QoS for IP/MPLS Networks

A comprehensive guide to implementing QoS inIP/MPLSnetworksusingCiscoIOSandCiscoIOSXR Software

ciscopress.com

Santiago Alvarez, CCIE® No. 3621

QoS for IP/MPLS Networks

Santiago Alvarez

Cisco Press
800 East 96th Street
Indianapolis, IN 46240 USA

QoS for IP/MPLS Networks

Santiago Alvarez

Published by:
Cisco Press
800 East 96th Street
Indianapolis, IN 46240 USA

Printed in the United States of America 1 2 3 4 5 6 7 8 9 0

First Printing June 2006

Library of Congress Cataloging-in-Publication Number is on file.

ISBN: 1-58714-391-7

Trademark Acknowledgments

Warning and Disclaimer

This book is designed to provide information about quality of service in IP/MPLS networks using Cisco IOS and Cisco IOS XR. Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied.

The information is provided on an “as is” basis. The authors, Cisco Press, and Cisco Systems, Inc. shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book or from the use of the discs or programs that may accompany it.

The opinions expressed in this book belong to the author and are not necessarily those of Cisco Systems, Inc.

Corporate and Government Sales

Cisco Press offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales.

For more information please contact: **U.S. Corporate and Government Sales** 1-800-382-3419 corpsales@pearsontechgroup.com

For sales outside the U.S. please contact: **International Sales** international@pearsoned.com

Feedback Information

At Cisco Press, our goal is to create in-depth technical books of the highest quality and value. Each book is crafted with care and precision, undergoing rigorous development that involves the unique expertise of members from the professional technical community.

Readers' feedback is a natural continuation of this process. If you have any comments regarding how we could improve the quality of this book, or otherwise alter it to better suit your needs, you can contact us through e-mail at feedback@ciscopress.com. Please make sure to include the book title and ISBN in your message.

We greatly appreciate your assistance.

Publisher	Paul Boger
Cisco Representative	Anthony Wolfenden
Cisco Press Program Manager	Jeff Brady
Production Manager	Patrick Kanouse
Development Editor	Jill Batistick
Senior Project Editor	San Dee Phillips
Copy Editor	Keith Cline
Technical Editors	Mark Gallo, Raymond Zhang
Book and Cover Designer	Louisa Adair
Composition	Mark Shirar
Indexer	Keith Cline

Printed in the USA

About the Author

Santiago Alvarez, CCIE No. 3621, is a technical marketing engineering for Cisco Systems working on MPLS and QoS since 2000. He joined Cisco in the blazing days of 1997. Prior to Cisco, Santiago worked in software development for Lucent Technologies. He has been involved with computer networking since 1991. Santiago is a frequent speaker at Cisco Networkers and a periodic contributor to *Cisco Packet* Magazine. He holds a bachelor of science degree in computer science from EAFIT University, a master of Science degree in computer science from Colorado State University, and a master of science in telecommunications from the University of Colorado at Boulder. Outside work, he enjoys the outdoors, fine food, and exploring the world as an independent traveler. He can be reached at saalvare@cisco.com.

About the Technical Reviewers

Mark Gallo is a systems engineering manager at Cisco Systems within the channels organization. He has led several engineering groups responsible for positioning and delivering Cisco end-to-end systems, as well as designing and implementing enterprise LANs and international IP networks. He has a B.S. degree in electrical engineering from the University of Pittsburgh and holds Cisco CCNP and CCDP certifications. Mark resides in northern Virginia with his wife, Betsy, and son, Paul.

Raymond Zhang is a senior network architect for BT Infonet in the areas of Global IP backbone infrastructure, routing architecture design, planning, and its evolutions. Currently, his main areas of interest include large-scale backbone routing, traffic engineering, performance and traffic statistical analysis, and MPLS-related technologies (including interdomain traffic engineering, GMPLS, metro Ethernet, Diffserve, IPv6, and Multicast). Raymond participates in several IETF drafts relating to MPLS, BGP-based MPLS VPN, Inter-AS TE, and, more recently, PCE-based work.

Dedications

Thanks for withstanding long, long working hours.

Acknowledgments

I would like to give special thanks to Bob Olsen and Sandeep Bajaj for sharing their technical expertise through so many years. They have patiently tolerated my constant interruptions and have provided useful insight on different topics included in the book.

Special thanks to the reviewers, Mark Gallo and Raymond Zhang. I appreciate your detailed comments. I am to blame for any remaining inaccuracies or omissions.

Big thanks to Bruce Davie, whose responsiveness at key points encouraged me to persist in my goal. I highly regard his unusual ability to abstract complexity and clearly illustrate the essence of intricate technology concepts. Much of his work has directly and indirectly influenced the content of this book. Similarly, I extend my gratitude to François Le Faucheur and Jean Philippe Vasseur. They have had the patience to discuss with me many aspects of these technologies in numerous occasions. *Merci!*

Thanks to Ramesh Uppili for contributing to the presentation of key topics in multiple ways.

I also want to thank Rakesh Gandi, Prashanth Yelandur, Ashish Savla, Bobby Kaligotla, Lawrence Wobker, Ashok Ganesan, Jay Thontakudi, and Scott Yow for facilitating the discussion of Cisco IOS XR in this book.

Special thanks to the Cisco Press team: John Kane, Chris Cleveland, Jill Batistick, San Dee Phillips, and Elizabeth Peterson. I really appreciate your attention to detail and extraordinary patience with me. I wish John the best in his new endeavors.

