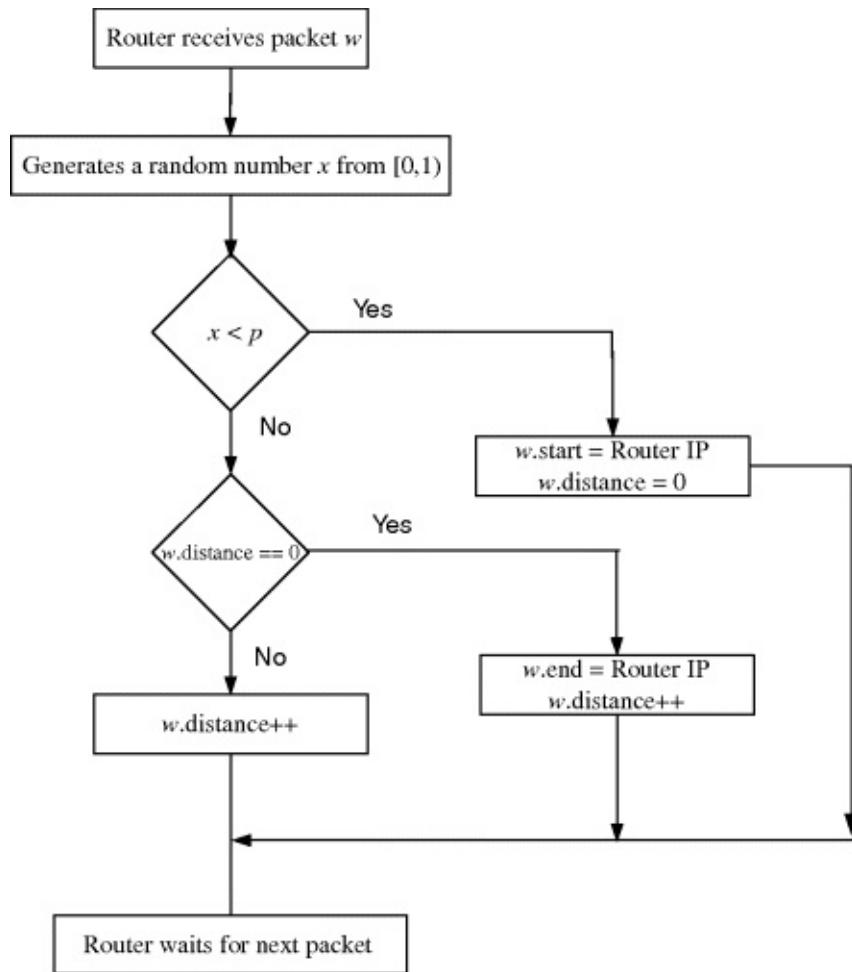
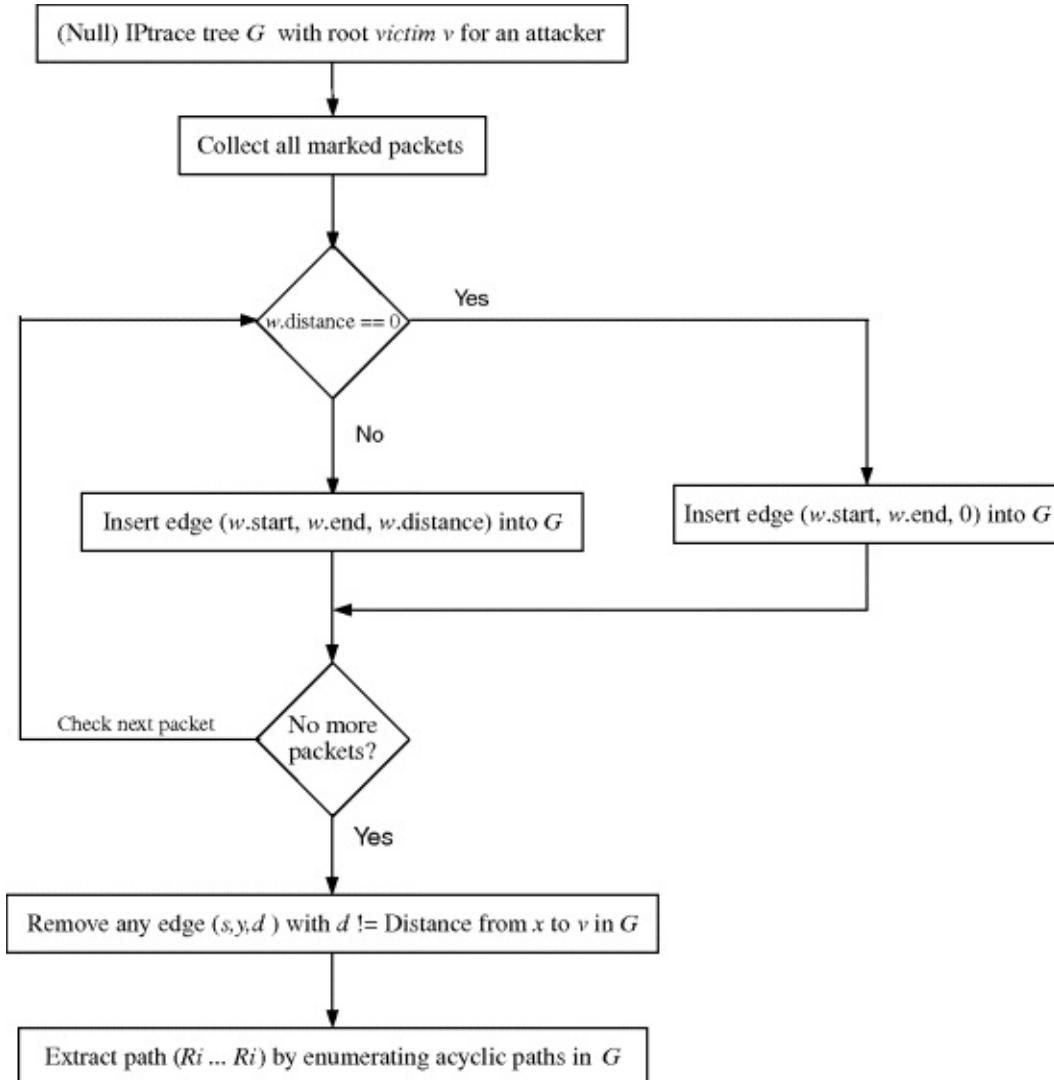


Figure 13.7 Attack path reconstruction algorithm.



In the edge sampling scheme, the probability of a router marking a packet closer to the victim is greater than the probability of a distant router marking it, as the packet marking procedure is modeled by the binomial property $p(1 - p)^{d-1}$, where d is the attacker–victim distance. Therefore, given a packet marking probability p and a distance d , the number of packets X required for the victim to reconstruct a path of length d is bounded by the following.

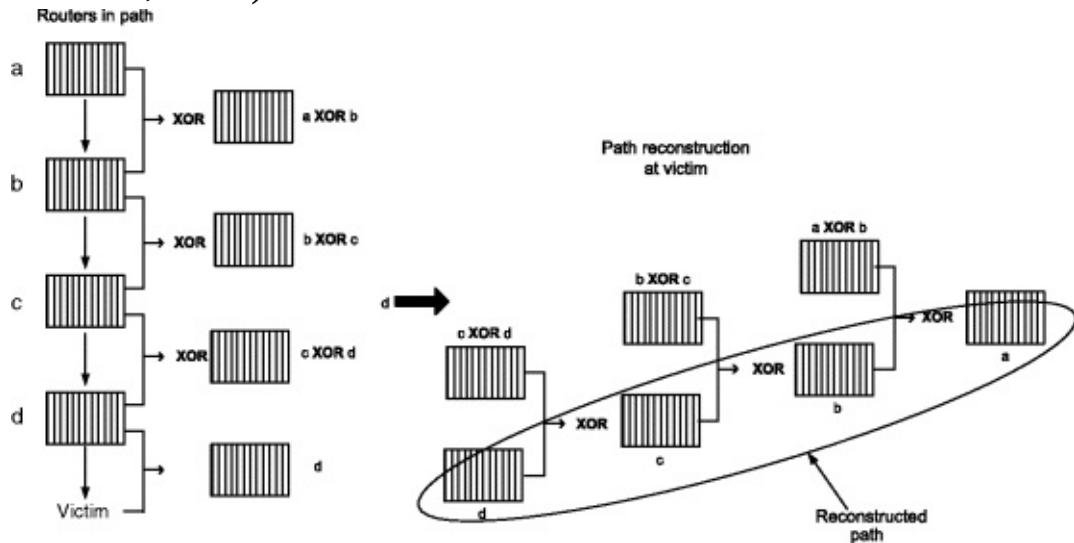
$$(13.4) \quad E(X) \leq \frac{\ln(d)}{p(1 - p)^{d-1}}$$

For example, if $p = \frac{1}{10}$ and the attack path has a length of 10, then a victim can typically reconstruct this path after receiving 75 packets from the attacker.

The edge sampling algorithm requires space for 72 bits in every IP packet

(two 32-bit IP addresses and 8 bits for the distance to represent the theoretical maximum number of hops allowed using IP). In order to prevent the IP packets from growing in size and reduce the space requirement, the scheme encodes the above information in the 16-bit IP identification field of the IP header. The encoding scheme for edge sampling is thus performed, as shown [Figure 13.8](#).

Figure 13.8 Edge data can be communicated in half the space by sending the XOR of the two nodes (i.e., router IP addresses) making up an edge, rather than sending each node separately. (2001 IEEE. Reprinted with permission from D.X. Song and A. Perrig, “Advanced and authenticated marking schemes for IP traceback,” In Proceedings of IEEE INFOCOM’01, vol. 2, pp. 878–886, 2001.)



First, each edge is encoded in half the space by representing it as the exclusive-or (XOR) of the two IP addresses making up the edge, as depicted in [Figure 13.8](#). When some router decides to mark a packet it writes its address a into the packet. The following router b notices that the distance field is 0 and (assuming it does not mark the packet) it reads a from the packet, XORs this value with its own address and writes the resulting value, $a \oplus b$, into the packet. The resulting value is considered the edge id for the edge between a and b . The edge ids in the packets received by the victim always contain the XOR of two adjacent routers, except for samples from routers one hop away from the victim, which arrive unmodified. Since $b \oplus a \oplus b = a$, marked packets from the final router can be used to decode the id of the previous edge, and so on, hop-by-hop until the first router is reached.

Second, routers send the edge id, which is subdivided into k non overlapping fragments. Therefore, when a router decides to mark a packet, it selects one of these fragments at random and stores it in the packet. A few additional bits ($\log_2 k$) store the offset of this fragment within the original address. When enough packets are eventually sent by the attacker, the victim receives all fragments from all edge ids. The expected number of packets X required to reconstruct an attack path of length d with a fragment size k is bounded by [Equation 13.5](#).

$$(13.5) \quad E(X) \leq \frac{k \cdot \ln(kd)}{p(1-p)^{d-1}}$$

13.5 Advanced Marking Scheme

The so-called *advanced marking* scheme proposes a variation of edge sampling schemes in the way it encodes the edge information (i.e., IP address) between routers to accommodate DDoS attacks, where multiple attack sources participate in service degradation of a single victim [10]. The advanced marking scheme also assumes that if the victim knows the map of its upstream routers, it does not require the full IP address in the packet marking that facilitates communication and computation efficiency. The above assumption is based on the fact that available network mapping tools (e.g., Skitter from CAIDA [15] and the traceroute-based tool from Lucent Bell Labs [16]) is a possible way to obtain a large map of upstream routers for a victim. Moreover, it is not necessary to have a very accurate and updated map as long as the graph is contained in it.

[Figure 13.9](#) shows the encoding method in the advanced marking scheme, where each router marks a packet with a 16-bit edge id in the IP identification field. The 16-bit edge-id consists of an 11-bit hashed IP address along with a 5-bit distance field. [Figure 13.10](#) shows the packet marking algorithm for the advanced scheme.

[Figure 13.9](#) Encoding in advanced marking scheme. (2001 IEEE. Reprinted with permission from D.X. Song and A. Perrig, “Advanced and authenticated marking schemes for IP traceback,” In Proceedings of IEEE INFOCOM'01, vol. 2, pp. 878–886, 2001.)

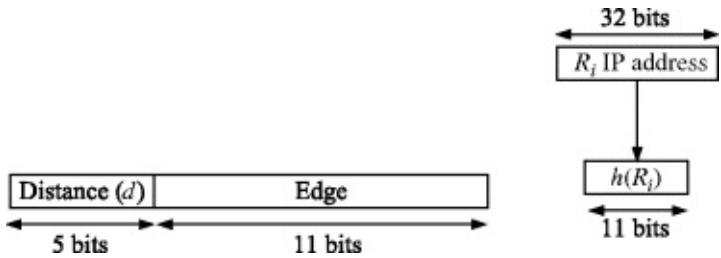
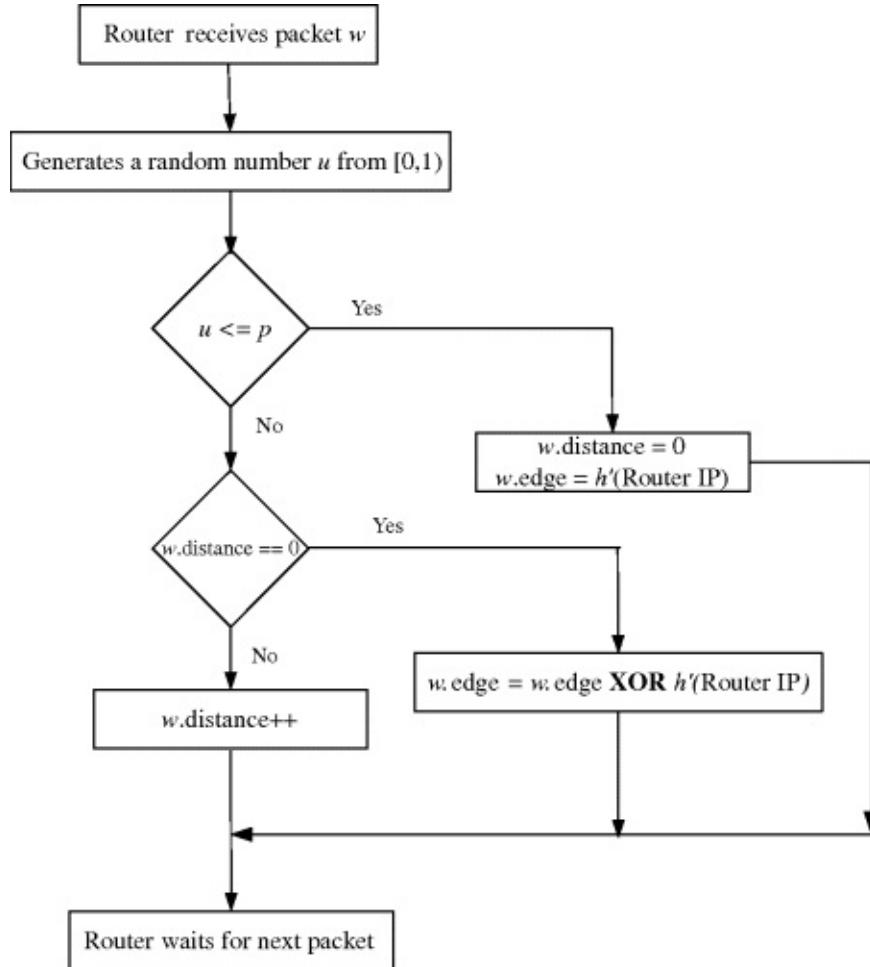
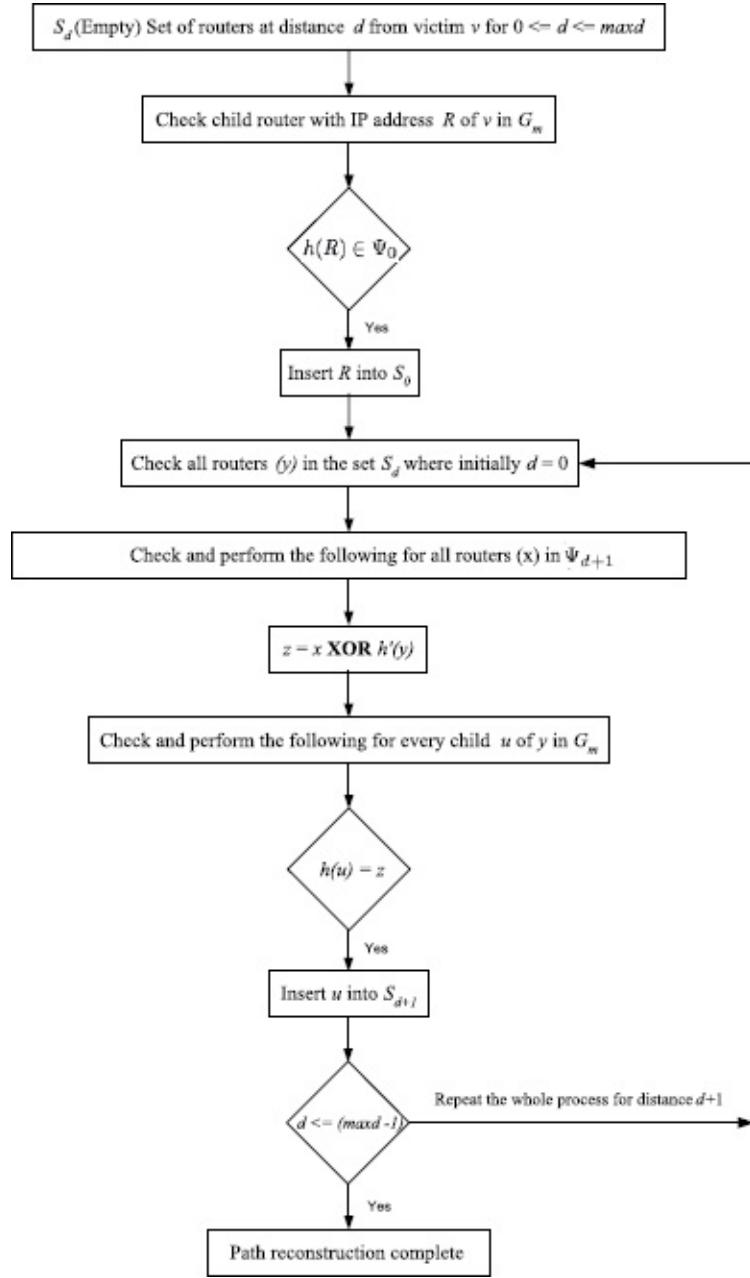


Figure 13.10 Advanced probabilistic edge sampling algorithm.



