- Table of Contents
- Index

Cisco Catalyst QoS: Quality of Service in Campus Networks

By Mike Flannagan CCIE® No. 7651, Richard Froom CCIE No. 5102, Kevin Turek CCIE No. 7284

Publisher: Cisco Press
Pub Date: June 06, 2003
ISBN: 1-58705-120-6
Pages: 432

End-to-end QoS deployment techniques for Cisco Catalyst series switches

- Examine various QoS components, including congestion management, congestion avoidance, shaping, policing/admission control, signaling, link efficiency mechanisms, and classification and marking
- Map specified class of service (CoS) values to various queues and maintain CoS values through the use of 802.1q tagging on the Cisco Catalyst 2900XL, 3500XL and Catalyst 4000 and 2948G/2980G CatOS Family of Switches
- Learn about classification and rewrite capabilities and queue scheduling on the Cisco Catalyst 5000
- Implement ACLs, ACPs, ACEs, and low-latency queuing on the Cisco Catalyst 2950 and 3550 Family of Switches
- Understand classification, policying, and scheduling capabilities of the Catalyst 4000 and 4500 IOS Family of Switches
- Configure QoS in both Hybrid and Native mode on the Catalyst 6500 Family of Switches
- Utilize Layer 3 QoS to classify varying levels of service with the Catalyst 6500 MSFC and Flexwan
- Understand how to apply QoS in campus network designs by examining end-to-end case studies

Quality of service (QoS) is the set of techniques designed to manage network resources. QoS refers to the capability of a network to provide better service to selected network traffic over various LAN and WAN technologies. The primary goal of QoS is to provide flow priority, including dedicated bandwidth, controlled jitter and latency (required by some interactive and delay-sensitive traffic), and improved loss characteristics.

While QoS has become an essential technology for those organizations rolling out a new

generation of network applications such as real-time voice communications and high-quality video delivery, most of the literature available on this foundation technology for current and future business applications focuses on IP QoS. Equally important is the application of QoS in the campus LAN environment, which is primarily responsible for delivering traffic to the desktop.

Cisco Catalyst QoS is the first book to concentrate exclusively on the application of QoS in the campus environment. This practical guide provides you with insight into the operation of QoS on the most popular and widely deployed LAN devices: the Cisco Catalyst family of switches. Leveraging the authors' extensive expertise at Cisco in the support of Cisco Catalyst switches and QoS deployment, the book presents QoS from the campus LAN perspective. It explains why QoS is essential in this environment in order to achieve a more deterministic behavior for traffic when implementing voice, video, or other delay-sensitive applications. Through architectural overviews, configuration examples, real-world deployment case studies, and summaries of common pitfalls, you will understand how QoS operates, the different components involved in making QoS possible, and how QoS can be implemented on the various Cisco Catalyst platforms to enable truly successful end-to-end QoS applications.

This book is part of the Networking Technology Series from Cisco Press, which offers networking professionals valuable information for constructing efficient networks, understanding new technologies, and building successful careers.

- Table of Contents
- Index

Cisco Catalyst QoS: Quality of Service in Campus Networks

By Mike Flannagan CCIE® No. 7651, Richard Froom CCIE No. 5102, Kevin Turek CCIE No. 7284

Publisher: Cisco Press
Pub Date: June 06, 2003
ISBN: 1-58705-120-6
Pages: 432

Copyright

Published by:
Cisco Press
201 West 103rd Street
Indianapolis, IN 46290 USA

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

Library of Congress Cataloging-in-Publication Number: 2002109166

Warning and Disclaimer

This book is designed to provide information about *quality of service* (QoS) for the Cisco Catalyst switch platform. 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.

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.

Trademark Acknowledgments

All terms mentioned in this book that are known to be trademarks or service marks have been

Publisher	John Wait
Editor-In-Chief	John Kane
Cisco Representative	Anthony Wolfenden
Cisco Press Program Manager	Sonia Torres Chavez
Cisco Marketing Communications Manager	Scott Miller
Cisco Marketing Program Manager	Edie Quiroz
Executive Editor	Brett Bartow
Production Manager	Patrick Kanouse
Development Editor	Jennifer Foster
Copy Editor	Keith Cline
Technical Editors	Jason Cornett
	Lauren Dygowski
	Balaji Sivasubramanian
Team Coordinator	Tammi Ross
Book Designer	Gina Rexrode
Cover Designer	Louisa Adair
Compositor	Mark Shirar
Indexer	Tim Wright
Proofreader	Missy Pluta

Corporate Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 526-4100

European Headquarters
Cisco Systems International BV
Haarlerbergpark
Haarlerbergweg 13-19

1101 CH Amsterdam
The Netherlands
www-europe.cisco.com
Tel: 31 0 20 357 1000
Fax: 31 0 20 357 1100

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
www.cisco.com
Tel: 408 526-7660
Fax: 408 527-0883

Asia Pacific Headquarters
Cisco Systems, Inc.
Capital Tower
168 Robinson Road
#22-01 to #29-01
Singapore 068912
www.cisco.com
Tel: +65 6317 7777
Fax: +65 6317 7799

Cisco Systems has more than 200 offices in the following countries and regions. Addresses, phone numbers, and fax numbers are listed on the Cisco.comWeb site at www.cisco.com/go/offices.