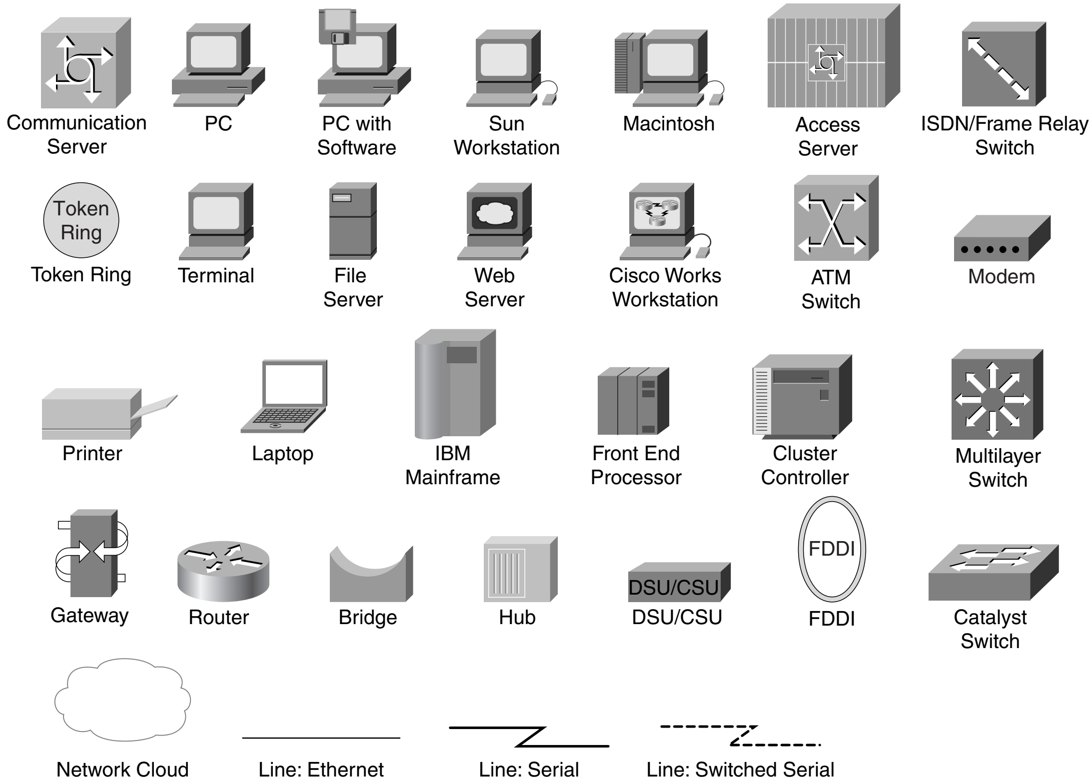
Finally, if you have read this far in search of your name, this paragraph is for you. I have to acknowledge that numerous individuals contributed through insightful discussions. They unhappily or maybe happily remain anonymous. Thanks!

This page intentionally left blank

Contents at a Glance

Contents

Icons Used in This Book



Command Syntax Conventions

The conventions used to present command syntax in this book are the same conventions used in the IOS Command Reference. The Command Reference describes these conventions as follows:

- **Boldface** indicates commands and keywords that are entered literally as shown. In actual configuration examples and output (not general command syntax), boldface indicates commands that are manually input by the user (such as a **show** command).
- *Italics* indicate arguments for which you supply actual values.
- Vertical bars (|) separate alternative, mutually exclusive elements. Note, however, that the vertical bar (pipe operand) is also used to filter command-line interface command output; in that scenario, the operand (|) precedes the **begin**, **exclude**, or **include** keywords, which are then followed by a regular expression.
- Square brackets [ ] indicate optional elements.
- Braces { } indicate a required choice.
- Braces within brackets [{ }] indicate a required choice within an optional element.

Foreword

The phrase “IP QoS” was for many years considered an oxymoron. Indeed, much of the success of the IP architecture could be traced to its adoption of a "best effort" service model, enabling IP to run over just about any underlying network technology. Best effort service, however, is defined by a lack of assurance that packets will be delivered in a timely manner, or even delivered at all. Such a service model limits the potential of IP networks to support applications that demand timely packet delivery, such as interactive telephony and multimedia applications.

As far back as 1979, there were proposals to extend the IP service model to support applications with stronger QoS requirements. However, this remained a research topic until the early 1990s. By that point, the idea of convergence—carrying many applications with diverse QoS needs on a single network—was gaining currency, although the word “convergence” would not become a buzzword for several years. ATM was widely expected to be the packet switching technology that would enable this convergence, but a concerted effort to add QoS to IP was also getting underway. The seminal 1992 paper by Clark, Shenker, and Zhang on support of real-time applications in the Internet put a serious stake in the ground for IP QoS, and work at the IETF to standardize a set of IP QoS mechanisms began shortly thereafter. The Integrated Services architecture and Resource Reservation Protocol resulted, and the Differentiated Services architecture followed.

Another technical development with big implications for IP QoS was Multiprotocol Label Switching, which grew out of work on Tag Switching at Cisco begun in 1996. There was considerable confusion about exactly what impact MPLS would have on IP QoS, in part because of the resemblances between MPLS and ATM, which had its own QoS model. In reality, the biggest single effect MPLS had on QoS was to add another tool to the QoS toolbox, in the form of traffic engineering with constraint-based routing. It is for this reason more than any other that MPLS and QoS deserve to be covered in a single book.