The advanced marking scheme is similar to that of the encoded edge sampling scheme, however, without the fragmentation of the encoded edge id. Also, this scheme uses two different hash functions, $h(\cdot)$ and $h'(\cdot)$, for encoding. Because the hash function encoding is used, the expected number of packets is required to reconstruct an attack path of length d , as defined in [Equation 13.1](#). This is equivalent to the unencoded and unfragmented version of the edge sampling scheme. The procedure for path reconstruction of the advanced marking scheme is shown in [Figure 13.11](#).

Figure 13.11 Advanced attack path reconstruction algorithm.



According to [Figure 13.11](#), the victim uses the upstream router map G_m (i.e., a-priori knowledge of upstream routers) as a road map and performs a breadth-first search from the root (i.e., itself). In this algorithm, S_d denotes the set of routers at distance d to the victim in the reconstructed attack graph. The set of edge fields marked with a distance d is denoted as Ψ_d (do not include duplicates). Initially, the victim enumerates all routes one hop away from itself (i.e., $d = 0$) in the G_m and checks which routers have the hash

value of their IP addresses, $h(R_i)$, matched with the edge fields in Ψ_0 , and denotes the set of matched IP addresses as S_0 . Therefore S_0 is the set of routers one hop away from the victim in the reconstructed attack graph. Then for each edge x in Ψ_{d+1} and for each element y in S_d , the victim decodes $z = x \oplus h'(y)$ and checks whether any child R_j of y in G_m has the hash value of its IP address, $h(R_j)$, equal to x . If the victim finds a matched IP address R_u , then it adds R_u to the set S_{d+1} (initially S_{d+1} is empty). The victim repeats the steps until it reaches the maximal distance marked in the packets, denoted as $maxd$.

There is another version of the advanced marking scheme where the encoding of the edge id consists of a 3-bit flag and 5-bit distance field along with an 8-bit hashed IP address to reduce false positives (i.e., appearance of a router non-existent in the attack path) in the constructed attack path. The modified scheme also introduces two authenticated packet marking schemes: authentication with message authentication codes (MAC) and time-released key chains. Authenticated packet marking helps to protect against mischievous activities (e.g., forging of packet marking) of compromised routers in the network upstream.

Note

[1. Figure 13.2](#) illustrates this. Exchanging the network interfaces switches between attacker-oriented and victim-oriented mode.

References

1. Handley, M. and Rescorla, E., “Internet Denial-of-Service Considerations,” IETF RFC 4732, Nov. 2006.
2. Gil, T.M. and Poletto, M., “MULTOPS: A Data-Structure for Bandwidth Attack Detection,” Proc. 10th USENIX Security Symposium, Aug. 2001.
3. Carl, G., Kesidis, G., Brooks, R.R. and Rai, S., “Denial-of-Service Attack-Detection Techniques,” IEEE Internet Computing, vol. 10 no. 1, pp. 82-89, Feb. 2006.
4. Feinstein, L. and Schnackenberg, D., “Statistical Approaches to DDoS attack Detection and Response,” Proc. DARPA Information Survivability Conf. and Exposition, vol. 1, pp. 303-314, 2003.

5. Blazek, R.B., Kim, H., Rozovskii, B. and Tartakovsky, A., “A Novel Approach to Detection of Denial-of-Service Attacks via Adaptive Sequential and Batch-Sequential Change-Point Detection Methods,” Proc. IEEE Workshop Information Assurance and Security 2001, pp. 220-226, 2001.
6. Wang, H., Zhang, D. and Shin, K., “Detecting SYN Flooding Attacks,” Proc. IEEE INFOCOM, pp. 1530-1539, 2002.
7. Burch, H. and Cheswick, B., “Tracing Anonymous Packets to Their Approximate Source,” Unpublished paper, Dec. 1999.
8. Savage, S., Wetherall, D., Karlin, A. and Anderson, T., “Practical Network Support for IP Traceback,” Proc. ACM SIGCOMM Conference, Stockholm, Sweden, pp. 295-306, Aug. 2000.
9. Belenky, A. and Ansari, N., “On Deterministic Packet Marking,” vol. 51, no. 10, pp. 2677-2700, Jul. 2007.
10. Song, D.X. and Perrig, A., “Advanced and Authenticated Marking Schemes for IP Traceback,” Proc. IEEE INFOCOM, vol. 2, pp. 878-886, Apr. 2001.
11. Rayanchu, S.K. and Barua, G., “Tracing Attackers with Deterministic Edge Router Marking (DERM),” Proc. 1st Intl. Conf. on Distributed Computing and Internet Technology, pp. 400-409, 2004.
12. Shokri, R., Varshovi, A., Mohammadi, H., Yazdani, N. and Sadeghian, B., “DDPM: Dynamic Deterministic Packet Marking for IP Traceback,” Proc. IEEE International Conference on Networks, 6 p, Sep. 2006.
13. Snoreren, A.C., Partridge, C., Sanchez, L.A., Jones, C.E., Tchakountio, F., Schwartz, B., Kent, S.T. and Strayer, W.T., “Single-Packet IP Traceback,” IEEE/ACM Trans. Netw., vol. 10, no. 6, pp. 721-734, Jan. 2003.
14. Bellovin, S.M., “ICMP Traceback Messages,” Internet Draft, draft-bellovin-itrace-00.txt, Work in Progress, Mar. 2000.
15. McRobb, D., “Skitter,” Cooperative Association for Internet Data Analysis (CAIDA), <http://www.caida.org/tools/measurement/skitter>, Aug. 2010.
16. Burch, H. and Cheswick, B., “Mapping the Internet,” IEEE Computer, vol. 32, no. 4, pp. 97-98, Apr. 1999.

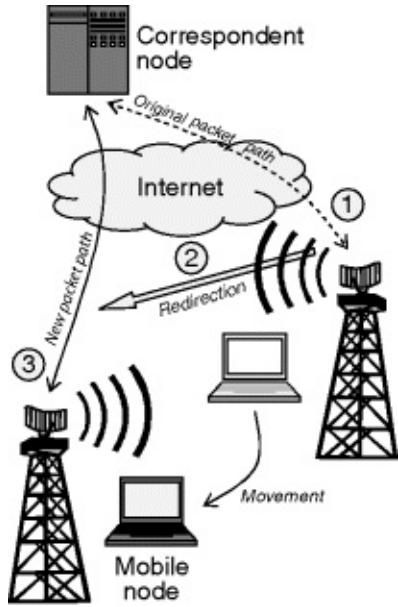
Chapter 14

Mobility Support for IP

Patterns of Internet use have been evolving rapidly in recent years, shifting from delay-and bandwidth-tolerant applications like Web browsing, electronic mail, and file transfer toward more delay-and loss-sensitive applications like audio and video streaming or broadcasts [1], as well as real-time services such as Internet telephony or videoconferencing [2, 3]. At the same time, people are increasingly expecting these services to be available anywhere, anytime, and through any access technology. These developments are the onset of a converged Internet, embracing wireless home and enterprise networks, hotspots in coffee shops and airport lounges, as well as wide-area cellular networks.

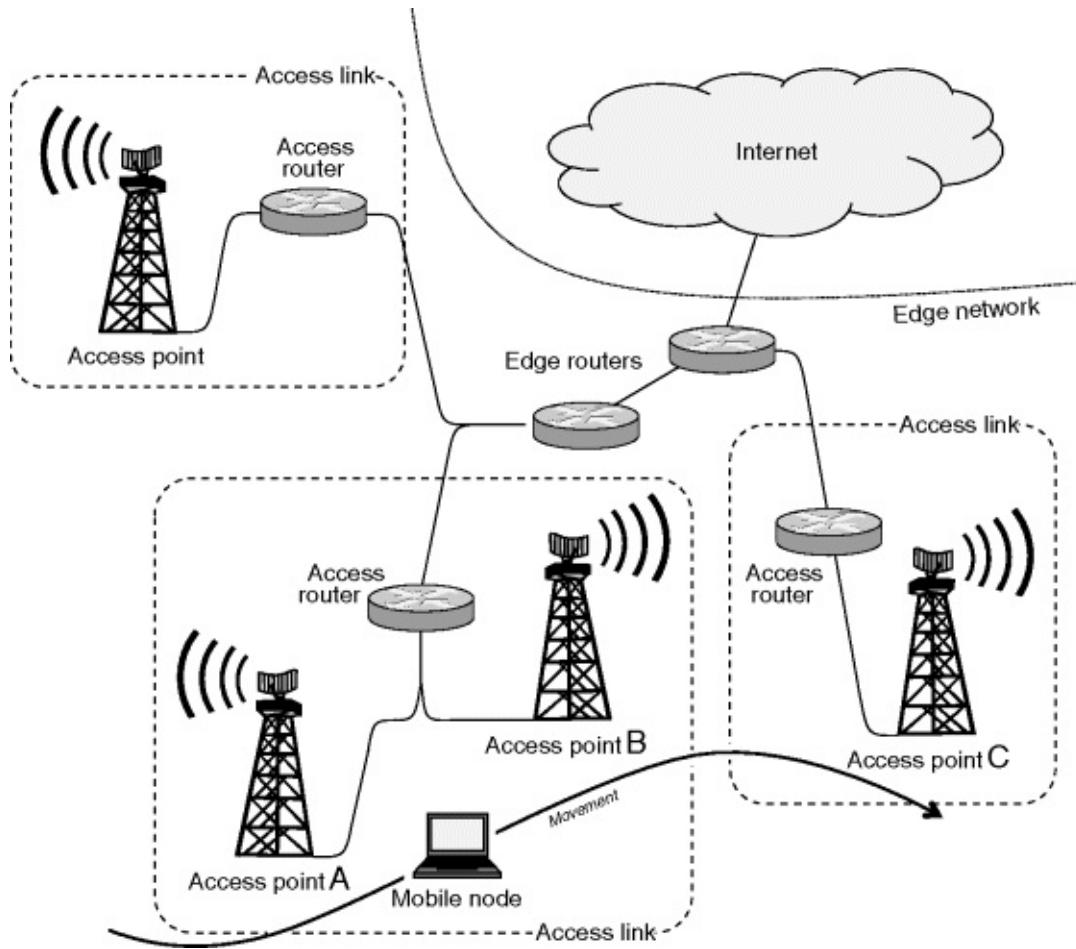
The new heterogeneity in access technologies and the increasing mobility of Internet users calls for an efficient mechanism to handle active communication sessions among different points of attachment to the Internet. Users may change their point of attachment during movements, for better service quality through a different access technology or provider, to save access costs, or for fail-over or load balancing. The base Internet Protocol (IP) does not provide such flexibility. It was mainly designed for stationary nodes with fixed IP addresses. Much effort has hence gone into mechanisms that transfer an active communication session from one IP address to another when a mobile node changes its point of attachment to the Internet. [Figure 14.1](#) illustrates such a redirection based on a communication session between a mobile node and the correspondent node that it communicates with. The mobile node in the figure hands over from its current point of attachment to a new point of attachment. At this point, the IP address changes and the correspondent node can reach the mobile node. The mobile node signals the new IP address to the correspondent node, and the correspondent node redirects the communication session to that new IP address. 199

[Figure 14.1](#) Packet redirection for a mobile node.



[Figure 14.2](#) shows the edge network topology depicted in [Figure 1.22](#) with a mobile node instead of stationary nodes. The trajectory of the mobile node passes three access points. Access points A and B belong to the same access link; they can directly communicate without an intermediate router. A switch between access points A and B is called a *link-layer handover*. It occurs unnoticed by the IP layer and normally does not force active sessions to abort. On the other hand, access points B and C belong to different links. A switch between them causes the mobile node to change its location in the Internet topology and therefore forces active sessions to abort. This switch is called an *IP-layer handover* or, for brevity, simply a *handover*. An IP-layer handover includes a link-layer handover.

[Figure 14.2](#) Edge network topology with a mobile node.



Some existing mobility mechanisms augment applications so as to survive IP address changes on either side of a connection. Others leave the applications untouched and add the necessary functionality into transport protocols that applications may use. A third approach is to build mobility support into IP directly. Changes in a mobile node's IP address are then invisible to transport protocols and applications. An example of a protocol for such IP mobility support is Mobile IP [4, 5].

Related to mobility is multihoming [6], which addresses the challenge of redirecting a packet flow between alternative IP addresses for the purpose of load balancing or fault tolerance. Mobility and multihoming are in many aspects akin. Most mobility protocols, including Mobile IP, also support multihoming [7, 8].

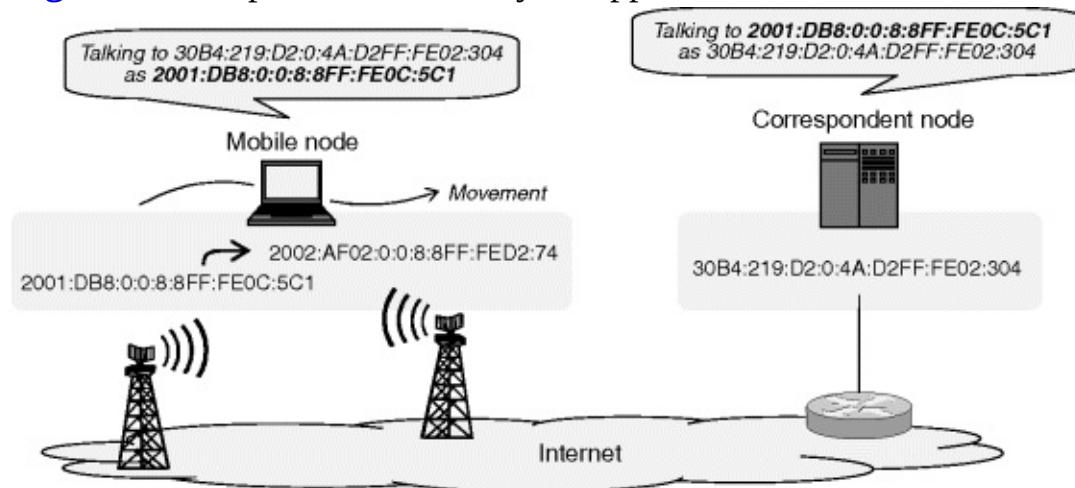
This chapter explains the different ways of augmenting the Internet for mobility support, as well as the security threats that need to be considered as part of this. The relationship between mobility and multihoming will be explored further. The chapter also gives an overview of auxiliary protocols

that play a fundamental role in IP mobility management.

14.1 Mobility Management Approaches

As Internet users are becoming increasingly mobile, and delay-and loss-sensitive real-time applications are getting more and more popular, efficient mobility support is needed. The architecture of today's Internet fails to provide such support natively. With both IPv4 and IPv6, however, when a mobile node moves, it configures a new IP address with a subnet prefix obtained of the new access link, but the new "identity" breaks active transport-layer connections and applications. [Figure 14.3](#) illustrates the adverse effect of mobility. It shows a mobile node that initiates a communication session with a correspondent node at the time it uses local IP address 2001:DB8:0:0:8:8FF:FE0C:5C1. The application and transport protocol on both nodes thus uses this IP address to identify the mobile-node side of the session. At some point, the mobile node hands over to a different point of network attachment and replaces the existing IP address with 2002:AF02:0:0:8:8FF:FED2:74. This causes the communication session to break because the application and transport protocol attempt to continue to use the mobile node's old IP address.