Argentina • Australia • Austria • Belgium • Brazil • Bulgaria • Canada • Chile • China PRC • Colombia • Costa Rica • Croatia • Czech Republic • Denmark • Dubai, UAE • Finland • France • Germany • Greece • Hong Kong SAR • Hungary • India • Indonesia • Ireland • Israel • Italy • Japan • Korea • Luxembourg • Malaysia • Mexico • The Netherlands • New Zealand • Norway • Peru • Philippines • Poland • Portugal • Puerto Rico • Romania • Russia • Saudi Arabia • Scotland • Singapore • Slovakia • Slovenia • South Africa • Spain • Sweden • Switzerland • Taiwan • Thailand • Turkey • Ukraine • United Kingdom • United States • Venezuela • Vietnam • Zimbabwe

Printed in the USA

Dedications

Mike Flannagan:

I would like to dedicate this book to Anne. Thank you for your loving support and encouragement during the seemingly endless nights and weekends spent on this project.

Richard Froom:

I would like to dedicate this book to my wife Elizabeth for her support, understanding, and patience while I was authoring the book. I would also like to thank Elizabeth for her cooperation and assistance in reviewing my material.

Kevin Turek:

I would like to thank Tara for all of her encouragement and patience during my involvement in this project. I would not have been able to complete my portion of the book if not for your sacrifices and understanding. I would also like to thank the members of the Federal Support Program for their support. Finally, I would like to thank my family and friends for instilling in me the values and ethics that have allowed me to get to the point where I am today.

About the Authors

Mike Flannagan, CCIE No. 7651, is a manager in the High Touch Technical Support (HTTS) group at Cisco Systems in Research Triangle Park, North Carolina. Mike joined Cisco in 2000 as a network consulting engineer in the Cisco Advanced Services group, where he led the creation and deployment of the QoS Virtual Team. As a member of the QoS Virtual Team, Mike was involved with the development of new QoS features for Cisco IOS and developed QoS strategies and implementation guidelines for some of the largest Cisco enterprise customers. Mike teaches QoS classes and leads design clinics for internal and external audiences and is the author of *Administering Cisco QoS for IP Networks*, published by Syngress.

Richard Froom, CCIE No. 5102, is a software and QA engineer for the Financial Test Lab at Cisco Systems in Research Triangle Park. Richard joined Cisco in 1998 as a customer support engineer in the Cisco Technical Assistance Organization. Richard, as a customer support engineer, served as a support engineer troubleshooting customers' networks and as a technical team lead. Being involved with Catalyst product field trials, Richard has been crucial in driving troubleshooting capabilities of Catalyst products and software. Currently, Richard is working with the Cisco storage networking products. Richard earned his bachelor of science degree in computer engineering at Clemson University.

Kevin Turek, CCIE No. 7284, is currently working as a network consulting engineer in the Cisco Federal Support Program in Research Triangle Park. He currently supports some of the Cisco Department of Defense customers. Kevin is also a member of the internal Cisco QoS virtual team, supporting both internal Cisco engineers and external Cisco customers with QoS deployment and promoting current industry best practices as they pertain to QoS. Kevin earned his bachelor of science degree in business administration at the State University of New York, Stony Brook.

About the Technical Reviewers

Jason Cornett is a customer support engineer at Cisco Systems, where he is a technical leader for the LAN Switching team in Research Triangle Park. Jason joined Cisco in 1999 and has a total of five years of networking experience. He holds a diploma in business technology information communications systems from St. Lawrence College. Previously Jason was a network support specialist with a network management company.

Lauren L. Dygowski, CCIE No. 7068, is a senior network engineer at a major financial institution and has more than eight years of networking experience. He holds a bachelor of science degree in computer science from Texas Tech University and a M.S.B.A. from Boston University. Previously, Lauren was a network manager with the United States Marine Corps and MCI. He resides with his wife and three sons in Charlotte, North Carolina.

Balaji Sivasubramanian is part of the Technical Assistance Center (TAC) based out of Research Triangle Park, where he acts as the worldwide subject matter expert in LAN technologies. He has been with Cisco TAC for more than three years. He has authored and reviewed many technical white papers on Cisco.com in the LAN technologies area. He has been a presenter/moderator of the Technical Virtual Chalk Talk Seminars for Partners. He has actively participated in early field trial testing of the Catalyst 4000/4500 platform series. Balaji holds a master of science degree in computer engineering from the University of Arizona and holds a bachelor of science degree in electrical engineering.

Acknowledgments

Richard would like to acknowledge his co-authors for asking him to work on this project and their efforts on their material. Furthermore, Richard would like to acknowledge Balaji Sivasubramanian for his aid and outstanding reviews of technical content pertaining to the Catalyst 4000 Family of switches.

Mike would like to acknowledge Dave Knuth and his other friends in Network Optimization Support for their support. Mike would especially like to thank Chris Camplejohn, Nimish Desai, and Zahoor Khan for asking the tough questions that always helped me learn. Thanks to the members of the AS QoS virtual team for their outstanding teamwork and technical expertise in support of our customers. Mike would especially like to recognize Richard Watts for his support of and contributions to the team. Finally, thanks to my new team in Cisco's HTTS group for an exciting new opportunity to learn.