Which brings us to the current volume. IP QoS can now be considered a mature technology, not just something for the bleeding edge. It is also notoriously complex to understand and to configure correctly. Some of this complexity is intrinsic; some is an accident of history. On the intrinsic side, understanding QoS is hard because it requires the ability to operate at many different levels of abstraction. One needs to understand the high level QoS architectures, to have a behavioral model of QoS features inside a router, to know how those features map onto a particular piece of hardware, and to understand the CLI that is used to control those features. This is where this book sets itself apart from the pack of QoS books. Some cover QoS architecture and IETF standards. Some provide information on CLI commands. But this is the only book I've found that walks the reader through the levels of abstraction from high level architecture to low level CLI, with a clear explanation of the abstract QoS behavior model that all routers support providing the bridge between the levels. By reading this book, you will understand both the big picture of QoS and the details necessary to deploy it in a real network.

Another factor that made QoS difficult to manage in the past was a somewhat ad hoc approach to its implementation. Combinations of features were sometimes implemented in a monolithic way, and inconsistency across platforms was the norm. This situation has improved massively in recent years, notably with the adoption of the Modular QoS CLI across most of the Cisco product line. Thus, QoS deployment is much more straightfoward than it once was, and this book's timely coverage of the MQC and its underlying behavioral model will make it even easier.

Many readers may be tempted to jump straight to the last chapter's guidance on how to design and deploy a QoS strategy in a backbone network. Santiago's extensive real-world deployment experience certainly makes this chapter especially valuable. However, the preceding four chapters are the ones that will provide you with a fundamental understanding of QoS. Thus, rather than blindly following a QoS "recipe," you'll be able to make the right design decisions to meet the needs of your own applications and customers. If you really want to understand QoS fully, this is the book to read, from start to finish.

Bruce Davie
Cisco Fellow

.

Introduction

The motivation behind this book is the continued interest in the implementation of *quality of service* (QoS) in IP/MPLS networks. QoS arises as a key requirement for these networks, which have become the preferred technology platform for building converged networks that support multiple services. The topic can be one of the most complex aspects of the network design, implementation, and operation. Despite the importance of and interest in this topic, no other Cisco Press title provides a detailed discussion of this subject. A significant amount of the content of this book also applies to pure IP networks that do not have immediate plans to migrate to a full IP/MPLS network.

This material covers both QoS and *Multiprotocol Label Switching Traffic Engineering* (MPLS TE). In particular, it covers MPLS TE as a technology that complements traditional QoS technologies. MPLS TE can be an instrumental tool to improve the QoS guarantees that an IP/MPLS network offers. As such, it can contribute to improving both network performance and availability. However, this book provides a concise discussion of MPLS TE. Those readers interested in further information should consult the Cisco Press title *Traffic Engineering with MPLS*.

The book takes the point of view of those individuals responsible for the IP/MPLS network. Other Cisco Press titles describe the details of the QoS implementation for those devices receiving the services that the network offers.

You should have a basic understanding of both IP and MPLS to obtain the most benefit from this book. That understanding should include basic IP addressing and routing, along with the basics of MPLS forwarding. However, the book provides a technology overview of QoS and MPLS TE to help those with less exposure to these technologies or to serve as a review/reference to those more familiar with those topics.

This book touches a broad topic and does not pretend to address all QoS aspects of interest. You can expect future Cisco Press books to cover important areas, including the following:

- Implementation of QoS for specific services (for instance, IP, Ethernet, ATM)
- QoS management (including monitoring and provisioning)
- Interprovider QoS

Visit this book's website, http://www.ciscopress.com/title/1587052334, for further information.

Who Should Read This Book?

This book's primary audience is the technical staff of those organizations building IP/MPLS networks as an infrastructure to provide multiple services. The material includes technology, configuration, and operational details to help in the design, implementation, and operation of QoS in IP/MPLS networks. Service providers are a prime example of the organizations that this book targets. However, government agencies, educational institutions, and large enterprises pursuing IP/MPLS will find the material equally useful.

A secondary audience for this book is those individuals in charge of service definition or those individuals subscribing to network services. Both types can benefit from a better understanding of the differentiation capabilities that IP/MPLS networks can offer.

How This Book Is Organized

Although this book could be read cover to cover, it is designed to be flexible and allow you to easily move between chapters and sections of chapters to cover just the material that you need more work with. The content is roughly divided into three parts:

- Chapters 1 and 2 provide a technology overview.
- Chapters 3 and 4 discuss Cisco implemenation.
- Chapter 5 covers different backbone design options.

Here is a brief synopsis of each chapter:

Chapter 1, "QoS Technology Overview"—This chapter provides a review of QoS technology for IP and IP/MPLS networks. The chapter initially discusses the IP QoS architectures and how they apply to MPLS. Multiple sections elaborate on MPLS support for *Differentiated Services* (DiffServ), including a detailed discussion on *EXP-inferred-class link switched path* (E-LSP), *Label-inferred-class LSP* (L-LSP), and DiffServ tunneling models (pipe, short pipe, and uniform). This dicussion leads into a summary of traffic-management mechanisms with a detailed look at traffic policing, traffic shaping, traffic scheduling, active queue manangemt, and so on. The chapter also discusses QoS signaling with a focus on the *Resource Reservation Protocol* (RSVP).