[Figure 14.3](#) Impact of IP mobility on applications.



Different packet redirection techniques have been proposed to solve this addressing problem at the IP layer. The techniques can be classified into host routes, tunneling, and route optimization. Common to all of the mechanisms is that they mask changes in a mobile node's IP connectivity from protocols

at layers above IP, called *upper-layer protocols* in the following. To differentiate the packets generated by upper-layer protocols from signaling packets used by a mobility protocol, the former will henceforth be referred to as *payload packet*.

14.1.1 Host Routes

One approach toward supporting mobility is to use IP addresses only as node identifiers and to abandon the function of subnet prefixes as locators [9,10]. A mobile node then has a *stable IP address* despite movements across different points of attachment. Since packets now can no longer be routed based on the subnet prefix alone, routers on the path from a correspondent node to the mobile node need to maintain one *host route* per mobile node in their routing table. An existing host route is replaced by a new one when the mobile node changes IP connectivity. Such a route update affects routers between the mobile node's new point of attachment and the correspondent node. It typically takes the form of a route update request message, which is generated either by the mobile node or by the mobile node's new access router, and which propagates in a hop-by-hop manner toward the correspondent node. Each router involved sets up a host route for the mobile node pointing toward the mobile node's new point of attachment. Where the old and the new host route merge in one router, the route update is complete because the part of the old host route between that router and the correspondent node is the same for the old and the new host route. That router then simply updates its existing host route to the new next-hop on the path toward the mobile node, and it does not propagate the route update request message further upstream. The existing state at routers that are not on the new host route eventually times out.

The participation of routers in mobility management renders host-route-based techniques little scalable and, hence, inappropriate for global use. Another issue with a global deployment of host routes is that it would require external help for correspondent nodes that wish to contact a mobile node when a host route to the mobile node is not yet in place. Host-route-based techniques were therefore proposed only for mobility management within one or a few access networks that would typically be administratively contiguous. The subnet prefix of a mobile node's stable IP address is selected such that it routes to an access network router that terminates all host routes

to the mobile node. A packet destined to the stable IP address is then routed to that access network router based on its subnet prefix, and from there it gets forwarded along a host route toward the mobile node. Routers outside the access network can thus continue to rely on subnet prefixes for forwarding. Host routes within the access network are maintained proactively, such that mobile nodes are always reachable by correspondent nodes elsewhere on the Internet. Changes in a mobile node's point of attachment are invisible to correspondent nodes.

Due to the fundamental way in which host-route-based mobility approaches change classic routing as well as the scalability issues that go along with this, the techniques were never standardized. They eventually lost attention in the research community, mostly in favor of tunneling solutions.

14.1.2 Tunneling

Tunneling provides an alternative to host routes that upholds the standard use of subnet prefixes for routing, while still providing mobile nodes with a stable IP address. The stable IP address in this case routes to a stationary proxy of the mobile node, its *global mobility anchor*. The global mobility anchor intercepts packets that correspondent nodes send to the stable IP address, encapsulates the packets, and forwards them through a tunnel to the IP address that the mobile node has temporarily configured while it stays at a particular access link. To differentiate the mobile node's configured temporary IP address from the stable IP address that locates the global mobility anchor, the former is also called the mobile node's current *on-link IP address*. The on-link IP address changes whenever the mobile node hands over to a different access link, and it is the mobile node's responsibility to inform its global mobility anchor about the new on-link IP address in such a case. The mapping between a mobile node and its current on-link IP address is called a *binding*; the process of establishing a new binding at a correspondent node or changing an existing one to a new on-link IP address is a *binding update*.

Early tunneling approaches were unidirectional from the global mobility anchor to the mobile node, and packets destined to a correspondent node were sent directly without tunneling. This is shown in [Figure 14.4](#), where the black router with the home symbol on top denotes the global mobility anchor at the mobile node's stable IP address. Due to the asymmetric routing, term

triangular routing was coined. Triangular routing has substantial disadvantages [11], however. Most importantly, the technique requires a mobile node to use its stable IP address as an IP source address for the packets it sends. This is topologically incorrect because the subnet prefix of a stable IP address in general matches none of the subnet prefixes valid on a visited access link. Packets from the mobile node are hence at increased risk of getting erased by IP source address filters [12], which many access network operators deploy to eliminate packets with IP source addresses that are—deliberately or accidentally—invalid. Other problems with triangular routing include issues with IP multicast, as well as different hop counts on the forward and reverse paths that may cause packets from a mobile node to get dropped en route. In an effort to accommodate these difficulties, *bidirectional tunneling* has replaced triangular routing. This technique differs from triangular routing in that packets sourced at a mobile node are *reversetunneled* [reversetunneling]11 to the global mobility anchor, which in turn decapsulates the packets and sends them, in a topologically correct manner, from the mobile node's stable IP address to the respective correspondent node. [Figure 14.5](#) illustrates this operation.

Figure 14.4 Triangular routing.

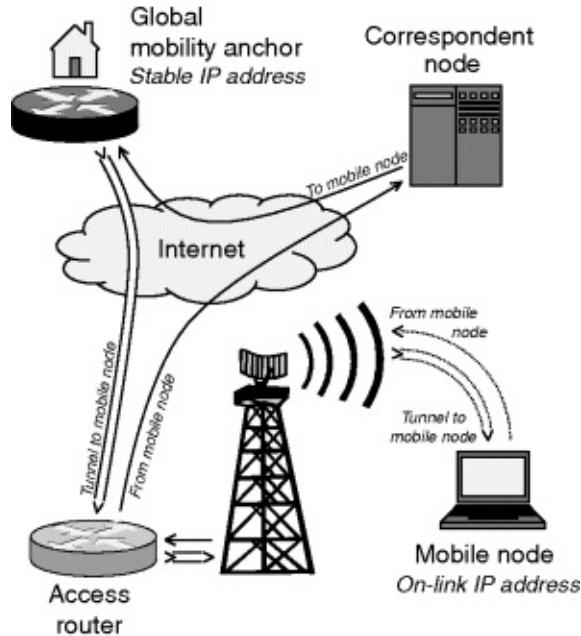
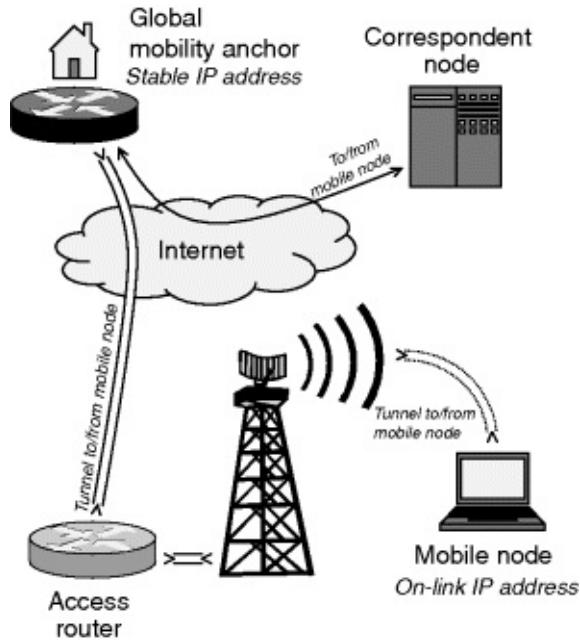


Figure 14.5 Bidirectional tunneling.



The advantage of tunneling in general is that it is compatible with the classic use of IP addresses in upper-layer protocols, so a correspondent node does not need to be mobility-aware. At the same time, the technique does not change the way in which packets are routed through the Internet. Therefore, tunneling scales well to global use across the Internet. Mobile IPv4 [4, 11] relies on triangular routing or bidirectional tunneling for this reason, and Mobile IPv6 uses bidirectional tunneling as a default. Bidirectional tunneling has further been adopted for localized mobility management [13,14]. The disadvantage of tunneling is that it causes encapsulation overhead and increased packet propagation times. This is in particular an issue for hard-real-time applications such as Internet telephony or videoconferencing, which strongly depend on timely data delivery.

14.1.3 Route Optimization

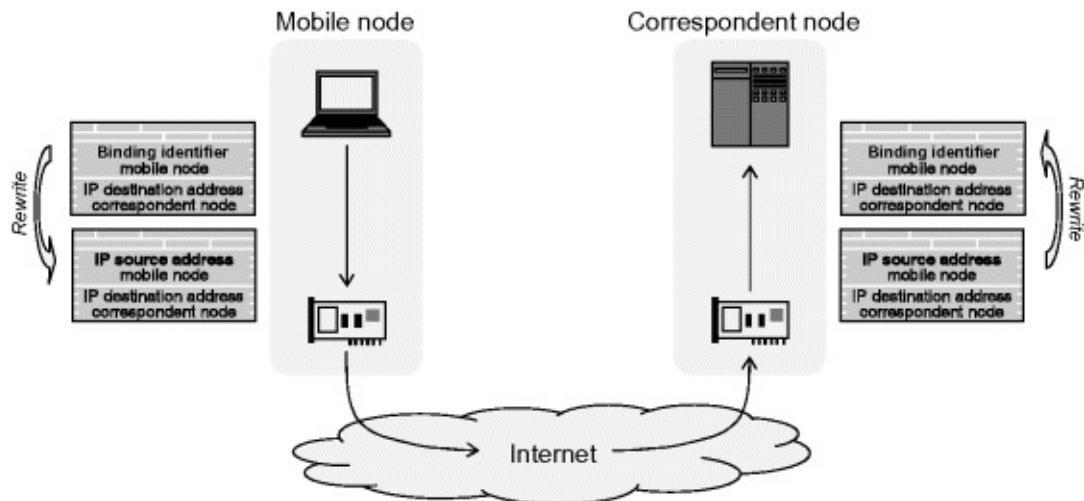
Route optimization enables mobile and correspondent nodes to exchange packets along a direct path rather than to communicate via a global mobility anchor. This minimizes propagation latencies and packet overhead, and thus accommodates the growing popularity of real-time applications with strict delay requirements. Route optimization conceptually establishes a virtual tunnel directly between a pair of communicating mobile and correspondent nodes. Both end points maintain a binding that identifies the mobile node's current on-link IP address, and the mobile node is responsible for updating

the binding at the correspondent node when its on-link IP address changes during a handover.

Upper-layer protocols may again identify the mobile node based on a stable IP address. This approach is followed by Mobile IPv6, where the stable IP address locates a roaming mobile node's global mobility anchor and can be used for both bidirectional tunneling or route optimization, depending on the availability of mobility support on the correspondent node. Route optimization per se does not require a global mobility anchor, so the mobile node may more generally be identified by a routable or nonroutable *binding identifier*. The mobility extensions [7] to the *Host Identity Protocol* [15] are an example of a mobility protocol that uses pure route optimization. The binding identifier is in this case nonroutable. It is cryptographically generated and bound to a public/private key pair of the mobile node, enabling the mobile node to authenticate securely to any correspondent node. The Host Identity Protocol requires cryptographic protection of all payload packets through IP Security (IPsec).

Route optimization saves the additional overhead of packet encapsulation by rewriting the IP source and destination addresses in payload packets exchanged between mobile and correspondent nodes. [Figure 14.6](#) illustrates this procedure for packets that the mobile node sends to the correspondent node. Upper-layer protocols on the mobile node use the binding identifier as an IP source address for outgoing packets, and the mobile node replaces this with its current on-link IP address when a packet reaches the IP layer. Packets received from the correspondent node include the current on-link IP address as the IP destination address. The mobile node overwrites this with its binding identifier before it hands the packet on to upper-layer protocols. On the correspondent node side, outgoing packets use the mobile node's binding identifier as an IP destination address when generated by upper-layer protocols, and the correspondent node overwrites this with the mobile node's current on-link IP address as the packets traverse its IP layer. The correspondent node further substitutes the mobile node's binding identifier for the on-link IP address when processing a packet received from the mobile node at the IP layer.

[Figure 14.6](#) IP address rewriting in route optimization.



Mobility protocols with support for route optimization further include the binding identifier—or information sufficient to deduce the binding identifier—in route-optimized payload packets before sending the packets through the network. This enables both mobile and correspondent nodes to indicate whether an on-link IP address used in a payload packet is supposed to be replaced by the binding identifier. Payload packets that use the on-link IP address, but do not include the binding identifier or equivalent information, are exempt from mobility-specific processing at the IP layer. For example, route-optimized payload packets in Mobile IPv6 include the mobile node's stable IP address as a binding identifier within IPv6 extension headers. The mobility extensions for the Host Identity Protocol identify a binding based on a security parameter index of the mandatory IPsec security association between the end nodes. This, too, is contained in every payload packet exchanged.

Route optimization is a pure end-to-end mechanism and as such is not suited for localized mobility management. Its appealing property of reduced propagation latencies, however, comes at the cost of more intractable security challenges. Mobile and correspondent nodes usually do not know each other *a priori* and do not preshare credentials based on which mobility management could be secured. Classic means for authentication are hence not applicable. Tunneling approaches are typically easier to secure because a mobile node and its global mobility anchor are supposed to be from the same administrative domain and therefore can be more practically preconfigured with the credentials required for mutual authentication. This and other security issues related to mobility in general and route optimization in particular are discussed in 14.2 Section. Another disadvantage of route

optimization is that it requires mobility support on the correspondent node side and so may not always be applicable. Correspondent node support may eventually become ubiquitous, however, given that it is a recommended feature for all IPv6 nodes as per the IPv6 Node Requirements RFC [16].

To enable correspondent nodes to contact a roaming mobile node, route optimization is typically supplemented with tunneling or some other mechanism that provides a stable IP address. Mobile IPv6, for instance, incorporates route optimization, but mobile nodes still continue to be reachable via a stable IP address and bidirectional tunneling. Route optimization is also used in the mobility extensions of the Host Identity Protocol, where the binding identifier is cryptographic and nonroutable. Mobile nodes can here be reached by help of a stationary *rendezvous server*. This maintains a binding between a mobile node's binding identifier and current on-link IP address, and it can so serve as an indirection mechanism when a correspondent node contacts a mobile node.

14.2 Security Threats Related to IP Mobility

Any approach to support mobility at the IP layer modifies the traditional way in which packets are routed. This may invalidate assumptions that upper-layer protocols have on IP and thus lead to vulnerabilities to attacks that exploit the assumptions. These threats shape the security design of mobility protocols, and they limit the solution space for mobility protocol optimizations. The following is an overview of security threats for route optimization. Similar threats apply when mobility is realized through tunneling or host routes. However, the threats are typically harder to mitigate for route optimization since an a-priori security or trust relationship between an arbitrary pair of mobile and correspondent nodes in general does not exist. An exhaustive synopsis of mobility-related security threats is given in [17] in the context of Mobile IPv6.

14.2.1 Impersonation

A classic assumption of upper-layer protocols is that, when a peer is known to “own” a particular IP address, packets sent to that IP address will eventually reach the peer or they get lost in the network. This assumption is based on the generally reliable path selection in the Internet's routing

infrastructure for a given packet's IP destination address. If an attacker was to change the way packets are routed, it would at least have to compromise a router on the legitimate path of the packets so as to divert the packets from there. On the other hand, IP mobility protocols establish a level of indirection between a mobile node's binding identifier at upper-layer protocols and the on-link IP address that locates the mobile node in the network. By way of mapping a binding identifier onto an on-link IP address, they thus introduce a mechanism through which the routing of packets can be changed at the edge of the Internet. Therefore, for a packet to eventually reach the intended destination, not only needs the routing infrastructure to be reliable, but also must the respective binding be legitimate.

A mobility protocol would allow an attacker to circumvent the reliable path selection of the Internet's routing infrastructure and launch an *impersonation attack* against a victim if it enabled the attacker to establish a binding at a correspondent node on behalf of the victim. In a typical impersonation attack, the attacker makes upper-layer protocols on the correspondent node side believe that they communicate with the victim, although they in fact communicate with the attacker. This attack is illustrated in [Figures 14.7](#) and [14.8](#). [Figure 14.7](#) shows the original packet flow between the victim's IP address, IP_V , and the IP address of a correspondent node, IP_{CN} . At some time, the attacker sends the correspondent node a binding update, tricking it into associating the victim's binding identifier, ID_V , with the attacker's on-link IP address, IP_A . Packets for the victim, which the correspondent node would normally send to IP_V , are now redirected to IP_A , as shown in [Figure 14.8](#). Likewise, the correspondent node's IP layer modifies any packets received from IP_A such that they appear to originate from ID_V at upper-layer protocols. The victim may still send packets to the correspondent node, but any response from the correspondent node is directed to the attacker.