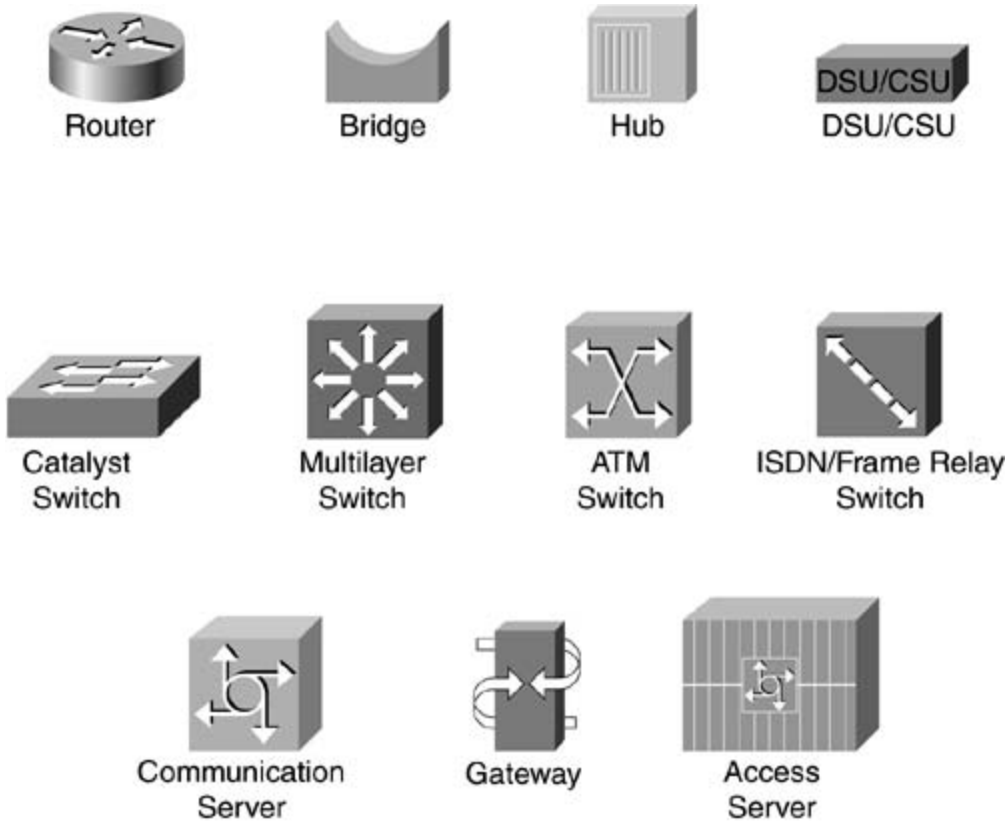
Kevin would like to personally thank his co-authors for agreeing to do this book, and for their motivation, professionalism, and expertise, which ensured its timely completion without sacrificing quality. Thanks especially to Richard for his focus and acceptance of an additional load, despite his already busy schedule.

All the authors would like to thank Jeff Raymond for his time and recommendations that contributed to the technical accuracy and quality of the content pertaining to the Catalyst 6500 Family of switches.

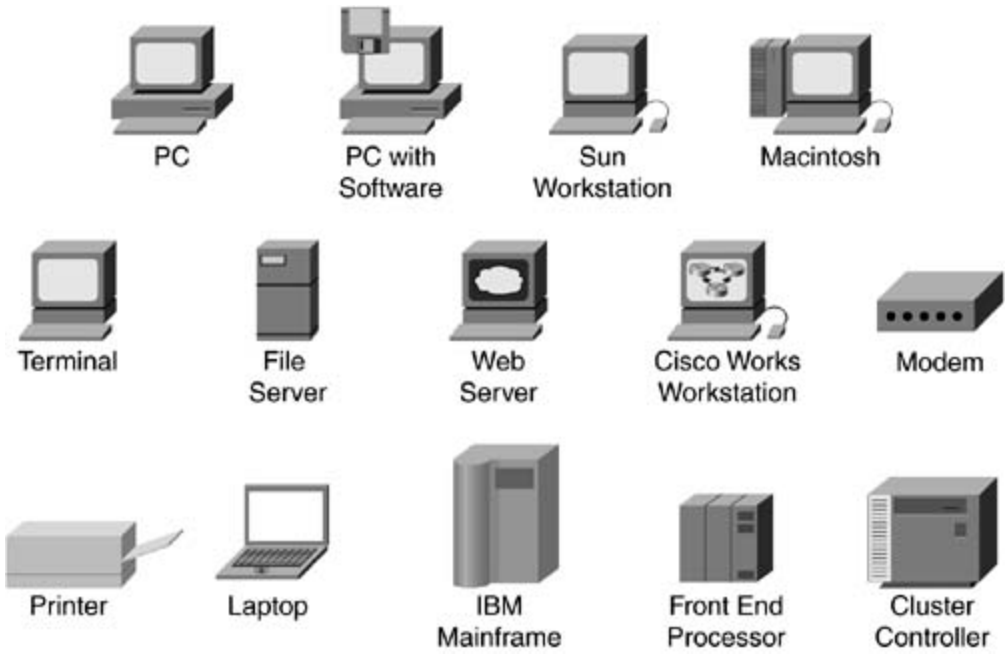
A big thank you goes to the team at Cisco Press, especially Brett Bartow for his overall support of this project and Christopher Cleveland and Jennifer Foster for their suggestions and input. We would like to give special thanks to our review team of Jason, Lauren, and Balaji for their dedication to making this project successful.