Chapter 2, "MPLS TE Technology Overview"—This chapter reviews the basic operation of this technology with its DiffServ extensions and applicability as a traffic-protection alternative. This review elaborates on the concepts of contraint-based routing, *DiffServ-aware Traffic Engineering* (DS-TE) and *fast reroute* (FRR) (including link, shared-risk link group, and node protection).

Chapter 3, "Cisco QoS"—This chapter covers the Cisco QoS behavioral model and the *modular QoS command-line interface* (MCQ). The chapter abstracts the platform specifics to facilitate the understanding of Cisco QoS and provides a complete reference of the configuration commands. In addition, the chapter includes numerous examples to illustrate the configuration and verification of different traffic-management mechanisms in Cisco IOS and Cisco IOS XR. This material is equially relevant to IP and IP/MPLS networks.

Chapter 4, "Cisco MPLS Traffic Engineering"—This chapter presents Cisco implementation of MPLS Traffic Engineering in both Cisco IOS and Cisco IOS XR. It includes multiple configuration and verification examples illustrating the implementation of basic MPLS TE, DS-TE, and FRR.

Chapter 5, "Backbone Infrastructure"—This chapter discusses the backbone performance requirements and the different design options. The chapter reviews different designs, ranging from a best-effort backbone to the most elaborate scenarios combining DiffServ, DS-TE, and FRR. Numerous configuration examples illustrate their implementation using Cisco IOS and Cisco IOS XR.

CHAPTER 1

QoS Technology Overview

In this chapter, you review the following topics:

- IP QoS Architectures
- MPLS Support for IntServ
- MPLS Support for DiffServ
- Traffic-Management Mechanisms
- QoS Signaling

This chapter provides a review of the key technology components of *quality of service* (QoS) in IP/MPLS networks. This review discusses the IntServ and DiffServ architectures including their relationship with MPLS. The chapter covers the traffic management mechanisms that enable QoS implementation and reviews different QoS signaling alternatives in IP/MPLS with a special focus on RSVP protocol.

This book assumes that you are already familiar with the basic concepts behind these topics. You should also be familiar with the basics of *Multiprotocol Label Switching* (MPLS) in general. This chapter and Chapter 2, "MPLS TE Technology Overview," serve as a technology review and quick reference for later content. Chapter 3, "Cisco QoS," covers the specifics on Cisco implementation of QoS technology. The "References" section at the end of this chapter lists sources of additional information on the topics that this chapter covers.

IP QoS Architectures

Originally, IP was specified a best-effort protocol. One of the implications of this service definition was that the network would attempt to deliver the traffic to its destination in the shortest time possible. However, the network would provide no guarantee of achieving it.

This service definition proved successful during the early Internet years, when data applications constituted the bulk of Internet traffic. Generally, these applications used TCP and therefore adapted gracefully to variations in bandwidth, latency, jitter, and loss. The amount of interactive traffic was minimal, and other applications requiring stricter guarantees were at an experimental stage.

However, a new generation of applications with new service requirements emerged as the Internet grew in success. The increasing reach and capacity of the Internet made it an attractive infrastructure to support an increasing number of applications. In addition, corporations, governments, and educational institutions, among others, found the IP protocol an appealing option to build their private data networks. Many of the new IP applications (for example, voice and video) had a real-time nature and limited tolerance to variations in bandwidth, latency, jitter, and loss. The service expectations of network users and their application requirements made the best-effort service definition insufficient.

The definition of a QoS architecture started in the middle of the 1990s. Since then, the *Internet Engineering Task Force* (IETF) has defined two QoS architectures for IP: *Integrated Services* (IntServ) and *Differentiated Services* (DiffServ). The IntServ architecture was the initial proposed solution. Subsequently, the DiffServ architecture came to life. MPLS later incorporated support for the DiffServ architecture, which the IETF had defined exclusively for IP.

These two architectures use different assumptions and take different approaches to bringing QoS to IP. Although sometimes considered opposite and competing architectures, they tend to complement each other. Moreover, the QoS mechanisms used ultimately to manipulate traffic are essentially the same in both architectures.

Integrated Services

The IntServ working group was responsible for developing the specifications of this architecture at the IETF. The group met for the first time during the twenty-ninth IETF in 1994. The architecture specifications have a close relationship with the work of the *IntServ over Specific Link Layers* (ISSLL) and the *Resource Reservation Protocol* (RSVP) working groups. The ISSLL working group defined the implementation of IntServ over different link-layer protocols (for example, Ethernet and ATM). The RSVP working group defined the RSVP protocol that the IntServ group selected as the signaling protocol. The three working groups collectively produced 32 RFCs, of which 24 are in the IETF standards track. The working groups eventually closed between the years 2000 and 2002.

The IETF decided to modify the original Internet architecture to support real-time applications. The IETF considered simpler alternatives, but they offered less-complete solutions. For instance

- Fair-queuing algorithms solved the unfairness between data and real-time applications, but they could not guarantee the delay and jitter.
- The use of separate networks for separate services was less efficient due to the lower levels of statistical multiplexing.
- Bandwidth overprovisioning was not a realistic solution when bandwidth was offered as a service.

- A simple priority mechanism could not prevent a growing number of real-time flows from causing degradation of all flows.
- The rate and delay adaptation of real-time applications had limits, especially when no admission control was used.