[**Figure 14.7**](#) Original packet flow.

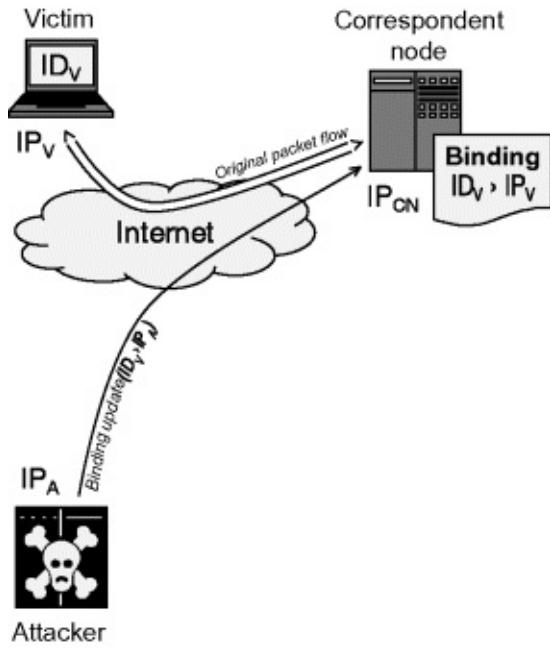
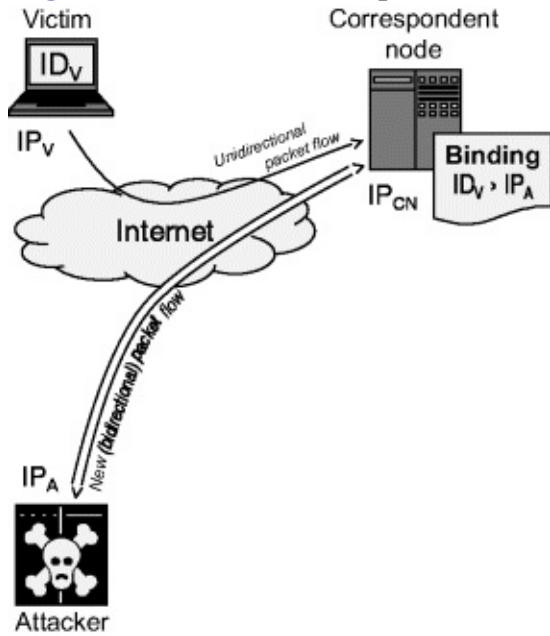


Figure 14.8 Redirected packet flow.



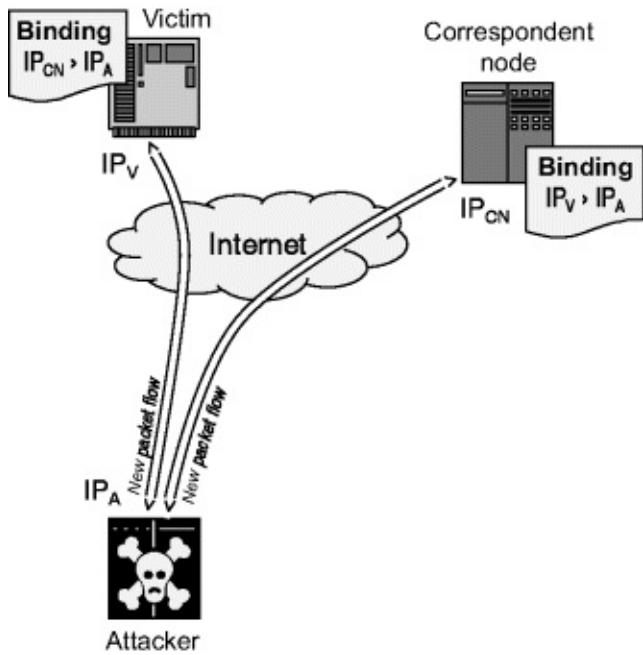
The victim shown in Figures 14.7 and 14.8 is a mobile node, so it already has a binding identifier that the attacker can misuse. In general, the same type of attack is also possible against stationary victims, which do not participate in a mobility protocol. The attacker must then use its victim's on-link IP address as a binding identifier when spoofing a binding update for the correspondent node. This is particularly easy if the mobility protocol uses stable IP addresses as binding identifiers, such as Mobile IPv6, because

binding identifiers and IP addresses are then from the same name space. The correspondent node would accept such a binding update even if it affects a stationary peer because it generally does not know whether a peer is mobile or stationary. Mobility protocols that do not allow regular IP addresses to be adopted as binding identifiers cannot be misused to impersonate stationary victims. The mobility extensions to the Host Identity Protocol belong to this class.

The issue with impersonation is actually not new to the Internet. An attacker on the communication path between two nodes may already be able to eavesdrop on packets exchanged by the nodes, intercept the packets, and/or modify and eventually forward them. Mobility aggravates the problem in that an insecure mobility protocol may allow even an attacker *off* the communication path to do this, as it happens in [Figures 14.7](#) and [14.8](#). An attacker may also install a false binding before the victim starts using the attacked IP address.

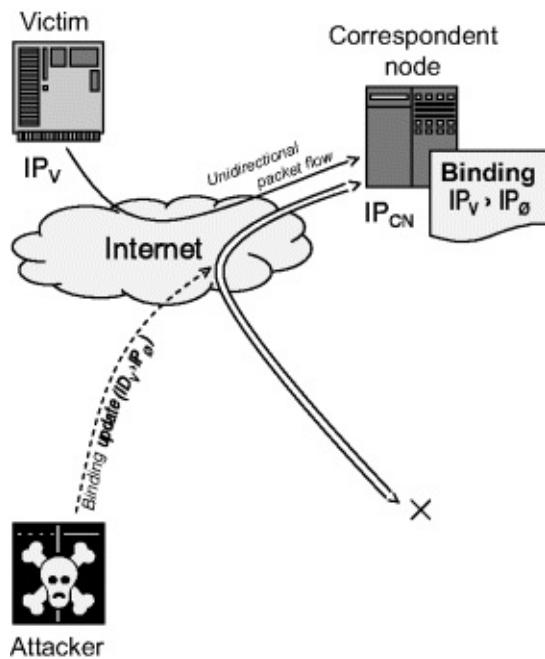
The impersonation attack, as shown in [Figures 14.7](#) and [14.8](#) can be extended into a true man-in-the-middle attack, where the attacker performs a binding update both with the victim and with the correspondent node. (The roles of the victim and the correspondent node are then actually interchangeable.) [Figure 14.9](#) illustrates this attack for the case where both the victim and the correspondent node are stationary. The attacker here uses the correspondent node's IP address, IP_{CN} , as a binding identifier in the binding update for the victim, and it uses the victim's IP address, IP_V , as a binding identifier in the binding update for the correspondent node. The on-link IP address in both binding updates is the attacker's own IP address, IP_A . Packets that the victim and the correspondent node send are now all routed to the attacker. The attacker can read the packets and possibly forward them to the original recipient, either modified or as is. The two binding updates are not shown in the figure.

[**Figure 14.9**](#) Man-in-the-middle attack.



An impersonation attack can further be used for denial of service. The purpose of a *denial-of-service attack* in general is to compromise a victim in terms of its ability to communicate and to respond to requests from legitimate peers. When this is accomplished through an impersonation attack, packets that are destined to the victim by some correspondent node are redirected to a random or nonexisting IP address so that bidirectional communications become impossible between the victim and the correspondent node. [Figure 14.10](#) illustrates this attack, again for the case where both the victim and the correspondent node are stationary.

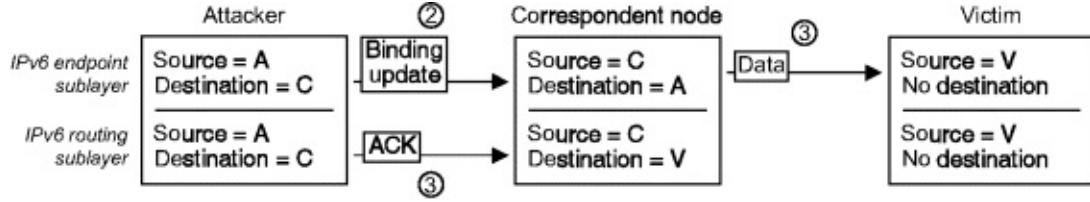
Figure 14.10 Denial-of-service attack.



14.2.2 Redirection-Based Flooding

Appropriate authentication of a mobile node's binding identifier can mitigate impersonation attacks. But a malicious, yet properly authenticated node may still register a false on-link IP address, and so redirect packets that would otherwise be delivered to the malicious node itself to a third party. This introduces a vulnerability to a new type of denial-of-service attack, redirection-based flooding attacks. A *flooding attack* in general defeats a victim's ability to communicate by overloading the victim with a high number of needless packets. In the specific case of a *redirection-based flooding attack*, the flooding packets originate with one or multiple correspondent nodes of the attacker. The attacker is supposed to receive the packets itself, but it tricks the correspondent nodes, through misuse of a mobility protocol, into substituting the victim's on-link IP address for the binding identifier of the attacker. The packets thus get redirected to the victim. [Figure 14.11](#) illustrates a redirection-based flooding attack where the attacker establishes TCP connections with two correspondent nodes for downloading some large data files, and then misuses the mobility protocol to make all correspondent nodes redirect outgoing packets to the IP address of the victim.

[Figure 14.11](#) Flooding attack.



A redirection-based flooding attack may target an entire network, rather than a single node, either by loading the network with packets, or by overwhelming a router or other critical network device further upstream. The attacker then directs the flooding packets against an arbitrary IP address matching a subnet prefix of the victim network or, respectively, against the IP address of the network device in focus. An attack against a network potentially impacts a larger number of nodes than an attack against a specific node, although neighbors of a victim node on a broadcast link typically suffer the same damage as the victim itself.

14.2.3 Possible Solutions

Impersonation attacks can be prevented by authenticating a mobile node to the correspondent node during a binding update, and at the same time verifying that the mobile node is in fact authorized to add and remove an on-link IP address for the claimed binding identifier. One way to authenticate the mobile node is cryptographically based on secret credentials, preconfigured on both nodes [18]. Pair-wise preconfiguration is relatively straightforward, but it is inappropriate for global use since mobile and correspondent nodes may communicate without prior contact. The technique also causes significant administrative overhead.

Authentication based on asymmetric public-key cryptography does not require pair-wise preconfiguration and hence involves substantially less administrative labor. Here, the correspondent node tests the mobile node's knowledge of the private component of a public/private key pair given only the public key. The mapping between the mobile node's binding identifier and its public key, in turn, is attested by a *certificate*, issued by a *public key infrastructure* that both end nodes trust. However, large-scale use of a public key infrastructure for global mobility management is considered unsuitable for multiple reasons [19], foremost due to scalability and performance concerns.

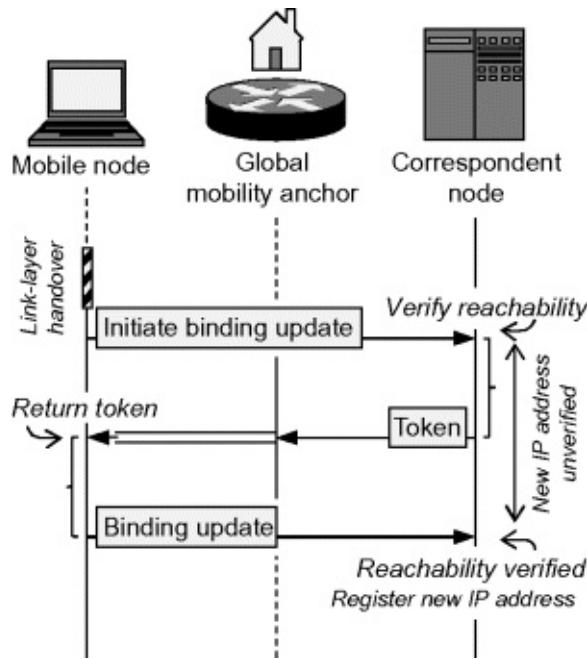
A third technique to protect against impersonation attacks is to tie a mobile node's binding identifier cryptographically to the public component of the

mobile node's public/private key pair, for example, by taking bits for the binding identifier from the output of a hash on the public key [20]. The mapping between the binding identifier and the public key is then verifiable without a public key infrastructure. This technique is used in enhanced route optimization for Mobile IPv6 [cba]21 as well as in the mobility extensions of the Host Identity Protocol.

Mobility protocols that use a stable IP address as a binding identifier, such as Mobile IPv6, can further verify the mobile node's reachability at the stable IP address in order to validate that the mobile node really owns the binding identifier. *Reachability verification* in general is realized through the exchange of an unpredictable piece of information, which the correspondent node sends to the to-be-verified IP address, and which the node claiming to be reachable at that IP address echoes back. The new on-link IP address is called *unverified* until the correspondent node has received the token back from the mobile node, and it is called *verified* afterwards.

[Figure 14.12](#) illustrates how reachability verification can be used for authenticating a mobile node. After the link-layer handover has completed, the mobile node initiates the binding update by sending the correspondent node a message. To be able to authenticate the mobile node, the correspondent node sends an unpredictable token to the mobile node's stable IP address. The global mobility anchor of the mobile node receives this token and tunnels it to the mobile node. The mobile node can then use the token to send an authenticated message requesting a binding update. It should be noted that the message exchange depicted in [Figure 14.12](#) is conceptual and abstracts from some of the details of a typical mobility protocol.

[Figure 14.12](#) Preventing impersonation.



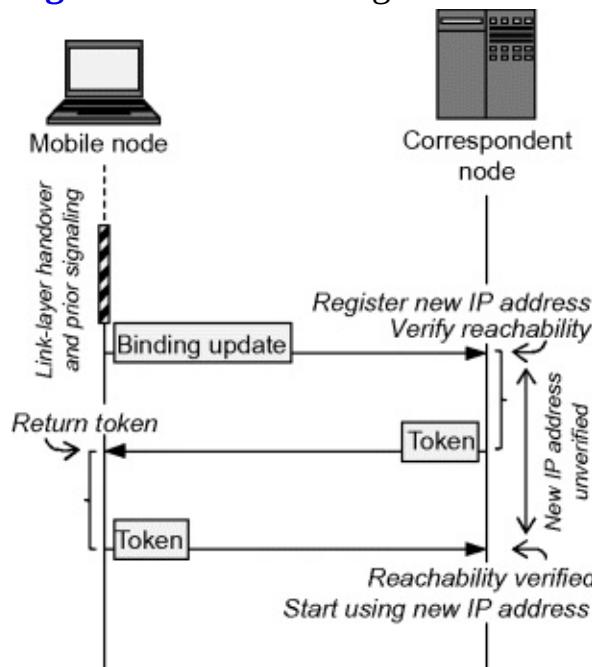
Reachability-based authentication is less secure than cryptographic authentication because it does not prevent impersonation attacks initiated from a position on the path from the correspondent node to the stable IP address. An attacker in such a position could always pass reachability verification for the stable IP address, so it could use the IP address as a binding identifier in a spoofed binding update for the correspondent node. Reachability verification further induces potentially long handover delays due to the end-to-end message exchange. Yet, the appealing property of reachability-based authentication is that it works without preconfiguration of shared secret credentials and also without a public-key infrastructure.

Redirection-based flooding attacks, too, can be protected against by reachability verification. In this case, it is the mobile node's on-link IP address that is subject to verification rather than the stable IP address. The way reachability verification is used for this purpose by existing mobility protocols is that no payload packets are sent to the on-link IP address until it becomes verified. This conservative form is henceforth referred to as *standard reachability verification*. Both Mobile IPv6 and the mobility extensions for the Host Identity Protocol apply it.

[Figure 14.13](#) illustrates standard reachability verification conceptually. At some point after the link-layer handover is complete, the mobile node sends the correspondent node a message to initiate the binding update. The message that is shown in the figure is assumed to already be authenticated, hence

some signaling might precede the message to facilitate the authentication, such as the message exchange shown in [Figure 14.12](#). When the correspondent node processes the message, it registers the mobile node's new on-link IP address and labels it unverified for the time being. The correspondent node then sends an unpredictable token to the new on-link IP address, but it refrains from sending any payload packets yet. The mobile node returns the token to the correspondent node once it receives it. The correspondent node, in turn, considers the new on-link IP address legitimate when it receives the token and relabels the IP address verified. At that point, the correspondent node begins sending payload packets to the new on-link IP address. The mobile node can send payload packets from the new on-link IP address right after initiating the binding update; it does not need to wait for the reachability verification to complete.

Figure 14.13 Preventing redirection-based flooding.



The aforementioned vulnerability of reachability verification to attackers on the path from a correspondent node to the IP address being verified is acceptable in the case of flooding attack prevention for three reasons:

1. An attacker on the path toward a victim can trick a correspondent node into sending packets to the victim already in the nonmobile Internet of today.
 2. The set of correspondent nodes that an on-path attacker can potentially induce to send packets to the victim is confined to upstream nodes.