Icons Used in This Book

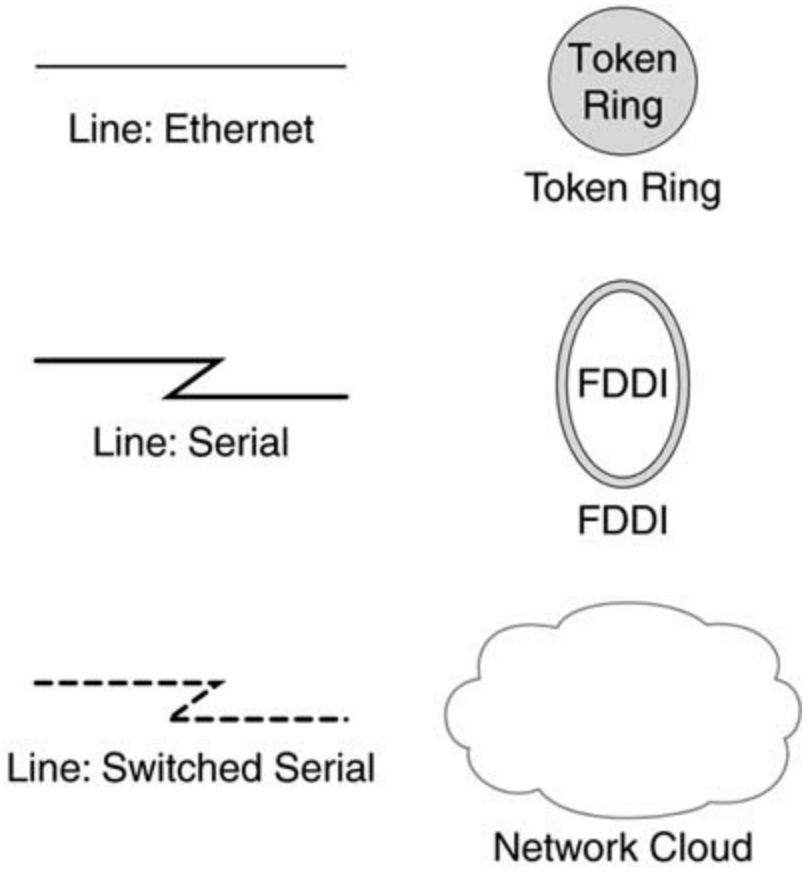
Throughout this book, you will see the following icons used for networking devices:



The following icons are used for peripherals and other devices:



The following icons are used for networks and network connections:



Command Syntax Conventions

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

- Vertical bars (|) separate alternative, mutually exclusive elements.
- Square brackets [ ] indicate optional elements.
- Braces { } indicate a required choice.
- Braces within brackets [{ }] indicate a required choice within an optional element.
- 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.

Introduction

This book is intended for all network engineers who deal with Catalyst switches and specifically for those looking for a deeper understanding of the QoS capabilities of those switches. In addition, any network engineer responsible for end-to-end QoS policies in a Cisco network will find, either now or in the near future, the need to thoroughly understand Catalyst QoS.

Besides the configuration syntax information provided in this book, the authors have made every attempt to discuss common practices and provide case studies. This treatment of the subject matter provides readers with more than just commands for configuring QoS on Catalyst switches; specifically, the authors wanted to make sure that readers understand the reasons for certain policy decisions, as those decisions would relate to a production environment.

Motivation for This Book

After countless hours spent trying to locate various pieces of information for our customers, the authors realized that there was not a good Catalyst QoS book anywhere to be found. Rather than continue to answer the same questions, we decided to publish our collection of the most commonly requested information plus some not-so-common information that would give readers a strong foundation in Catalyst QoS.

Goal of This Book

The purpose of this book is to provide readers who have a curiosity about Catalyst QoS, and end-to-end QoS involving Catalyst switches, with a well-rounded baseline of information about the RFCs involved in QoS, the configuration steps for enabling QoS, and the command syntax for fine-tuning the operation of QoS on Catalyst switches. In addition, the authors want to make sure that our readers walk away with more than command syntax; after completing this book, readers should actually be prepared to deploy Catalyst QoS in a production environment.

Prerequisites

Although anyone can read this book, the authors assume reader understanding of some fundamental networking concepts discussed in this book. If you do not have basic knowledge in the following areas, you might have trouble understanding certain examples and concepts presented in this text:

- Cisco IOS—Basic syntax
- Cisco Catalyst OS (CatOS)—Basic syntax
- Access-control list (ACL) configuration
- Virtual LANs (VLANs)
- IP addressing
- Routing protocols—Basic syntax and concepts
- TCP/UDP port assignments

How This Book Is Organized

Although this book could be read cover to cover, it is designed to be flexible and enable you to move easily between chapters and sections of chapters to cover just the material that you need more work with. Each chapter stands by itself; however, most chapters reference earlier chapter material. Overall, therefore, the order in the book is an excellent sequence to follow.