IntServ Terminology

This section lists several important terms that IntServ introduces. The next two sections provide more detail about these abstractions:

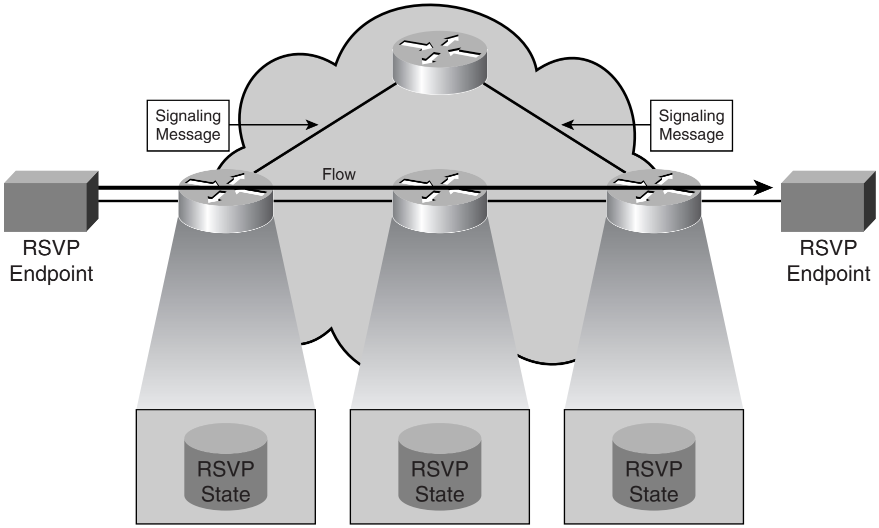
- **Flow**—An identifiable stream of packets that a network node associates with the same request for QoS. A flow may span a single or multiple application sessions.
- **Traffic specification (TSpec)**—Characterization of the traffic pattern of a flow over time.
- **Service request specification (RSpec)**—Characterization of the QoS a flow desires.
- **Flow specification (flowspec)**—Combination of a TSpec and an RSpec. Network nodes use the flowspec as input for admission-control decisions.

Architecture Principles

A crucial principle of the IntServ architecture is the requirement for resource reservation. This requirement implies admission control to manage finite resources. IntServ nodes need to avoid accepting unauthorized requests or requests that can affect existing reservations with service commitments. Different types of users are expected to have different rights to reserve network resources. In addition, the network load has to be controlled to meet the quantitative specification of the service-quality commitments of existing flows. IntServ leaves the selection of the QoS to the application rather than the network.

The architecture defines a flow as the basic service unit. This abstraction represents a distinguishable stream of packets that requires the same QoS. Flows are unidirectional. They have a single source and one or many destinations. IntServ requires the use of per-flow state in network nodes. This requirement results from the flow granularity and the use of resource reservation with admission control. Having network nodes maintaining per-flow state represents a significant change to the original IP architecture that left per-flow state to end systems. The architecture recommends the use of a signaling protocol to set up and refresh the state to preserve the robustness of the IP protocol. RFC 1633 introduces the architecture. Figure 1-1 shows a simple example of an IntServ network.

Figure 1-1 *Overview of a Network Implementing IntServ*



Service Model

The IntServ architecture defines an extensible service model with a common framework. An important component of the definition of a service is the information that the receiver, which requests the service, must specify to the network. A service request includes a TSpec and, possibly, an RSpec. When a service request is accepted, the network nodes must guarantee the service as long as the TSpec continues to describe the flow. The combination of a TSpec and an RSpec receives the name of flowspec.

The architecture service model uses a common TSpec definition. Four parameters characterize the traffic:

- **A token bucket (r, b)**—The token bucket includes a token rate (r) and a token bucket size (b).
- **A peak rate (p)**—Flow traffic may not arrive at a rate higher than the peak rate.
- **A minimum policed unit (m)**—Network nodes treat packets of a size smaller than the minimum policed unit as packets of size m. This term facilitates the estimation of the actual bandwidth that a flow requires (including the Layer 2 header overhead).
- **A maximum packet size (M)**—A node considers packets with a size larger than M as packets that do not conform to the traffic specification. Those nonconforming packets might not receive the same service as conforming packets.

Table 1-1 summarizes the TSpec parameters.

Table 1-1 *TSpec Parameters*

Parameter	Description
r	Token rate
b	Token bucket size
p	Peak rate
m	Minimum policed unit
M	Maximum packet size

NOTE The token bucket is an important concept in traffic management. The sections "Traffic Policing" and "Traffic Shaping" describe it in more detail later in this chapter.

The architecture defined two services: *Guaranteed Service* (GS) and *Controlled Load Service* (CLS). They complement the best-effort service that is part of the definition of the IP protocol. IntServ does not introduce any changes to the operation of the best-effort service. In addition, the IntServ service model does not mandate a particular implementation for the traffic-management mechanisms that implement a service. The next two sections explain the GS and CLS services, focusing on their end-to-end behavior and their flow specifications.

NOTE RFC 2997 later defined a Null Service type. Applications can use this service to let the network determine the appropriate service parameters for the flow. This service type has special applicability for the integration of IntServ and DiffServ architectures.

Guaranteed Service

GS provides flows with a delay and bandwidth guarantee. GS ensures a firm bound on the maximum end-to-end queuing delay for packets that conform to the flowspec. Furthermore, a properly policed flow should not experience queuing drops in the absence of network failures or routing changes. The service does not consider fixed-delay components that are a property of the flow path (for example, propagation delay or serialization delay). A flow receives guaranteed service if all nodes along the path support the service. GS does not guarantee an average or minimum delay, just a maximum bound. Therefore, this service does not provide any jitter guarantees. RFC 2212 defines GS.