3. The attraction of an on-path attack is limited given that the flooding packets also pass the attacker.

Some transport protocols already pursue reachability verification during connection establishment. A TCP responder, for example, chooses a random 32-bit sequence number and sends it to the initiator's claimed IP address. The initiator must send the sequence number back to the responder, allowing the latter to verify the former's reachability.¹ IP mobility protocols defeat the purpose of such transport-layer reachability verification because they introduce the ability to redirect packets after connection establishment. Therefore, an additional reachability verification must be pursued at the IP layer whenever a mobile node changes its on-link IP address.

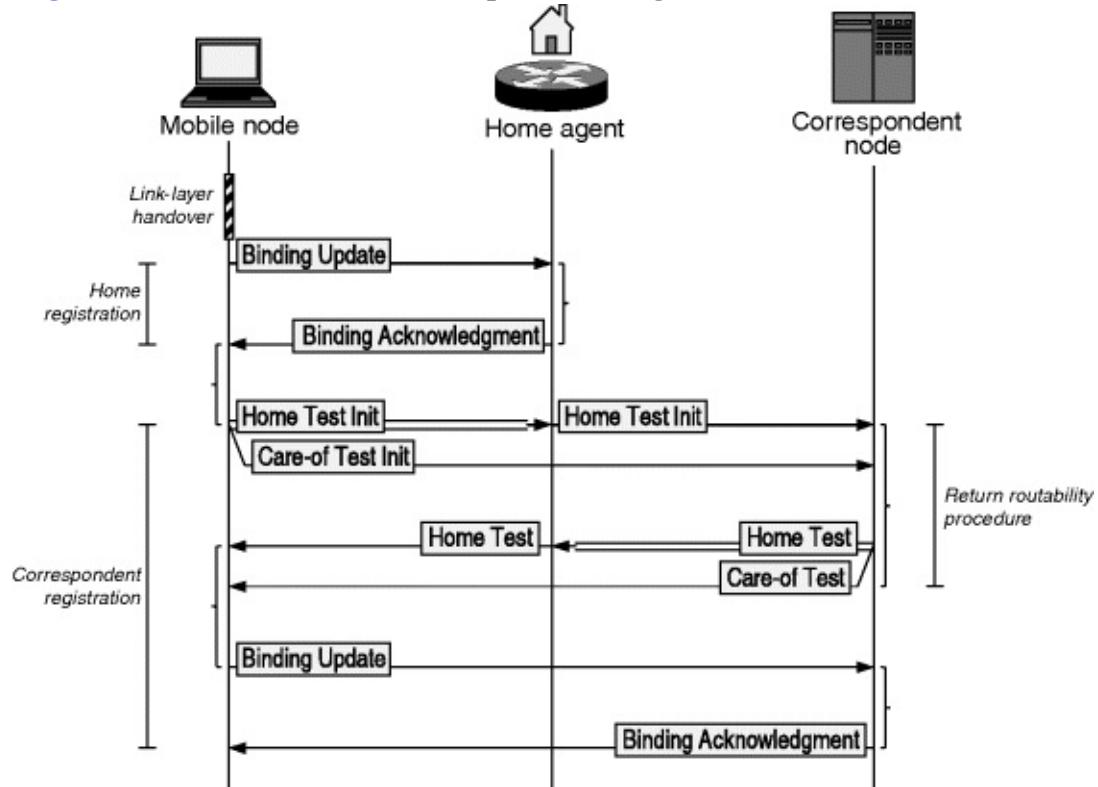
14.3 Mobility Support in IPv6

Mobility support in IPv6, *Mobile IPv6* [5], is an IP-layer mobility protocol providing permanent reachability and session continuity to mobile nodes. Mobile IPv6 uses bidirectional tunneling in combination with a stable IP address as a default packet redirection technique. It further offers route optimization for reduced packet propagation delays between a mobile node and a correspondent node that supports the optimization. A mobile node's stable IP address, called a *home address* in Mobile IPv6 terminology, has a subnet prefix from the mobile node's *home link*. The mobile node can directly communicate via its home address when it attaches to the home link. It operates like a stationary IPv6 node without mobility support in this case.

When the mobile node moves to a foreign access link, it becomes reachable at an on-link IP address, or *care-of address*, that differs from the home address. The mobile node then requests its global mobility anchor, a dedicated router on the home link called a *home agent*, to proxy the mobile node at the home address and provide a bidirectional tunneling service. The mobile node initiates this *home registration* with a Binding Update message that it sends to the home agent, as shown in [Figure 14.14](#). The home agent creates a binding between the home address and the care-of address when it receives the Binding Update message. The home agent maintains a *binding cache* in which it stores information related to the binding, such as the home address, the care-of address, and the binding lifetime. The binding cache holds a separate entry for each registered binding. The home agent further

sets up a bidirectional tunnel to the care-of address, through which the mobile node can continue to remotely communicate via its home address. The home agent finally assigns the binding a maximum lifetime and confirms the home registration with a Binding Acknowledgment message that includes the binding lifetime. The mobile node must renew the home registration when the binding lifetime elapses. The Mobile IPv6 RFC does not limit the binding lifetime for home registrations. The mobile node establishes a tunnel from the care-of address to the home agent as part of the home registration. Information related to the home registration is stored in a new entry of the mobile node's *binding update list*. This includes the home and care-of addresses, the binding lifetime granted by the home agent, and any state related to pending retransmissions and re-registrations.

Figure 14.14 Home and correspondent registration in Mobile IPv6.



The Binding Update and Binding Acknowledgment messages exchanged during a home registration are authenticated and integrity protected based on an IPsec security association, typically using the IPv6 Encapsulating Security Payload extension headers in transport mode. The credentials required to bootstrap the security associations may be preconfigured onto the mobile node and the home agent. The nodes are assumed to be administered by the

same authority, so the preconfiguration should in general be feasible. Technically, IPsec protection alone cannot prevent redirection-based flooding attacks against a spoofed care-of address since it does not incorporate a verification of the mobile node's reachability at the care-of address. This makes home registrations in general misusable for flooding attacks. The security design of the home registration is instead founded on an assumed trust relationship between the mobile node and the correspondent node. This permits the home agent to accept a care-of address from the mobile node without previous reachability verification.

The mobile node retransmits any messages that it sends to the home agent or a correspondent node and for which it does not receive a response within appropriate time. The standard retransmission algorithm calls for the mobile node to wait 1 second before it initiates the first retransmission,² and to double the waiting time for each further retransmission. A maximum interval of 32 seconds is reached after six retransmissions. The mobile node may then continue retransmitting at the same rate.

It is the mobile node's responsibility to update a binding at the home agent whenever its care-of address changes due to a handover. The bidirectional tunnel is then moved to the new care-of address. Each such binding update requires another exchange of IPsec-protected Binding Update and Binding Acknowledgment messages. When the mobile node returns to its home link after a period of roaming, it initiates a *home deregistration*, requesting the home agent to discontinue proxy service, remove any existing binding for the mobile node, and tear down the bidirectional tunnel to the mobile node.

The increased packet propagation latencies that go along with bidirectional tunneling can be avoided if a correspondent node supports Mobile IPv6 route optimization. The mobile node initiates this through a *correspondent registration* with the correspondent node when it receives an encapsulated payload packet that originates from this correspondent node, and no recent indication exists that the correspondent node does not support route optimization. The purpose of the initial correspondent registration is to probe whether the correspondent node supports route optimization and, in case it does, have the correspondent node cache a binding between the mobile node's home and current care-of address. [Figure 14.14](#) depicts the messages exchanged during a correspondent registration. As with the home registration, these include a Binding Update message and, in this case

optionally, a Binding Acknowledgment message. The registrations differ in the way they secure this exchange, however. There is generally no a-priori security or trust relationship between an arbitrary pair of mobile and correspondent nodes based on which a correspondent registration could be protected. So to mitigate the threats described in Section 14.2, Mobile IPv6 uses a *return routability procedure*. This is based on a verification of the mobile node's reachability at the home address and the care-of address. Reachability at both IP addresses is taken as an indication that the mobile node is the legitimate owner of these two IP addresses, and hence that it is secure to bind the IP addresses. The *home address test* weakly authenticates the mobile node because the home address identifies the mobile node at upper-layer protocols. The *care-of address test* validates the mobile node's liveliness at the claimed point of attachment, so it protects against redirection-based flooding attacks.

The mobile node initiates the home address test with a Home Test Init message, which it reversetunnels to the home agent, and which the home agent forwards from the home address to the correspondent node. The confidentiality of the Home Test Init message is protected inside the tunnel, typically using the IPv6 Encapsulating Security Payload extension header. But there is no cryptographic protection on the path between the home agent and the correspondent node. To provide some minimum level of security also on the latter path, the Home Test Init message includes a random *home init cookie* to be returned by the correspondent node. A returned cookie proves that the message was processed by a node on the path from the home agent to the correspondent node, thus protecting against off-path attackers that cannot see the Home Test Init message. This mechanism is ineffective against eavesdroppers on the path from the home agent to the correspondent node. Yet, the motivation here is that such attackers could compromise communications between the mobile node and the correspondent node unless the packets are protected otherwise.

The correspondent node responds to the Home Test Init message with a Home Test message. This is directed to the mobile node's home address and hence again routes via home agent. The Home Test message contains a home keygen token, a home nonce index, and the home init cookie copied from the Home Test Init message. The mobile node considers the returned home init cookie as sufficient proof that the Home Test message was generated by the right correspondent node. The *home keygen token* is a keyed one-way hash

on the concatenation of the mobile node's home address, a random *nonce*, and a “0” indicating that the token corresponds to a home address rather than a care-of address. The hash is keyed with a secret known only to the correspondent node. The mobile node uses the home keygen token to prove its reachability at the home address when it eventually sends a Binding Update message to the correspondent node. The *home nonce index* identifies the nonce based on which the correspondent node has computed the home keygen token. The mobile node includes the home nonce index in the Binding Update message to allow the correspondent node to reproduce the home keygen token without having to store it explicitly. This kind of resource preservation is a precaution against denial-of-service attacks that aim at exhausting the correspondent node's memory resources. The Home Test message is sent through the same authenticated and encrypted tunnel between the home agent and the mobile node as the Home Test Init message.

The care-of address test consists of a pair of messages directly exchanged between the mobile node and the correspondent node, and it does not involve the mobile node's home agent. The mobile node initiates the care-of address test by sending a Care-of Test Init message to the correspondent node, as shown in [Figure 14.14](#). This typically happens in parallel with the initiation of the home address test. The Care-of Test Init message includes a random *care-of init cookie* that is to be returned by the correspondent node. This serves as proof that the message was processed by a node on the path from the mobile node to the correspondent node.

The correspondent node sends a Care-of Test message directly to the mobile node's care-of address when it receives the Care-of Test Init message. The Care-of Test message contains a care-of keygen token, a care-of nonce index, and the care-of init cookie copied from the Care-of Test Init message. The *care-of keygen token* is a keyed one-way hash on the concatenation of the mobile node's care-of address, a random nonce, and a “1” indicating that the token corresponds to a care-of address. The hash is again keyed with the correspondent node's secret. The mobile node uses the care-of keygen token to prove its reachability at the care-of address. The *care-of nonce index* identifies the nonce based on which the correspondent node has computed the care-of keygen token. As with the home nonce index, this is communicated in the protocol so that the correspondent node can reproduce the care-of keygen token without having to explicitly store it.

The return routability procedure enables the mobile node to transact a

binding update at the correspondent node. Specifically, the mobile node computes a *binding management key* as a one-way hash on the concatenation of the home keygen token and the care-of keygen token. Knowledge of the right binding management key hence proves the mobile node's reachability at the home address and at the care-of address. Based on the binding management key, the mobile node calculates a message authentication code for a Binding Update message that it subsequently sends to the correspondent node. The Binding Update message further includes the home and the care-of nonce indices received during the return routability procedure. The mobile node may request the correspondent node to return a Binding Acknowledgment message by raising a flag in the Binding Update message.

The home and care-of nonce indices included in the Binding Update message allow the correspondent node to reproduce the home and care-of keygen tokens, respectively, on receipt of the Binding Update message. The correspondent node can then calculate the binding management key in the same way as the mobile node, and use this to verify the message authentication code of the Binding Update message. If the message authentication code is correct, the correspondent node knows that the mobile node has received the Home Test and Care-of Test messages and, consequently, that the mobile node is reachable at the claimed home and care-of addresses. A correct message authentication code also indicates that the message has not been tampered with in flight. The correspondent node can hence assume with reasonable certainty that the mobile node is the legitimate owner of the home and care-of addresses included in the Binding Update message, and that it is secure to bind the two IP addresses. The correspondent node accordingly creates a new entry in its binding cache and begins route-optimizing payload packets. The binding cache of a correspondent node is fundamentally the same as the binding cache of a home agent. Finally, if requested by the mobile node, the correspondent node confirms the successful correspondent registration with a Binding Acknowledgment message. This message also includes the binding lifetime granted by the correspondent node, and it is again authenticated with the binding management key. The Mobile IPv6 RFC limits the binding lifetime for correspondent registrations to seven minutes in an attempt to ward off time-shift attacks [17]. A mobile node that does not change IP connectivity for this period must refresh any active correspondent registrations. This requires another run of the return routability procedure. The mobile node

stores information related to a correspondent registration in its binding update list.

A correspondent node that does not support Mobile IPv6 route optimization sends an ICMPv6 Parameter Problem message when it receives a Home Test Init, Care-of Test Init, or Binding Update message, indicating that it does not understand the payload in the received packets. The ICMPv6 Parameter Problem messages enable the mobile node to determine the lack of support on the correspondent node side. The mobile node caches such information in its binding update list to avoid repeated attempts to initiate route optimization with correspondent nodes that do not support it.

The mobile node updates the binding at each of its correspondent nodes when its care-of address changes during a handover. This involves another care-of address test. The home keygen token from the previous home address test may be reused if it has been obtained within the last 3.5 minutes. Otherwise, the token has passed its lifetime and the entire return routability procedure must be redone. The lifetime of a care-of keygen token is 3.5 minutes as well. But since a care-of keygen token is bound to the care-of address to which it was sent, it typically cannot be reused. The mobile node may reuse a previously obtained care-of keygen token only in the special case where it returns to a previously visited access link and configures an old care-of address again.

The mobile node initiates a *correspondent deregistration* when it eventually returns to its home link. The correspondent node then removes the existing binding for the mobile node, and subsequent payload packets are exchanged via the home address. The binding management key needed to authenticate the Binding Update and Binding Acknowledgment messages exchanged during the correspondent deregistration is a one-way hash on only a home keygen token. The return routability procedure hence reduces to a home address test in the case of a correspondent deregistration.

The Mobile IPv6 specification restricts the scheduling of home and correspondent registrations. Accordingly, the mobile node must send a Binding Update message to its home agent before it initiates the return routability procedure with a correspondent node. The mobile node must also wait for a positive Binding Acknowledgment message from the home agent before it sends a Binding Update message to a correspondent node. The same message order applies to home and correspondent deregistrations. The purpose of these rules is to establish the mobile node's ability to

communicate via the home address before a binding for the home address can be cached at correspondent nodes.³

[Figure 14.14](#) illustrates a scenario where the mobile node communicates with only one correspondent node that supports Mobile IPv6 route optimization. The figure is therefore limited to a single correspondent registration. More generally, the mobile node may communicate with multiple correspondent nodes that support the optimization. It then pursues a correspondent registration with each of the correspondent nodes in parallel.

14.4 Reactive Versus Proactive Mobility Support

Mobility support at the IP layer has traditionally been considered a response to link-layer handover. Sophisticated protocol optimizations for such *reactive mobility management* can reduce handover delays considerably, but a minimum latency for payload packets to be redirected to the mobile node's new point of attachment is inherent in all reactive approaches: It always takes a one-way propagation time to register a new on-link IP address with a global or local mobility anchor, or with a correspondent node, plus another one-way propagation time for the first redirected payload packet to arrive at the new IP address. At the same time, packets in flight toward the old on-link IP address are lost.

Proactive mobility management can further reduce handover delays and handover-related packet loss. It requires a mobile node to anticipate changes in IP connectivity and, if a handover is imminent, obtain a new IP address from the target access link and register this with its global or local mobility anchor, or with its correspondent node, prior to initiating the link-layer handover. While efficient reactive mobility management cannot go without link-layer notifications that inform the IP layer about changes in link-layer attachment, proactive mobility management requires bidirectional interaction between the link and IP layers. The mobility protocol must now be able to issue commands to the link layer and receive events as well as anticipatory information from the link layer. The mobile node must further be able to match link-layer information from a discovered access link to subnet prefix information for that link. This typically requires a mapping between the link-layer address of a discovered access point and the set of subnet prefixes in

use on the link to which this access point connects.