The material covered in this book is as follows:

- Chapter 1, Quality of Service: An Overview— This chapter defines quality of service (QoS), as it pertains to Cisco networks, provides a general overview of the necessity for QoS in multiservice networks, and discusses various QoS models. Several of the key RFCs pertaining to IP QoS are also explored in this chapter.

- Chapter 2, End-to-End QoS: Quality of Service at Layer 3 and Layer 2— This chapter explores QoS components such as congestion management, congestion avoidance, traffic conditioning, and link efficiency. This chapter also explores per-hop behaviors in the differentiated services architecture and includes an entry-level discussion of QoS on Catalyst platforms. The Catalyst platform discussion also cotains discussions of the Catalyst voice VLAN and trust concept.

- Chapter 3, Overview of QoS Support on Catalyst Platforms and Exploring QoS on the Catalyst 2900XL, 3500XL, and Catalyst 4000 CatOS Family of Switches— This chapter provides a basic overview of QoS support on each platform. In addition, a detailed explanation of QoS feature supported is provided for the Catalyst 2900XL, 3500XL, and Catalyst 4000 CatOS switches.

- Chapter 4, QoS Support on the Catalyst 5000 Family of Switches— This chapter discusses the limited hardware and software feature support for QoS on the Catalyst 5000 family of switches. In addition, concepts such as multilayer switches are explained in this chapter.

- Chapter 5, Introduction to the Modular QoS Command-Line Interface— This chapter discusses the need for the MQC and the steps required to configure QoS mechanisms using the MQC. In addition to providing an explanation of the commands necessary for configuration, this chapter also provides sample show command output for the various commands needed to verify the functionality of the configuration.

- Chapter 6, QoS Features Available on the Catalyst 2950 and 3550 Family of Switches— This chapter covers QoS feature support on both the Catalyst 2950 and 3550 family of switches. The QoS examples in this chapter present these switches as access layer switches. This chapter also includes an Auto-QoS discussion for those switches applicable to Voice over IP.

- Chapter 7, QoS Features Available on the Catalyst 4000 IOS Family of Switches and the Catalyst G-L3 Family of Switches— This chapter covers QoS feature support on the Catalyst 4000 IOS family of switches and the Catalyst G-L3 switches. The Catalyst G-L3 switches include the Catalyst 2948G-L3, 4908G-L3, and the WS-X4232-L3 Layer 3 services module for the Catalyst 4000 CatOS family of switches.

- Chapter 8, QoS Support on the Catalyst 6500— This chapter focuses on the QoS architecture for the Catalyst 6500 series platform. Specifically, it demonstrates how QoS on the Catalyst 6500 can support voice and other mission-critical applications in a converged environment. The chapter further demonstrates configuring QoS features using both CatOS

and Cisco IOS.

- Chapter 9, QoS Support on the Catalyst 6500 MSFC and FlexWAN— This chapter discusses the QoS capabilities of the FlexWAN and MSFC in the Catalyst 6500. The chapter shows how the FlexWAN and MSFC extend the Catalyst 6500's QoS capabilities to the MAN and WAN. Examples demonstrate configuring QoS using the MQC available in Cisco IOS.

- Chapter 10, End-to-End QoS Case Studies— This chapter presents end-to-end QoS case studies using a typical campus network design. The network topology illustrates QoS end-to-end using the Catalyst 2950, 3550, 4500 IOS, 6500 switches.

Part I: Fundamental QoS Concepts

Chapter 1. Quality of Service: An Overview

The constantly changing needs of networks have created a demand for sensitive applications (such as *voice over IP* (VoIP) and video conferencing over IP), and networks are being asked to support increasingly mission-critical data traffic. Providing predictable service levels for all of these different types of traffic has become an important task for network administrators. Being able to provide predictable and differentiated service levels is key to ensuring that all application traffic receives the treatment that it requires to function properly.

This chapter covers several aspects of *quality of service* (QoS) and discusses how to provide QoS in Cisco networks. Specifically, this chapter covers the following topics:

- Understanding QoS
- Deploying QoS in the WAN/LAN: High-Level Overview
- Cisco AVVID (Architecture for Voice, Video, and Integrated Data)
- Overview of Integrated Services and Differentiated Services
- Differentiated Services: A Standards Approach

Understanding QoS

The following section defines QoS in terms of measurable characteristics. It is important, however, to recognize that fully understanding QoS requires more than a definition. To truly understand QoS, you must understand the concept of managed unfairness, the necessity for predictability, and the goals of QoS. In addition to a definition of QoS in measurable terms, the following section explains each of these things, to provide you with a well-rounded and practical definition of QoS

Definition of QoS

QoS is defined in several ways, and the combination of all of these definitions is really the best definition of all. A technical definition is that *QoS* is a set of techniques to manage bandwidth, delay, jitter, and packets loss for flows in a network. The purpose of every QoS mechanism is to influence at least one of these four characteristics and, in some cases, all four of these

Bandwidth

Bandwidth itself is defined as the rated throughput capacity of a given network medium or protocol. In the case of QoS, bandwidth more specifically means the allocation of bandwidth, because QoS does not have the capability to influence the actual capacity of any given link. That is to say that no QoS mechanism actually creates additional bandwidth, rather QoS mechanisms enable the administrator to more efficiently utilize the existing bandwidth. Bandwidth is sometimes also referred to as *throughput*.