A receiver provides a TSpec and an RSpec when requesting GS. The RSpec contains a *service rate* (R) and a *time slack* (S). Network nodes must approximate the service that a dedicated line at that rate would provide to the flow. Nodes make available the margin of error of their approximation. Applications can use this information to compute the maximum end-to-end delay that the flow will experience. The slack term in the RSpec specifies the incremental end-to-end delay that the sender can tolerate if a node modifies the flow resource allocation. Applications can adjust the flowspec if the delay bound is not acceptable. Table 1-2 summarizes the RSpec parameters.

Table 1-2 *RSpec Parameters*

Parameter	Description
R	Service rate
S	Time slack

Control Load Service

CLS approximates the behavior of best-effort service during unloaded conditions. Network nodes satisfy this behavior even in the presence of congestion. Applications can assume that the network will deliver a high percentage of all packets to their final destination. In addition, applications can assume that a high percentage of the delivered packets will experience a delay that will not greatly exceed the minimum delay of any packet. Applications do not receive any target values for packet delay or loss. This service supports those applications that operate satisfactorily with a best-effort service but are highly sensitive to congestion conditions. Applications do not require an RSpec to request CLS, only the flow TSpec. RFC 2211 introduces CLS.

Use of RSVP in IntServ

IntServ can use RSVP as the reservation setup protocol. One of the principles of this architecture is that applications communicate QoS requirements for individual flows to the network. These requirements are used for resource reservation and admission control. RSVP can perform this function. However, RSVP is frequently but inaccurately equated to IntServ. RSVP and IntServ share a common history, but they are ultimately independent. Two separate working groups at the IETF developed their specifications. RSVP has applicability as a signaling protocol outside IntServ. Similarly, IntServ could use other signaling mechanisms. The section "Resource Reservation Protocol" explains the protocol details later in this chapter. RFC 2210 describes how IntServ uses RSVP and defines the RSVP objects to implement the IntServ service model.

Differentiated Services

The DiffServ working group was responsible for the definition of this architecture at the IETF. The group met for the first time during the forty-first IETF in 1998. The working group was created with a charter to produce an architecture with a simple and coarse QoS approach that applied to both IPv4 and IPv6. The charter explicitly excluded microflow identification and signaling mechanisms (marking an explicit departure from the approach taken by IntServ). The working group produced 12 RFCs, with five of them being in the standards track and the rest being informational. The group eventually closed after its last meeting in 2001.

NOTE An IP traffic stream with a unique combination of source address, destination address, protocol, source port, and destination port defines a microflow.

DiffServ Terminology

The DiffServ architecture introduces many new terms. This section presents a simplified definition of a selected few of them. The upcoming sections explain the terms in more detail. RFC 2475 and 3260 introduce the complete list of terms:

- **Domain**—A network with a common DiffServ implementation (usually under the same administrative control).
- **Region**—A group of contiguous DiffServ domains.
- **Egress node**—Last node traversed by a packet before leaving a DiffServ domain.
- **Ingress node**—First node traversed by a packet when entering a DiffServ domain.
- **Interior node**—Node in a DiffServ domain that is not an egress or ingress node.
- **DiffServ field**—Header field where packets carry their DiffServ marking. This field corresponds to the six most significant bits of the second byte in the IP header (formerly, IPv4 TOS [*Type-of-Service*] octet and IPv6 Traffic Class octet).
- **Differentiated Services Code Point (DSCP)**—A specific value assigned to the DiffServ field.
- **Behavior aggregate (BA)**—Collection of packets traversing a DiffServ node with the same DSCP.
- **Ordered aggregate (OA)**—A set of BAs for which a DiffServ node must guarantee not to reorder packets.
- **BA classifier**—Classifier that selects packets based on DSCP.
- **Multifield (MF) classifier**—Classifier that selects a packet based on multiple fields in the packet header (for example, source address, destination address, protocol, and protocol port).

- **Per-hop behavior (PHB)**—Forwarding behavior or service that a BA receives at a node.
- **Per-hop behavior group**—One or more PHBs that are implemented simultaneously and define a set of related forwarding behaviors.
- **PHB scheduling class (PSC)**—A set of PHBs for which a DiffServ node must guarantee not to reorder packets.
- **Traffic profile**—Description of a traffic pattern over time. Generally, in terms of a token bucket (rate and burst).
- **Marking**—Setting the DSCP in a packet.
- **Metering**—Measuring of a traffic profile over time.
- **Policing**—Discarding of packet to enforce conformance to a traffic profile.
- **Shaping**—Buffering of packets to enforce conformance to a traffic profile.
- **Service level agreement (SLA)**—Parameters that describe a service contract between a DiffServ domain and a domain customer.
- **Traffic-conditioning specification**—Parameters that implement a service level specification.
- **Traffic conditioning**—The process of enforcing a traffic conditioning specification through control functions such as marking, metering, policing, and shaping.

NOTE This DiffServ section is consistent with the original architecture terms. Note, however, that Cisco documentation considers metering an implicit component of traffic shaping and policing. Furthermore, it regards dropping as just one of three possible actions (transmit, drop, mark) of a policer. The remainder of the book follows the policing and shaping definitions used in Cisco documentation.

Architecture Principles

The DiffServ architecture relies on the definition of classes of traffic with different service requirements. A marking in the packet header captures the traffic classification. Further network nodes inspect this marking to identify the packet class and allocate network resources according to locally defined service policies. The service characteristics are unidirectional with a qualitative description in terms of latency, jitter, and loss. DiffServ nodes are stateless from a QoS point of view and have no knowledge of individual flows. Relatively few packet markings are possible with respect to the number of microflows that a node may be switching at a given point in time. However, the concept of grouping or aggregating traffic into a small number of classes is inherent to DiffServ. The architecture intentionally makes a tradeoff between granularity and scalability. RFC 2475 introduces the architecture.