The requirements of reactive and proactive mobility management in terms of cross-layer interaction and network information retrieval can be satisfied with Media-Independent Handover Services [23], a standard developed by the IEEE 802.21 working group. Reference 24 specifies an alternative interface between the link layer and the IP layer, and other approaches to proactive subnet prefix discovery are described in [25]. Section 14.6.7 describes important parts of Media-Independent Handover Services.

14.5 Relation to Multihoming

While mobility concerns redirections to previously unknown IP addresses, multihoming [6] describes a node's ability to redirect packets between multiple IP addresses that it has configured simultaneously. This allows the node to split traffic across multiple connections to the Internet for increased quality of service, or to provide for rapid fail-over in the event one of the connections goes down. One differentiates between *node multihoming*, where the IP addresses are configured on different network interfaces of the same node, and *access network multihoming*, where the IP addresses have subnet prefixes that belong to the same access network, but have been allocated from different Internet service providers.⁴ Although the policies for IP address selection and switching typically differ between node and access network multihoming, from the perspective of this book, these differences are negligible. The techniques will therefore be more generally referred to as multihoming henceforth.

Given the strong technical relationship between mobility and multihoming, both techniques can be combined in one protocol. In fact, the mobility extensions for the Host Identity Protocol also support multihoming, and the NOMADv6 extensions [26] for Mobile IPv6 enable a mobile node to specify a care-of address per packet flow. These extensions have been developed with a focus on node multihoming, but principally, access network multihoming could be supported as well.

A side effect of the strong relationship between mobility and multihoming is that the threats described in Section 14.2 also apply to multihoming. The specific threat of redirection-based flooding attacks is even more significant in the case of multihoming because the attacker could send bogus transport-

layer acknowledgments without spoofing their IP source addresses. The attacker would register both its own and a victim's on-link IP addresses, and then send spoofed acknowledgments from its own IP address while redirecting the correspondent node's packets to the IP address of the victim. The binding at the correspondent node would then ensure that the acknowledgments are attributed to the packet flow toward the victim. An ability to avoid spoofing its packets' IP source address, also called *IP spoofing* [27], is attractive from an attacker's perspective because routers may verify the topologic correctness of the IP source addresses in to-be-forwarded packets and drop packets where the IP source address is incorrect. A mobility protocol without multihoming functionality permits only a single on-link IP address per binding and hence does not allow the attacker to use any other IP source address for the acknowledgments than the registered IP address of the victim.

14.6 Protocols Supplementing Mobility

Gaining IP connectivity on a new access link requires a node to discover its neighborhood and configure a new IP address. Many of the protocols that need to be executed for these tasks pertain to stationary nodes just as well as to mobile nodes. But since mobile nodes are required to gain IP connectivity more frequently and oftentimes with active communication sessions, it is important that this happens efficiently. This section explains, for IPv6, what steps a node needs to take in configuring IP connectivity on a new access link, and it introduces protocols that are used for this purpose. The section also considers optimized protocols that have been designed to meet the increased efficiency demands of mobile nodes, and it finally explains how proactive mobility management can be realized.

14.6.1 Router and Subnet Prefix Discovery

A node that attaches to a new access link learns about local access routers and subnet prefixes during *router discovery* and *subnet prefix discovery*, respectively. The default mechanism for this is defined as part of *Neighbor Discovery* [28]. This protocol calls for access routers to multicast Router Advertisement messages onto a local access link on a loosely periodic basis. A node may listen for advertisements or, if it is unwilling to wait, actively

request one by sending a Router Solicitation message. A Router Advertisement message identifies the originating access router and contains a set of subnet prefixes that belong to the local link. The node chooses one of the advertising access routers as a first hop for all packets it sends to destinations off the local link. Neighboring nodes are identified by an IP address that matches one of the subnet prefixes advertised on the local link. Packets destined to a neighbor can be transmitted directly to that neighbor without the help of an access router.

14.6.2 Movement Detection

Mobile nodes change their point of attachment as they move. Each such change may be limited to the link layer, in which case the mobile node simply switches between access points that connect to the same access link. The mobile node can in this case continue to communicate at the IP layer as before. It keeps any previously discovered subnet prefixes and configured IP addresses, and sticks to the access router selected as a first hop for packets destined off-link. However, when the old and new access points belong to different access links, the mobile node changes IP connectivity and needs to reconfigure IP. It invalidates its current first-hop access router along with any previously discovered subnet prefixes, and it removes existing global IP addresses. The mobile node must also rerun Duplicate Address Detection on its link-local IP address. This is necessary even though the link-local IP address keeps its subnet prefix during handover because a new neighbor may already be using the same link-local IP address. New global IP addresses are subsequently configured as the mobile node learns the subnet prefixes in use on the new access link. At the same time, the mobile node chooses a new first-hop access router. The change in IP connectivity may further cause the mobile node to initiate signaling for an IP mobility protocol.

Changes in link-layer attachment can typically be detected straightforwardly through notifications that the link layer may generate when it switches access points. Detecting changes in IP connectivity, or so-called *movement detection*, is less trivial. Since different access links have disjunct sets of subnet prefixes—barring the link-local prefix that is valid on every link—movement detection is commonly implemented by analyzing the subnet prefixes advertised in Router Advertisement messages and probing reachability of access routers considered off-link. When the subnet prefixes

in use by the mobile node are no longer seen to be advertised, but new subnet prefixes show up instead, the mobile node would typically decide that it has moved to a different access link. On the other hand, received subnet prefixes may also indicate that IP connectivity did not change in spite of a link-layer handover, for example, when the mobile node switches access points that connect to the same link. IP reconfiguration should then be avoided.

Movement detection is complicated by the fact that Router Advertisement messages may include incomplete sets of subnet prefixes. Reception of a single advertisement is therefore usually insufficient to decide whether IP connectivity has changed. Mobile nodes also get no indications of an access router's advertising rate. Failure to receive an expected Router Advertisement message therefore does not imply movement either. A typical movement detector would hence draw a possibly premature decision based on a small number of received Router Advertisement messages and, if a change in IP connectivity is assumed, perform Neighbor Unreachability Detection on the first-hop access router to corroborate this.

14.6.3 IP Address Configuration

A node may be preconfigured with one or more IP addresses, but more typically, the node autoconfigures a link-local IP address and one global IP address per subnet prefix assigned to the access link. The set of on-link subnet prefixes is announced by local access routers in multicast Router Advertisement messages, which are retransmitted on a loosely periodic basis. The node may listen for advertisements or, if it is unwilling to wait, actively request one by sending a Router Solicitation message. IP address autoconfiguration can be performed in either stateless or stateful manner, where the modes supported⁵ on an access link are indicated in the Router Advertisement messages.

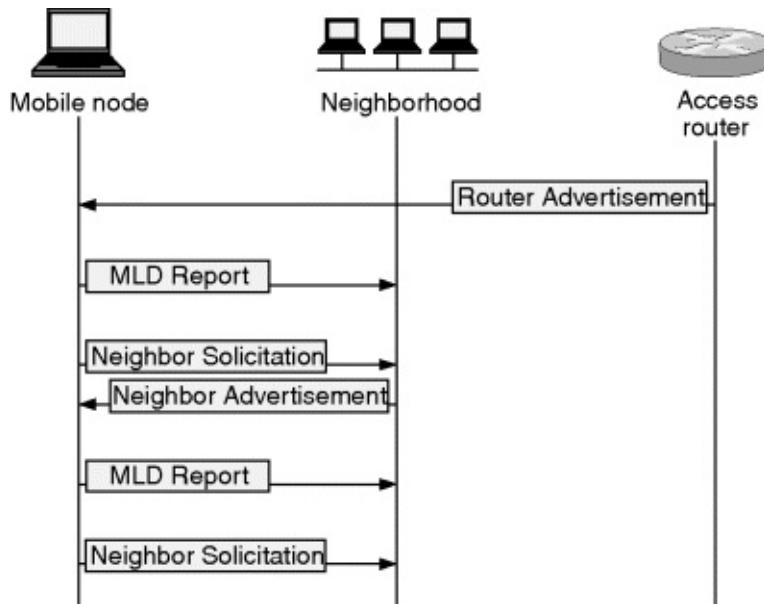
14.6.3.1 Stateless IP Address Autoconfiguration

Stateless Address Autoconfiguration [29] is a protocol that enables a node to autoconfigure new IP addresses for received subnet prefixes, or for the link-local subnet prefix, without external help, such as from an IP address server. The node chooses an interface identifier—based on the link-layer address of the interface to which the IP address is to be assigned [30], or randomly [31], or cryptographically [32] —and prepends to this the obtained subnet prefix.

Stateless Address Autoconfiguration requires the node to subscribe to a *solicited-node multicast group* derived from the desired IP address [33]. This is necessary to ensure nodes that simultaneously attempt to autoconfigure the same IP address can recognize the collision. The subscription is handled by *Multicast Listener Discovery (MLD)* [34], a protocol which provides for this purpose a Report message that the node multicasts onto the access link. The node then runs *Duplicate Address Detection*, a subprotocol of Stateless Address Autoconfiguration, to verify whether the desired IP address is unique: It sends a Neighbor Solicitation message for the IP address to the aforementioned solicited-node multicast group, and listens for a defending Neighbor Advertisement message that any node already using the IP address would send in response. If no responses are received within a period of 1 second, the node assigns the new IP address to the interface. If it receives a Neighbor Advertisement message or another Neighbor Solicitation message for the same IP address, then the node aborts the IP address autoconfiguration process and possibly retries with a different IP address. The probability for an IP address collision is small enough [35] to make it negligible, however. Multiple IP addresses can be autoconfigured in parallel.

The time-sequence diagram in [Figure 14.15](#) exemplifies the use of Stateless Address Autoconfiguration. It shows a node which, after receiving a Router Advertisement message that includes the set of on-link subnet prefixes, attempts to configure an IP address that is already in use. The node then generates a new IP address and tries again, this time with success.

Figure 14.15 Example of the Stateless Address Autoconfiguration procedure.



14.6.3.2 Stateful IP Address Autoconfiguration

Where access routers specify in transmitted Router Advertisement messages that global IP addresses may be configured in stateful manner, nodes can use the Dynamic Host Configuration Protocol for IPv6 (DHCPv6) [36], to request the assignment of a global IP address from a server. This involves either a four-way handshake or a two-way handshake, depending on the configuration of access routers. The node initiates either exchange with a multicast Solicit message. Per default, DHCPv6 servers that receive the Solicit message respond with an Advertise message. The node collects these responses for a random time of between 1.0 and 1.1 seconds, finally selects one of them, and sends the originating DHCPv6 server a Request message. The selected server completes the message exchange with a Reply message, including any IP addresses assigned to the node. Alternatively, a DHCPv6 server may be configured to respond to the node's Solicit message directly with a Reply message that contains any assigned IP addresses. This completes IP address assignment with a two-way handshake. Both message exchanges require the node to use a link-local IP address as its IP source address. Consequently, a link-local IP address cannot be autoconfigured through DHCPv6 and needs to be obtained via Stateless Address Autoconfiguration beforehand.

14.6.4 Neighbor Unreachability Detection

The mapping between IP addresses and link-layer addresses may grow stale as nodes shut down or leave the access link, for example, due to mobility. Nodes are recommended to take advantage of connectivity indications from protocol layers above IP in monitoring a neighbor's reachability. Such indications include acknowledgments received for previously transmitted data. Yet, upper-layer indications may not always be available. A node can then use *Neighbor Unreachability Detection* to actively probe a neighbor's liveliness. Neighbor Unreachability Detection is a subprotocol of Neighbor Discovery. A node interested in a neighbor's reachability sends up to three Neighbor Solicitation messages directly to the link-layer address known for that neighbor. Consecutive solicitations are spaced by 1 second during which the solicited node is expected to respond with a Neighbor Advertisement message. If such an advertisement is sent, the link-layer address known for the neighbor is still correct. If no advertisement is received after three solicitations have been sent, the link-layer address is assumed to be stale and the querying node starts with IP address resolution anew. The potential for packet loss during Neighbor Unreachability Detection is covered by the retransmissions of the Neighbor Solicitation message.

14.6.5 Internet Control Message Protocol for IP Version 6

The Internet Control Message Protocol for IPv6 (ICMPv6) [37], defines error and notification messages that IPv6 nodes exchange to convey or retrieve information relevant at the IP layer. For example, a host that receives a packet it cannot process due to lack of support for the included payload may return an ICMPv6 Parameter Problem message to the sender of the packet. Routers that do not know how to forward a received packet may inform the sender with an ICMPv6 Destination Unreachable message. Protocols at the IP layer or above use ICMPv6 messages to detect error conditions. For example, Mobile IPv6 utilizes ICMPv6 Parameter Problem messages to determine when a correspondent node does not support route optimization.

14.6.6 Optimizations

The default protocols for router and subnet prefix discovery, IP address autoconfiguration, and movement detection were designed for use in

environments where nodes are primarily stationary. Listen times, desynchronization delays, and rate limitations are hence dimensioned conservatively, precluding race conditions and transmission collisions at the cost of latency. For stationary nodes, this is typically not a problem since they can run a working IP configuration for a long time. But the extended latencies [38, 39] in standard protocols are highly disadvantageous in environments where mobile nodes frequently reconfigure IP. Handover delays may then be in the order of seconds. This makes it difficult to support non-real-time or soft-real-time services like file transfers or media streaming, and it precludes any meaningful use of hard-real-time applications such as Internet telephony. A multitude of optimizations have therefore recently been put forth to streamline handover-related activities and reduce handover delays. Particularly promising are the following approaches.

14.6.6.1 Router and Subnet Prefix Discovery

Strict rate limitations in the Neighbor Discovery protocol force access routers to space consecutive Router Advertisement messages by a random time of between 3 and 4 seconds at least. This leads to considerable delays for router and subnet prefix discovery. Router Solicitation messages may in some cases permit a mobile node to obtain an advertisement faster. But on a durable basis, solicited advertisements, too, cannot be sent more rapidly than once per 3.5 seconds on average. Less rigid rate limitations make router and subnet prefix discovery more efficient and thereby also improve movement detection. Accordingly, average advertising rates of between 30 ms and 70 ms were permitted a posteriori for mobile environments [5]. Rate limitations for solicited Router Advertisement messages were not relaxed at the same time, so unless access links are only very sparsely populated with mobile nodes and Router Solicitation messages are transmitted only very infrequently, any solicited Router Advertisement messages are highly likely to be substituted by unsolicited advertisements that are sent at a high rate anyway. The high advertising rates thus in fact redundantize the use of Router Solicitation messages. An obvious disadvantage of this approach to router and subnet prefix discovery is that it comes at the cost of increased bandwidth consumption.

More sophisticated scheduling intervals in access routers can improve router and subnet prefix discovery with respect to both bandwidth consumption and efficiency. *Fast Router Advertisement* [40] is an

optimization for Neighbor Discovery that permits a mobile node to solicit an immediate Router Advertisement message. Randomized desynchronization delays are here replaced by an algorithm that neighboring access routers use to determine an order in which they respond to the Router Solicitation message without collision, whereby the access router ranked first responds without delay. The algorithm is seeded with neighboring access routers' link-local IP addresses and the IP source address of the solicitation.

14.6.6.2 IP Address Autoconfiguration

Stateless Address Autoconfiguration suffers decreased efficiency due to both the mandatory 1 second listen time for nodes that have sent a Neighbor Solicitation message, and a random desynchronization delay of up to 1 second for Multicast Listener Discovery Report messages. *Optimistic Duplicate Address Detection* [41] eliminates both of these delays. This protocol is tailored to mobile environments in particular, so desynchronization delays are removed based on the assumption that natural movement patterns already provide sufficient randomness to avoid collisions in most cases. Moreover, the protocol enables a node to use an IP address in parallel with verification. Nodes temporarily change the rules by which they do Neighbor Discovery so as to avoid pollution of neighboring nodes' IP address resolution state with illegitimate information. This implies restrictions on direct packet exchanges with neighboring nodes, but does not impact communications with correspondent nodes off-link.

14.6.6.3 Movement Detection

Low rates of Router Advertisement messages in combination with the need to acquire multiple advertisements for complete subnet prefix information are responsible for long movement detection delays in standard IPv6. If Neighbor Unreachability Detection on the current access router is performed as part of movement detection, these delays increase even further. Moreover, Neighbor Unreachability Detection in this case happens at a time at which a change in IP connectivity may have invalidated all existing IP addresses. So since the mobile node cannot leave its IP source address unspecified during Neighbor Unreachability Detection, it must reverify the uniqueness of its link-local IP address before it initiates the procedure. Overall, reliable movement detection may thus easily take up to 10 seconds or more [39].