Delay

Delay has several possible meanings, but when discussing QoS, *processing delay* is the time between when a device receives a frame and when that frame is forwarded out of the destination port,*serialization delay* is the time that it take to actually transmit a packet or frame, and *end-to-end delay* is the total delay that a packet experiences from source to destination.

Jitter

Jitter is the difference between interpacket arrival and departure—that is, the variation in delay from one packet to another.

Packet Loss

Packet loss is just losing packets along the forwarding path. Packet loss results from many causes, such as buffer congestion, line errors, or even QoS mechanisms that intentionally drop packets.

Table 1-1 shows examples of the varying requirements of common applications for bandwidth, delay, jitter, and packet loss.

Table 1-1. Traffic Requirements of Common Applications

	Voice	FTP	Telnet
Bandwidth Required	Low to moderate	Moderate to High	Low
Drop Sensitive	Low	Low	Moderate
Delay Sensitive	High	Low	Moderate
Jitter	High	Low	Moderate

Managed Unfairness

Another, more pragmatic definition of QoS is *managed unfairness.* The best way to explain this definition is using an analogy to airline service. Sometimes airlines are unable to sell all of their seats in first class, but they rarely leave the gate with first class seats available, so there has to be some method by which they decide who gets those seats. The gate agent could swing open the door to the plane and tell everyone to rush onto the plane; whoever gets to the first class seats first gets them. Some would argue that would be the most *fair* way to handle the seating, but that would be very disorderly and probably not a very pleasant thing to watch. Anyone who flies regularly knows that frequent flier miles are valuable because you can earn free flights and so on. If you collect enough frequent flier miles from a specific airline in a single year, however, you will be moved into an *elite* frequent flier status and get some extra benefits. One of those benefits is typically some method by which the most frequent fliers are able to upgrade their coach seat to a first class seat, when available. Imagine paying full price for a coach ticket to Hawaii, and having no chance at all to upgrade, while the person beside you is able to upgrade just because he is a frequent flier. Some would argue that this is *unfair.* However, it is unfair in a very controlled way, because there is a specific policy in place that dictates who is eligible for this upgrade and who is not. This is managed unfairness.

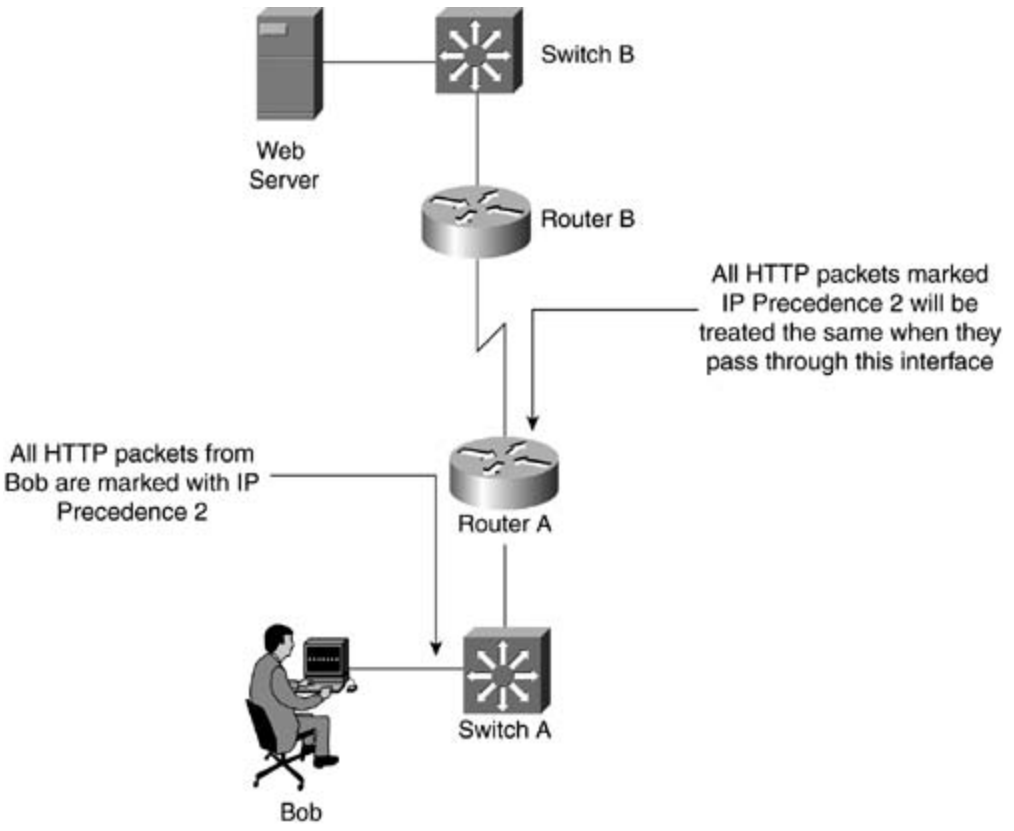
In QoS, managed unfairness is important because sometimes it is necessary to allocate more bandwidth to one application than another. This doesn't specifically indicate that either application is more or less important than the other; rather it indicates a different level of service that will be provided to each application. That is, the applications may well have different bandwidth needs, and dividing available bandwidth equally between the two applications, although fair, might not produce the best results. A good example of such a scenario is the case of an FTP flow sharing a link with a VoIP flow. The FTP flow is characterized by a large bandwidth requirement but has a high tolerance for delay, jitter, and packet loss; the VoIP flow is characterized by a small bandwidth requirement and has a low tolerance to delay, jitter, and packet loss. In this case, the FTP flow needs a larger share of the bandwidth, and the voice flow needs bounded delay and jitter. It is possible to provide each flow with what it needs without significantly impacting the service provided to the other flow. In this case, the allocation of bandwidth is unfair, because the FTP flow will get more bandwidth, but it is unfair in a very controlled manner. Again, this is an example of the need for managed unfairness.