Providing different levels of service using aggregate classification and marking is not a novel concept. As an analogy, consider a transoceanic commercial flight with multiple classes of service. During the check-in process, the passenger needs to provide some identification information (classification criteria). Based on this information, the agent identifies the passenger class (for instance, first, business, or tourist) and provides a boarding pass that reflects the assigned class (marking). The customer class influences the service the passenger receives during the duration of the flight (including access to airline lounge, boarding priority, in-flight service, and deboarding priority). The reduced number of classes allows the airline to provide some differentiation to customers without having to provide a totally individualized service to each passenger. As you can probably recognize, this is not the only instance of aggregate classification and marking used in real life.

Differentiated Services Code Point

Previous specifications of the IP protocol had already reserved header bits for QoS purposes. The IP version 4 specifications in RFC 791 defined the second header octet as the TOS octet. Also, the IP version 6 specifications in RFC 2460 defined the second header octet as the Traffic Class octet but with an undefined structure. In the original TOS octet in IP version 4, the three most significant bits specified the packet precedence (an indication of the relative importance or priority). The next 3 bits in the TOS octet indicated the delay, throughput, and reliability requirements. The final two (least significant) bits were undefined and set to zero. RFC 1349 introduced a small change in the TOS octet by defining a field that included a cost bit plus the existing delay, throughput, and reliability bits. Similarly, the IP version 6 specifications in RFC 2460 define the second header octet as the Traffic Class octet but with an undefined structure. Figure 1-2 illustrates these now-obsolete definitions.

The DiffServ architecture redefines the IPv4 TOS octet and the IPv6 Traffic Class octet. RFC 2474 and RFC 3260 name the DiffServ field as the six most significant bits in the previous IPv4 TOS and IPv6 Traffic Class octets. A particular value of the DiffServ field represents a DSCP. A DiffServ node services packets according to this code point. A group of packets sharing the same DSCP and traversing a link in a specific direction constitutes a BA. A class of traffic may include one or more BAs.

The architecture defines three code point pools for the DiffServ field. Two pools, representing 32 out of the 64 possible values, are reserved for experimental or local use. The existing DiffServ specifications provide recommendations for 21 out of the 32 code points available in the third pool.

The section "Per-Hop Behaviors" in this chapter introduces these values and their service definitions. Eight (class selector) code points provide backward compatibility with the previous Precedence field in the TOS octet. The DiffServ field does not provide any backward compatibility with the TOS field (delay, throughput, and reliability bits) previously defined in the TOS octet. Figure 1-3 shows the structure of the new DiffServ field, the code point pools, and the class selector code points.

Figure 1-2 *Previous Definitions of the TOS Octet for IPv4 and IPv6 That DiffServ Makes Obsolete*

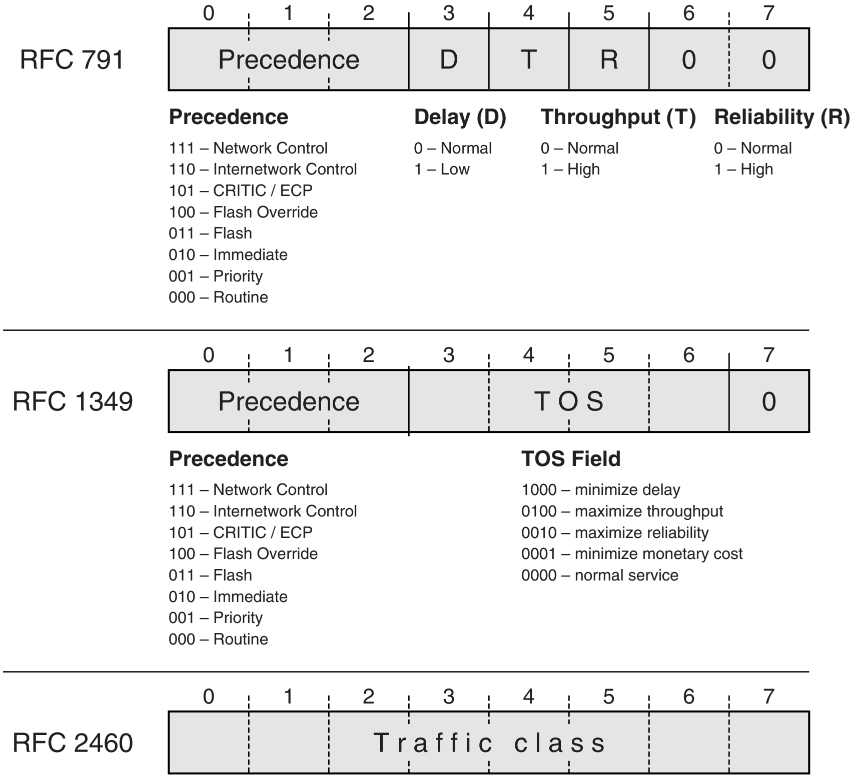
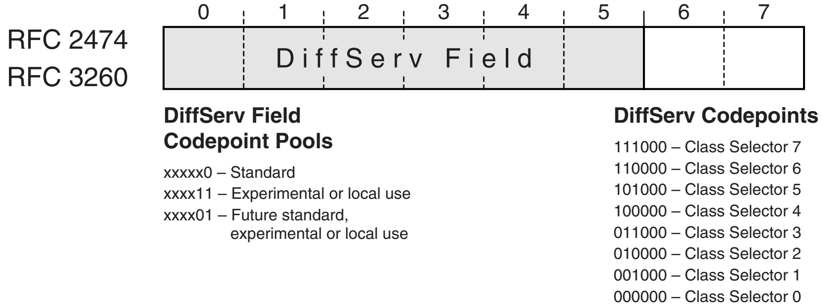