One reason for the low movement detection performance is that mobile nodes do not know in which intervals they can expect to receive Router Advertisement messages from a particular access router, so they cannot efficiently conclude a change in IP connectivity when such advertisements cease to appear. One important enhancement of Neighbor Discovery was therefore the introduction of an Advertisement Interval option [5] that access routers can include in their Router Advertisement messages to declare an upper bound on the intervals in which they transmit these messages. The advertised upper bound aids mobile nodes in deciding when the absence of advertisements from a specific access router indicates a change in IP connectivity. If combined with the increased advertising rates described above, this increases the efficiency of standard movement detection becomes considerably more efficient.

On the other hand, even with increased advertisement rates and the use of Advertisement Interval options, standard movement detection still fails to detect a change in IP connectivity on the absence of a single expected advertisement, due to the potential for packet loss, or on the presence of a single unexpected advertisement, due to the incompleteness of included subnet prefixes. Standard movement detection hence requires the mobile node to wait for the lack of multiple Router Advertisement messages until a change in IP connectivity can be concluded with reasonable certainty. To solve this problem, the IETF set about developing mechanisms that could quickly detect movement based on a single Router Advertisement message from a new access router, rather than waiting for missing advertisements from an old access router. Two mechanisms originated from this. *Complete Prefix List* [42] works with unmodified legacy access routers. A mobile node maintains a list of learned subnet prefixes, possibly obtained by reception of multiple Router Advertisement messages. After the list has matured for a while, the mobile node can assume a change in IP connectivity with high probability when a newly received Router Advertisement message exclusively contains subnet prefixes not in the list. However, such predictions are based on potentially incomplete information, so the mobile node might assert movement even when none actually occurred.

The *DNA Protocol* [43] integrates Fast Router Advertisement for timely transmissions of solicited Router Advertisement messages. Neighboring access routers additionally choose a certain subnet prefix to serve as an *access link identifier* and be as such indicated in all transmitted Router

Advertisement messages. This allows a mobile node to reliably detect changes in IP connectivity based on a single advertisement. Alternatively, the mobile node can explicitly check with access routers as part of the solicitation-advertisement exchange whether a subnet prefix used before a link-layer handover, as such called a *landmark*, is still valid after the link-layer handover. The DNA Protocol used to refer to Complete Prefix List as a fall-back mechanism for access to links with legacy access routers. More recently, the Complete Prefix List algorithms have been fully integrated into the DNA Protocol. They are now no longer maintained as a separate protocol.

14.6.7 Media-Independent Handover Services

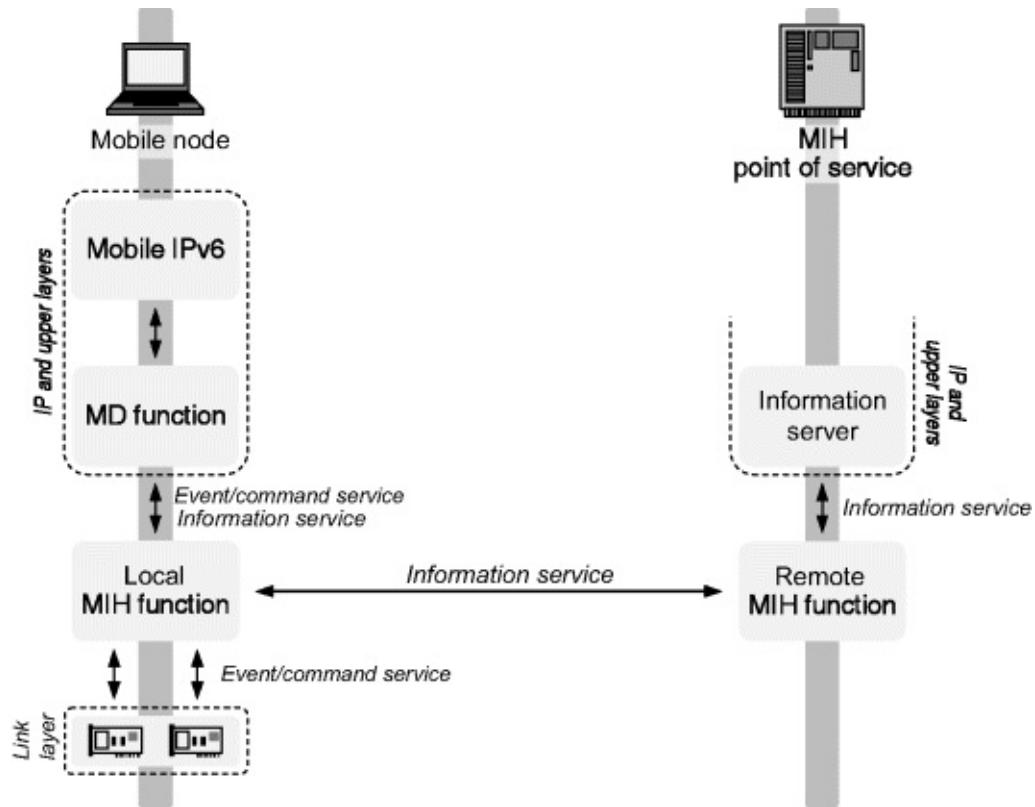
Efficient IP mobility management requires a mobile node to quickly identify and select suitable access points prior to handover, and to closely synchronize activities at the link and IP layers during handover execution. Both of these prerequisites are hardly met today, however. Classic access point selection techniques are based on signal strength alone. They disregard other important aspects such as operator, service provider, and cost; or link-layer properties such as security and quality of service. A mobile node may further be unable to connect at the IP layer despite a working link-layer attachment, for example, due to a mismatch in supported IP address configuration mechanisms. The selection of an access point becomes even more challenging if multiple access technologies are involved. And proactive mobility management introduces the additional requirement that mobile nodes must obtain subnet prefix information for a discovered access link so as to autoconfigure new IP addresses for that access link prior to handing over to it. Cross-layer interaction and synchronization is a concept that deviates from the traditional view that the responsibilities of different layers in a network stack should be separated [44].

To approach these challenges, the IEEE chartered its 802.21 working group to develop a set of Media-Independent Handover Services [23] for improved reactive and proactive mobility management. The standard includes a unified interface separating different heterogeneous technologies at the link layer from protocols at the IP layer or above. This interface defines a set of events and commands that protocols at the IP layer or above can read or issue, respectively, to synchronize their handover-related activities with the link layer. The standard further enables IP-layer protocols to acquire information

that assists them in access link selection and handover preparations. Overall, the functionality provided by Media-Independent Handover Services is split into three complementary services: an event service, a command service, and an information service.

Pivotal to all Media-Independent Handover Services operation is an *MIH function*, which serves as an intermediary between the handover-related activities at the link layer and the layers above. (Here, MIH denotes media independent handover.) One instance of the MIH function is located between a mobile node's link and IP layers, as shown on the left-hand side in [Figure 14.16](#). This shim provides an abstraction from the specifics of different access technologies and presents a homogeneous interface to protocols at the IP layer or above. Interested protocols include mobility or multihoming protocols, movement detection mechanisms, as well as protocols for neighbor discovery or IP address autoconfiguration. For example, the DNA Protocol may send a router solicitation message for renewed router discovery and movement detection when the mobile node switches to a different access point. The DNA Protocol may further interface to Stateless Address Autoconfiguration so as to create a new on-link IP address when the link-layer handover to the new access point is found to cause a change in IP connectivity. A mobility protocol such as Mobile IPv6 may finally initiate a binding update for the new on-link IP address. In the reference architecture in [Figure 14.16](#), the MD function implements movement detection (so MD stands for movement detection).

[Figure 14.16](#) Reference architecture of Media-Independent Handover Services.



A companion entity of the MIH function is located on an *MIH point of service* in the access network. This is depicted on the right-hand side of [Figure 14.16](#). The remote MIH function on the MIH point of service may assist the MIH function on a mobile node during handovers that are controlled by the mobile node. It further facilitates network-controlled handovers, where it is in charge of handover-related activities on the mobile node. For remote information retrieval, the MIH point of service may further run an *MIH information server*. Interested mobile nodes can access information from this database via the remote MIH function. Communications between the MIH functions on the mobile node and on an MIH point of service use the *MIH protocol*. This defines a set of packet formats that encapsulate events, commands, and information for transmission via the network.

As a mobile node moves across access links, its peer MIH point of services may change. Media-Independent Handover Services define an *MIH function discovery* to aid the mobile node in finding another MIH point of service after attaching to a new link. The draft standard limits the discovery to the link layer, so the mobile node can only find MIH points of service within the same broadcast domain. A mobile node typically discovers new MIH points

of service immediately after a handover, but uses the available services only later when preparing for a subsequent handover. This generally provides for a sufficient time frame for MIH function discovery to proceed without delaying handover-related activities.

14.6.7.1 Event Service

Protocols at the mobile node's IP layer or above can use the event service to register with the local MIH function for notifications about specific events at the link layer, such as changes in link-layer attachment or in the quality of a particular connection. The MIH function monitors the mobile node's interfaces for the respective events in a technology-specific manner and generates a technology-independent notification for the interested protocols at higher layers as the events occur.

An example for the utility of the event service is movement detection for mobile nodes. Movement detection mechanisms at the IP layer are interested in changes in link-layer attachment, so they may subscribe to notifications about these events from the MIH function. The MIH function monitors interfaces at the link layer for changes in attachment and accordingly generates technology-independent Link Up Indication and Link Down Indication messages for the movement detection mechanism. This allows a movement detection mechanism to quickly review the current IP configuration in case of a link-layer attachment change and to reconfigure if need be. In proactive mobility management, the MIH function may anticipate changes in link-layer attachment and notify protocols at the IP layer about this with a Link Going Down Indication message. This facilitates handover preparations and proactive mobility management at the IP layer.

Events may also originate with a remote MIH function on an MIH point of service in the network. The events are in this case sent via the network using the MIH protocol. For example, an MIH function in the network may send an Link Going Down Indication message to the MIH function on a mobile node when the access point to which the mobile node attaches is about to be shut down for administrative reasons. The MIH function on the mobile node can then inform IP-layer protocols about the forthcoming link break.

14.6.7.2 Command Service

The command service enables protocols at the IP layer or above to control

handover-related activities at the local link layer or remotely in the network. A command is passed to the local MIH function and is as such technology independent. If the command is directed to the local link layer, the MIH function translates it into technology-specific commands and sends these to the respective interfaces. The interfaces execute the commands and return a confirmation each, which the MIH function aggregates and forwards to the protocol that invoked the command. A remote command is directed to the MIH function on an MIH point of service in the network. It, too, originates with a protocol at the IP layer or above and traverses the local MIH function. But from there the command is forwarded to the remote MIH function by help of the MIH protocol. The remote MIH function may forward the command to a protocol at a higher layer, or translate it into technology-specific commands for the link layer beneath.

The command service can be used by a protocol at the IP layer or above to find a suitable access point and to control a link-layer handover thereto. For example, an IP-layer movement detection mechanism or mobility protocol on a mobile node may periodically send a Get Status Request message to the local MIH function for a survey on available points of attachments and their properties such as bandwidth, packet loss rate, and medium access delays. The MIH function translates this command into technology-specific Get Status Request messages, one for each of the mobile node's interfaces, and receives Get Status Confirm messages containing the desired information in response. The MIH function aggregates the responses within a technology-independent Get Status Confirm message and relays this to the protocol that asked for it. In a similar manner, the Scan Request message can be used by protocols at the IP layer or above to measure the received signal strength of available access points. The local MIH function maps this command onto a set of technology-specific Link Scan Request messages, one for each interface, and receives a set of Link Scan Confirm messages in return. The responses are aggregated and forwarded as an Scan Confirm message to the calling protocol.

Another example for the use of the command service is in proactive mobility management, where the IP layer needs to be in control of link-layer detachment and attachment. A movement detection or mobility protocol may here send the local MIH function a Switch Request message identifying the old and selected target access link. The MIH function translates the Switch Request message into a sequence of commands specific to the one or two

involved link-layer interfaces: If the link layer technology supports make-before-break handovers, a Link Connect Request message for the target access point is sent first, and a Link Disconnect Request message for the old access point is sent second. Where only break-before-make handovers are possible, the sequence of messages is reverse. The link layer responds to the requests with Link Disconnect Confirm and Link Connect Confirm messages, and the MIH function finally sends a Switch Confirm message to the protocol that invoked the handover.

14.6.7.3 Information Service

The information service enables a mobile node to discover auxiliary information about access links within reachability. Such information includes the access technology; access link, operator, and service provider identification; link-layer security and quality-of-service properties; permitted IP configuration methods such as Stateless Address Autoconfiguration or DHCPv6; the link-layer address of the access point and the channels it is operating on; as well as the IP address of available access routers and the set of subnet prefixes in use on the access link. The information service thus supports the mobile node in preparing a handover and selecting a suitable target access point. One particular use case for the information service is in proactive mobility management, where a mobile node must obtain the subnet prefixes of a new access link before it actually attaches to that link. The mobile node can here use the information service to resolve the link-layer address of a discovered access point into a list of subnet prefixes in use on the link to which this access point attaches.

Retrievable information is organized in *information elements*. Protocols at the mobile node's IP layer or above may request these either from the local MIH function or, more generally, via a remote MIH function from an MIH information server in the network. In either case, an interested protocol sends a Get Information Request message to the local MIH function, identifying one or multiple information elements that are to be retrieved. If the requested information is available locally, the MIH function responds directly with an Get Information Confirm message including the respective information elements.

On the other hand, if the requested information is not available locally, the MIH function on the mobile node forwards the Get Information Request

message to the remote MIH function on a previously discovered MIH point of service. The remote MIH function translates the received Get Information Request message into an Get Information Indication message and sends this to an MIH information server to retrieve the requested information. The MIH information server looks up the desired information elements and returns them within a Get Information Response message. The remote MIH function copies the information elements into a Get Information Confirm message and sends it to the local MIH function on the mobile node. The latter finally relays the message to the protocol that requested the included information elements.

Notes

- 1.** Truly unpredictable sequence number selection was added to TCP a posteriori [attacks]22. The original TCP specification suggested initializing sequence numbers from a monotonically increasing clock, thus allowing an attacker to guess an initial sequence number.
- 2.** An initial waiting time of 1.5 seconds is required before the mobile node retransmits a Binding Update message for its home agent if the home agent does not yet have a binding registered for the mobile node. The increased waiting time then allows the home agent to complete Duplicate Address Detection on the mobile node's home address before it sends a Binding Acknowledgment message.
- 3.** While the a-priori transmission of a Binding Update message to the home agent helps the latter to forward home address test messages correctly, there is technically no reason for requiring the mobile node to wait for the completion of the home registration before a Binding Update message can be sent to a correspondent node. The rule is also not needed from a security standpoint since neither the home agent nor a correspondent node can verify that a mobile node conforms to it.
- 4.** Another form of access network multihoming is for an access network to obtain an individual block of IP addresses that does not belong to any particular Internet service provider. Multihoming is in this case transparent to the nodes, hence node support is not required.
- 5.** There is currently disunity in the Internet Engineering Task Force (IETF) regarding whether the IP address autoconfiguration mode advertised in Router Advertisement messages should be interpreted as mandatory or as optional.

References

1. Etoh, M. and Yoshimura, T., "Wireless Video Applications in 3G and Beyond," IEEE Wireless Commun, vol. 12, no. 4, pp. 66-72, Aug. 2005.
2. Mockapetris, P.V., "Telephony's Next Act," IEEE Spectrum, vol. 43, no. 4, pp. 28-32, Apr. 2006.
3. Ortiz, S. Jr., "Internet Telephony Jumps off the Wires," IEEE Computer, vol. 37, no. 12, pp. 16-19, Dec. 2004.