Predictability: The Goal of QoS

The successful management of bandwidth, delay, jitter, and packet loss allows for the differentiated treatment of packets as they move through the network. Unless an implementation

error occurs, all implementations of the differentiated services architecture should provide the same treatment to each packet of the same type when those packets pass through a given interface. In Figure 1-1, multiple packets are sent from Bob to the web server, marked with IP precedence 2.

Figure 1-1. Multiple Packets from Bob Are Sent to the Web Server Marked with IP Precedence 2



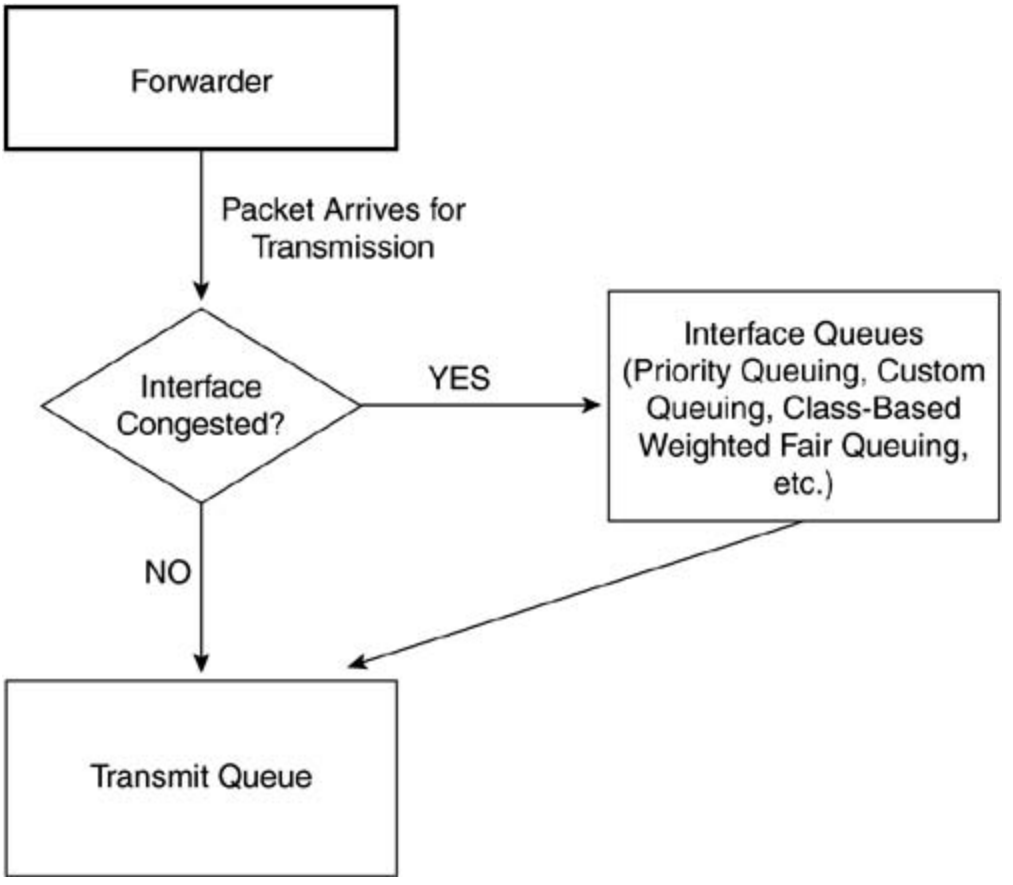
In this example, because all HTTP packets from Bob are marked with IP precedence 2, and the policy on router A's serial interface is to classify all HTTP packets with IP precedence 2 into the same class, you can assume that these packets will all receive the same treatment. Because all packets of the same type are going to be treated the same as they egress that interface, it's easy to predict the treatment that the next HTTP packet from Bob will receive. This is a simplified example of the overall goal of QoS: providing predictable service levels to packets as they move through a network.

It is very important to be able to predict the bandwidth, delay, jitter, and packet loss that can be expected as packets of a given flow traverse multiple hops in a network. Voice packets, for example, must not have a one-way delay greater than 150 ms and are very intolerant of jitter. Being able to say with confidence that voice packets will experience low latency and jitter at each hop along a given path is critical when provisioning IP telephony solutions.