Figure 1-3 *DiffServ Field, Code Point Pools, and Class Selector Code Points*



Nodes, Domains, and Regions

The DiffServ architecture defines a hierarchy that goes from a single device, to a network, to a group of networks. A set of nodes with a common DiffServ implementation forms a domain. The nodes inside a domain perform similar service definitions and policies. A domain is typically under a single administrative control. A set of contiguous domains defines a DiffServ region. The domains within the region must be able to provide DiffServ to traffic traversing the different domains in the region. Individual domains may use different service definitions, policies, and packet markings. In those cases, the domains must have peering agreements that specify how traffic is handled when crossing domains.

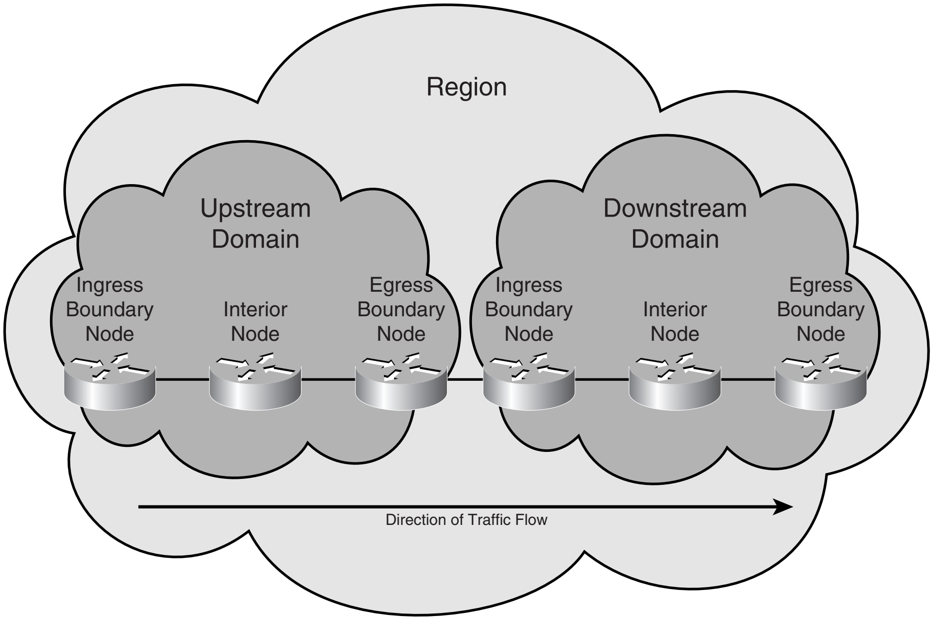
Boundary nodes and interior nodes constitute the two main types of nodes that reside in a DiffServ domain. Boundary nodes interface with the outside of the domain and ensure that any traffic is properly classified, marked, and within the agreed amounts. DiffServ defines these operations as traffic classification and conditioning. Boundary and interior nodes implement local service policies according to the packet marking. These local policies provide different levels of service to each BA that DiffServ calls PHBs. The section "Per-Hop Behaviors" discusses this concept in detail. A domain may have some nodes that do not support DiffServ. The service impact of these nodes depends on their number and their location within a domain. Figure 1-4 illustrates a DiffServ region with two domains.

Traffic enters a domain at an ingress boundary node and leaves the domain at an egress boundary node. The ingress boundary node typically performs the traffic-classification and conditioning function according to a specification or contract. Boundary nodes generally act as both ingress and egress nodes because traffic differentiation is desirable for the traffic that flows in both directions.

Traffic Classification and Conditioning

Traffic classification and conditioning identifies the traffic that will receive a differentiated service and ensures that it conforms to a service contract. Outside nodes connecting to the DiffServ domain have agreed to some service terms that the architecture defines as an SLA. The boundary node enforces this SLA, or contract, using traffic classification and conditioning. This enforcement uses a combination of packet classification, marking, metering, shaping, and policing to ensure that the traffic conforms to the contract terms.

Figure 1-4 *Functional View of Nodes and Domains in a DiffServ Region*



NOTE RFC 3260 refined the initial DiffServ definition of SLA and introduced a new term, *service level specification* (SLS). Even though the term clarification is useful, this book uses the term SLA (which is more commonly used).

Traffic classification is the first action that a boundary node performs on traffic entering the domain. The boundary node examines the packet and, according to the SLA terms, implements an appropriate action (for example, marking, metering, policing). The architecture describes two types of classifiers: BA classifier and MF classifier. The BA classifier classifies packets using the DSCP in the packet. The MF classifier classifies packets using one or more fields of the packet header (for example, source IP address, destination IP address, protocol, source port, destination port) and other packet information (for example, input interface). The final result of packet classification is the local association of each packet with a class.

NOTE The DiffServ architecture did not consider packet payload as input for packet classification. Actual implementations support packet classification using payload inspection and other classification criteria. You can consider those classifiers as an extension of the MF classifiers.