- 4.** Perkins, C. (Editor), “IP Mobility Support for IPv4,” IETF RFC 3344, Aug. 2002.
- 5.** Johnson, D., Perkins, C.E. and Arkko, J., “Mobility Support in IPv6,” IETF RFC 3775, Jun. 2004.
- 6.** de Launois, C. and Bagnulo, M., “The Paths Towards IPv6 Multihoming,” IEEE Commun. Surveys & Tutorials, vol. 8, no. 2, pp. 38-51, Apr. 2006.
- 7.** Nikander, P., “End-Host Mobility and Multihoming with the Host Identity Protocol,” IETF RFC 5206, Apr. 2008.
- 8.** Stewart, R., Xie, Q., Tuexen, M., Maruyama, S. and Kozuka, M., “Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration,” IETF RFC 5061, Sep. 2007.
- 9.** Ramjee, R., Varadhan, K., Salgarelli, L., Thuel, S.R., Wang, S.Y. and La Porta, T., “HAWAII: A Domain-based Approach for Supporting Mobility in Wide-Area Wireless Networks,” IEEE/ACM Trans. on Networking, vol. 10, no. 3, pp. 396-410, June 2002.
- 10.** Valkó., A.G., “Cellular IP: A New Approach to Internet Host Mobility,” ACM SIGCOMM Computer Communication Review, vol. 29, no. 1, pp. 50-65, Jan. 1999.
- 11.** Montenegro, G. (Editor), “Reverse Tunneling for Mobile IP, Revised,” IETF RFC 3024, Jan. 2001.
- 12.** Ferguson, P. and Senie, D., “Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing,” IETF RFC 2827, May 2000.
- 13.** Soliman, H., Castelluccia, C., El Malki, K. and Bellier., L., “Hierarchical Mobile IPv6 Mobility Management (HMIPv6),” IETF RFC 4140, Aug. 2005.
- 14.** Fogelstroem, E., Jonsson, A. and Perkins, C., “Mobile IPv4 Regional Registration,” IETF RFC 4857, Jun. 2007.
- 15.** Moskowitz, R. and Nikander, P., “Host Identity Protocol (HIP) Architecture,” IETF RFC 4423, May 2006.
- 16.** Loughney, J. (Editor), “IPv6 Node Requirements,” IETF RFC 4294, Apr. 2006.
- 17.** Nikander, P., Arkko, J., Aura, T., Montenegro, G. and Nordmark, E., “Mobile IP Version 6 Route Optimization Security Design Background,” IETF RFC 4225, Dec. 2005.

- 18.** Perkins, C., “Securing Mobile IPv6 Route Optimization Using a Static Shared Key,” IETF RFC 4449, June 2006.
- 19.** Vogt, C. and Arkko, J., “A Taxonomy and Analysis of Enhancements to Mobile IPv6 Route Optimization,” IETF RFC 4651, Feb. 2007.
- 20.** Montenegro, G. and Castelluccia, C., “Crypto-based Identifiers (CBIDs): Concepts and Applications,” ACM Transactions on Information and System Security, vol. 7, no. 1, pp. 97-127, Feb. 2004.
- 21.** Arkko, J., Vogt, C. and Haddad, W., “Enhanced Route Optimization for Mobile IPv6,” IETF RFC 4866, May 2007.
- 22.** Bellovin, S.M., “Defending Against Sequence Number Attacks,” IETF RFC 1948, May 1996.
- 23.** IEEE, “P802.21a -Standard for Local and Metropolitan Area Networks: Media Independent Handover Services,” P802.21/D01.00, Mar. 2006.
- 24.** Krishnan, S., Montavont, N., Njedjou, E., Veerepalli, S. and Yegin, A., “Link-Layer Event Notifications for Detecting Network Attachments,” IETF RFC 4957, Aug. 2007.
- 25.** Koodli, R. (Editor), “Fast Handovers for Mobile IPv6,” IETF RFC 4068, Jul. 2005.
- 26.** Kuladinithi, K., Fikouras, N.A., Goerg, C., Georgios, K. and Pavlidou, F.N., “Filters for Mobile IPv6 Bindings (NOMADv6),” IETF Internet Draft, draft-nomadv6-mobileip-filters-03.txt (work in progress), Oct. 2005.
- 27.** CERT Coordination Center, “CERT Advisory CA-1996-21 TCP SYN Flooding and IP Spoofing Attacks,” <http://www.cert.org/advisories/CA-1996-21.html>, Sep. 1996.
- 28.** Narten, T., Nordmark, E., Simpson, W. and Soliman, H., “Neighbor Discovery for IP version 6 (IPv6),” IETF RFC 4861, Sep. 2007.
- 29.** Thomson, S., Narten, T. and Jinmei, T., “IPv6 Stateless Address Autoconfiguration,” IETF RFC 4862, Sep. 2007.
- 30.** Hinden, R.M. and Deering, S.E., “Internet Protocol Version 6 (IPv6) Addressing Architecture,” IETF RFC 4291, Feb. 2006.
- 31.** Narten, T. and Draves, R., “Privacy Extensions for Stateless Address Autoconfiguration in IPv6,” IETF RFC 3041, Jan. 2001.
- 32.** Arkko, J., Kempf, J., Zill, B. and Nikander, P., “SEcure Neighbor Discovery (SEND),” IETF RFC 3971, Mar. 2005.

- 33.** Christensen, M., Kimball, K. and Solensky, F., "Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches," IETF RFC 4541, May 2006.
- 34.** Vida, R. and Costa, M.K. (Editors), "Multicast Listener Discovery Version 2 (MLDv2) for IPv6," IETF RFC 3810, June 2004.
- 35.** Bagnulo, M., Soto, I., Garcia-Martinez, A. and Azcorra, A., "Random Generation of Interface Identifiers," IETF Internet Draft, draft-soto-mobileip-random-iids-00.txt, Jan. 2002.
- 36.** Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C.E. and Carney, M., "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)," IETF RFC 3315, Jul. 2003.
- 37.** Conta, A., Deering, S. and Gupta, M. "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification," IETF RFC 4443, Mar. 2006.
- 38.** Vogt, C. and Zitterbart, M., "Efficient and Scalable, End-to-End Mobility Support for Reactive and Proactive Handoffs in IPv6," IEEE Commun. Mag., vol. 44, no. 6, pp. 74-82, June 2006.
- 39.** Vogt, C., "A Comprehensive Delay Analysis for Reactive and Proactive Handoffs with Mobile IPv6 Route Optimization," Technical Report TM-2006-1, Institute of Telematics, Universitaet Karlsruhe (TH), Germany, Jan. 2006.
- 40.** Daley, G., Pentland, B. and Nelson, R., "Movement Detection Optimizations in Mobile IPv6," Proc. of IEEE International Conference on Networks, pp. 687-692, Sep. 2003.
- 41.** Moore, N., "Optimistic Duplicate Address Detection (DAD) for IPv6," IETF RFC 4429, Apr. 2006.
- 42.** Choi, J. and Nordmark, E., "DNA with Unmodified Routers: Prefix List Based Approach," IETF Internet Draft draft-ietf-dna-cpl-01.txt, Apr. 2005.
- 43.** Narayanan, S., Kempf, J., Nordmark, E., Pentland, B., Choi, J., Daley, G., Montavont, N. and Moore, N., "Detecting Network Attachment in IPv6 Networks (DNAv6)," IETF Internet Draft, draft-ietf-dna-protocol-03.txt, Oct. 2006.
- 44.** Zimmermann, H., "OSI Reference Model –The ISO Model of Architecture for Open Systems Interconnection," IEEE Tran. Commun., vol. 28, no. 4, pp. 425-432, Apr. 1980.

Index

Access link identifier Access network multihoming Acknowledgment (ACK)
Add/drop multiplexer (ADM) Addr length
Address family identifier (AFI) Address sharing
Address-and-control-field-compression (ACFC) Admission control
Anycast address
Anycasting
AppleTalk
Arpanet
AS path length
Asymmetrical digital subscriber line (ADSL) Asynchronous transfer mode (ATM) Asynchronous-control-character-map (ACCM) ATM adaption layer 5 (AAL5) AToM and L2TPv3
Attribute value pairs (AVPs) Authentication
Authentication extension header Autonomous system (AS)
Backbone area
Backscatter analysis
Backward explicit congestion notification (BECN) Backwards compatibility
Bandwidth-tolerant
Bidirectional tunneling Bilateral peering arrangement (BLPA) Binding
Binding cache
Binding identifier
Binding update
Binding update message Border area router
Border gateway protocol (BGP) Bot
Bottom of stack
Bridge
Bridge protocol data unit (BPDU) Broadcast IP

CAIDA
Care-of address
Care-of address test

Care-of init cookie
Care-of keygen token
Care-of nonce index
Care-of test init message Cell loss priority (CLP) bits Certificate
Checksum
Clarification
Class-based triggering mechanism Classless inter-domain routing (CIDR)
Codepoint
Commercial internet exchange (CIX) Common object request broker architecture (CORBA) Common part convergence sublayer---service data unit (CPCSSDU) Community
Complete prefix list
Congestion control
Connectionless network protocol (CLNP) Constraint shortest path first (CSPF) Controlled link sharing Core-based tree (CBT) protocol
Correspondent deregistration Correspondent registration Count-to-infinity problem
Data link connection identifier (DLCI) Datagram
Default PHB (DE PHB)
Delay
Denial-of-service (DoS) Denial-of-service attack Destination address
Differentiated services (DiffServ, DS) Differentiated services code point (DSCP) Digital subscriber line access multiplexer (DSLAM) Dijkstra algorithm
Directed graph
Discard eligibility (DE) Distance vector
Distance vector multicast routing protocol (DVMRP) DNA protocol
Domain name system (DNS) DoS attack pattern
DoS detection
Dropper
DS codepoint
Dual stack
Duplicate address detection DVMRP, *see* Distance vector multicast routing protocol (DVMRP) Dynamic host configuration prefix (DHCP) Dynamic host configuration protocol for IPv6 (DHCPv6) Dynamic routing

E-bit

Edge network

Edge network topology EEAC-SV

Element management system (EMS) Emulated virtual circuits Encapsulating security payload extension header Encapsulation

Encryption

Enhanced interior gateway routing protocol (EIGRP) Equal-cost multipath (ECMP) Ethernet

Ethernet 802.1q VLAN

Ethernet emulation

Ethernet pseudowire

Ethernet virtual LAN (Ethernet VLAN) Expedited forwarding PHB (EF PHB) Experimental bits (EXP) Explicit endpoint admission control (EEAC) Explicit forward congestion indication (EFCI) Exterior gateway protocol (EGP) External BGP (eBGP)

Fail-over

Fast rerouting

Fast router advertisement Fault tolerance

Federal internet exchange (FIX) Fiber-switch capable (FSC) File transfer protocol (FTP) Fine two-phase routing (F-TPR) Flooding

Flooding attack

Flow control

Flow descriptor

Flow label

Flowspec

Forging

Forward compatibility Forward explicit congestion notification (FECN)

Forwarding equivalent class (FEC) Fragmentation extension header Frame

Frame relay

Generalized multiprotocol label switching (GMPLS) Generic routing encapsulation (GRE) Generic routing encapsulation (GRE) tunnel

Header
Hello
Hierarchical routing
High-level data link control (HDLC) Home address
Home address test
Home deregistration
Home init cookie
Home keygen token
Home link
Home nonce index
Home registration
Home test init message Hop limit
Hop-by-hop options extension Hose model
Host
Host address
Host identity protocol Host route
Hypertext transfer protocol (HTTP)
ICMP traceback
Impersonation attack
Information elements
Input driver
Integrated services (IntServ) Integrated services digital network (ISDN)
Interexchange carrier (IXC) Interior gateway protocol (IGP) Interior gateway
routing protocol (IGRP) Intermediate model
Intermediate system to intermediate system (IS-IS) Internet
Internet architecture board Internet control message protocol (ICMP) Internet
control message protocol for IPv6 (ICMPv6) Internet engineering task force
(IETF) Internet exchange point (IXP) Internet forwarder
Internet group management protocol (IGMP) Internet protocol (IP) Internet
service provider (ISP) IP address
IP address resolution IP anycasting
IP extension header
IP multicasting
IP spoofing
IP traceback

IP version 4 (IPv4)
IP version 6 (IPv6)
IP-in-IP
IP-layer handover
IP-only L2VPN
IPSec
IPSec/point-to-point tunneling protocol (PPTP)
 k shortest paths
L2TP access concentrator (LAC) L2TP control message header

L2TP data message

L2TP network server (LNS)

L2TPv3

Label

Label distribution protocol (LDP) Label edge router (LER) Label merging
Label stacking
Label switched path (LSP) Label switching router (LSR) Lambda-switch capable (LSC) Landmark
Layer
Layer-2 tunneling protocol (L2TP) Layer-2 tunneling protocol version 3 (L2TPv3) Layer-3 VPN (L3VPN)
LDP
LEC
Linear programming (LP) Link
Link management protocol (LMP) Link state
Link testing
Link weight
Link weight matrix
Link-layer handover

Link-state advertisement (LSA) Link-state database (LSDB) Link-state protocol

Load balancing

Local area network (LAN) Local exchange carrier (LEC) Local preference

Loss sensitive

Lucent Bell Labs

Martini-encaps

Martini-trans

Maximum receive unit (MRU) Maximum transmission unit (MTU) MC-bit

Media independent handover Message authentication codes (MAC) Meter

Microsoft point-to-point encryption (MPPE) MIH function

Mobile IPv6

Mobility support in IPv6

Movement detection

Multi exit discriminator (MED) Multi level tree for online packet statistics (MULTOPS) Multicast

Multicast listener discovery (MLD) Multicast OSPF (MOSPF) Multicast routing

Multicast scalability Multicast transaction protocol (MTP) Multicasting

Multihoming

Multiprotocol label switching (MPLS) MULTOPS, *see* Multi level tree for online packet statistics

National Science Foundation Neighbor discovery

Neighbor discovery protocol Neighbor solicitation message Neighbor unreachability detection Network access point (NAP) Network address

Network congestion ratio Network element (NE)

Network management system (NMS) Network restoration

Network time protocol (NTP) Next header

Node

Node multihoming

Nonlinear programming NSFnet

On-link IP address

Open shortest path first (OSPF) Open system interconnection reference (OSI)
Optical cross-connect (OXC) Optical interworking forum (OIF) Optical
transport network (OTN) Optical user network interface (O-UNI) Optimistic
duplicate address detection OSPF hello packets

Output driver

Packet

Packet classifier

Packet scheduler

Packet-switch capable (PSC) Packet-switched network (PSN) Path

Path computation element (PCE) Path state

PathTear

Payload

Payload length

Payload packet

PCE communication protocol (PCEP) Peer model

Peering

Per-hop behavior (PHB) PIM router

Pipe model

Plain old telephone service (POTS) Points of presence (POPs) Point-to-point
protocol (PPP) Pop switch

Port number

Post office protocol version 3 (POP3) PPTP tunnel protocol

Prefix

Preventive start-time optimization (PSO) Private branch exchange (PBX)

Proactive mobility management Protocol-independent multicast (PIM)

Protocol-independent multicast dense mode (PIM-DM) Protocol-independent
multicast sparse mode (PIM-SM) Provider

Pseudo wire emulation edge-to-edge (PWE3) Pseudowire service

Public key infrastructure Push

PWE3 network reference model

QoS guarantee

Quality of service (QoS)

Reachability verification Reactive mobility management Real time protocol
(RTP) Recursive enumeration algorithm (REA) Redirection-based flooding

attack Rendezvous router
Repeater
Resource reservation
Resource reservation protocol (RSVP) Resv
Return routability procedure Reverse path broadcasting (RPB) Reverse path forwarding (RPF) Reverse path multicasting Reverse-tunneled
Route optimization
Router
Router discovery
Routing
Routing extension header Routing information protocol (RIP) Routing policy
Routing protocol
Routing table
Rspec
RSVP traffic engineering (RSVP-TE) RTP control protocol (RTCP) Run-time optimization (RO)
Segment
Segmentation and reassembly (SAR) Service differentiation Service vectors (SV)
Session initiation protocol (SIP) Shaper
Shim header
Simple mail transfer protocol (SMTP) Simple multicast routing protocol (SMRP) Simple network management protocol (SNMP) Sliding window
Smart OSPF (S-OSPF)
SMRP multicast transaction protocol Solicited-node multicast group Source address
Source-based tree
Split horizon
Stable IP address
Standard reachability verification Start-time optimization (SO) Stateless address autoconfiguration Stationary rendezvous server Stream control transmission protocol (SCTP) Subnet
Subnet IP
Subnet mask
Subnet prefix discovery Swap

Switch
Synchronize sequence numbers (SYN) Synchronous digital hierarchy (SDH)
Synchronous optical network (SONET)
T-bit
TCP/IP
TDM-switch capable
Telecommunication management network (TMN) Three-way handshake
Throughput
Time to live (TTL)
Time-division multiplexing (TDM) Topology resolution
Traffic engineering (TE) link Transaction language 1 (TL1) Transit
Transit area
Translation
Transmission control protocol (TCP) Transport mode
Triangular routing
Trivial file transfer protocol (TFTP) Tspec
Tunnel
Tunnel mode
Tunneling
Two-phase routing (TPR)
%%
Type of service
Type of service (TOS) Type, length, and value (TLV)
UNI-C
Unicast
Unicast routing
UNI-N
Unverified
Upper-layer protocol
User datagram protocol (UDP)
Variable length subnet mask (VLSM) Verified
Version
Virtual circuit identifier (VCI) Virtual path identifier (VPI) Virtual private
LAN service (VPLS) Virtual private multicast service (VPMS) Virtual

private network (VPN) Virtual private wire service (VPWS) Virtual router redundancy protocol (VRRP) Virus scanning
Voice over internet protocol (VoIP) VPWS L2VPN employs layer-2

Wavelength-division multiplexing (WDM) Weight
Well-known anycast address Wide area network (WAN)
Zombie