Congestion Management

A variety of QoS mechanisms are available, but the most commonly used is congestion management. Congestion management provides the ability to reorder packets for transmission, which enables a network administrator to make some decisions about which packets are transmitted first and so on. The key to congestion management is that all congestion management QoS mechanisms only have an impact on traffic when congestion exists. The definition of congestion might differ from vendor to vendor and, in fact, is slightly different between Cisco platforms. However, one general statement can be made about the definition of congestion on Cisco platforms; *congestion* is defined as a full transmit queue. Figure 1-2 illustrates the decision model for congestion management.

Figure 1-2. Congestion Management Decision Model



The need for interface queuing results because just a finite amount of buffer space exists in the transmit queue. This finite amount of buffer space is true across all platforms, so this logic applies to both switches and routers. If there were no interface queues, packets would just be dropped when the transmit queue was full.

Deploying QoS in the WAN/LAN: High-Level Overview

Different networks deploy QoS in the WAN and LAN for different reasons, but the overall intent is always to provide different treatment to different types of traffic. Sometimes the requirement is to provide better treatment to a select group of applications, such as VoIP or Oracle. Other times, the requirement is to provide worse treatment to a select group of applications, such as peer-to-peer file sharing services such as Napster, KaZaa, and Morpheus.

Why QoS Is Necessary in the WAN

Originally, the queuing mechanism on all interfaces was *first-in, first-out* (FIFO), meaning that the first packet to arrive for transmission would be the first packet transmitted, the fifth packet arriving would be the fifth packets transmitted, and so on. This queuing mechanism works just fine if all of your traffic has no delay concerns (perhaps FTP or other batch transfer traffic). If you've ever worked with *data-link switching* (DLSw), however, you know how sensitive that traffic is to delay in the network.

For many networks, the first real need for QoS was to allow for priority treatment of DLSw traffic over low-speed WAN links. The need to provide basic prioritization of different types of data traffic was first seen in the WAN, because that is where bandwidth is most limited. In the case of Priority Queuing, the need was to prioritize a single traffic type (or a select few types of traffic) over all others. Custom Queuing addressed the need to provide basic bandwidth sharing, and Weighted Fair Queuing provided the ability to dynamically allocate more or less bandwidth to a given flow based on the IP precedence of the packets in that flow. *Class-Based Weighted Fair Queuing* (CBWFQ) and *Low Latency Queuing* (LLQ) are hybrid QoS mechanisms—that is, they provide a combination of the other functions to allow for greater flexibility.

As you can see, the Cisco congestion management mechanisms address a variety of needs in the WAN, and that only touches the surface of all the needs that can be addressed by QoS.

Why QoS Is Necessary in a Switched Environment

One of the most commonly asked questions is whether QoS is necessary in the Layer 2 environment. The basis for this question generally seems to be the belief that you can just "throw bandwidth at the problem"—that is, alleviate the problem of congestion by continuing to upgrade bandwidth.

Generally speaking, it's difficult to argue against the theory of providing *so* much bandwidth that congestion can be avoided. The truth is that if you install a 100-Mbps link between two switches that need only 10 Mbps, you're not going to have congestion. The cost of this theory starts getting ridiculous, however, when you approach congestion on higher-speed links.

The theory of continuing to upgrade bandwidth suffers from other problems: Primarily, the nature of TCP traffic is that it takes as much bandwidth as it can. For instance, you could upgrade your 100-Mbps link to a 200-Mbps Fast EtherChannel and still not have enough bandwidth to support good voice quality. In this case, perhaps FTP applications dominate the link. Whereas these applications were previously functioning well on the 100-Mbps link, due to TCP windowing, they are now taking far more bandwidth than before. In a case like this, it is difficult to get a true idea about how much bandwidth is required. You could upgrade to a 1-Gbps link, of course, but that confirms the fact that it's going to get expensive very quickly to follow that theory.

Still another problem helps make the case for QoS in the LAN. That problem is interactive traffic, such as voice and video conferencing. With most data traffic, there is no concern about jitter and little concern about delay, but that isn't the case with voice and video conferencing traffic. These real-time applications have special requirements with regard to delay and jitter that are just not addressed by adding more bandwidth. Even with abundant bandwidth, it is still possible that the packets of a voice flow could experience jitter and delay, which would cause call quality degradation. The only way to truly ensure the delay and jitter characteristics of these flows is through the use of QoS.

Cisco AVVID

As the incentives became greater and greater to migrate away from separate networks for data, voice, and video in favor of a single IP infrastructure, Cisco developed the Architecture for Voice, Video, and Integrated Data (AVVID), which provides an end-to-end enterprise architecture for deploying Cisco AVVID solutions. These solutions enable networks to migrate to a pure IP infrastructure; Cisco AVVID solutions include the following:

- IP telephony
- IP video conferencing
- MxU (that is, multi-tenant, hospitality, and so forth)
- Storage networking
- Virtual private networks
- Content networking
- Enterprise mobility
- IP contact center

The idea behind the AVVID architecture is that an enterprise environment can't possibly keep up with every emerging technology and adjust quickly to the individual changes in specific application deployments. With the AVVID architecture, Cisco provides a foundation of network engineering that allows an enterprise environment to quickly adapt to changes in all of these areas. In addition to the physical layer, the AVVID architecture also consists of the intelligent network services (such as QoS) that are necessary to transform a traditional data network into an advanced e-business infrastructure that provides customers with a competitive advantage.

AVVID is not a single mechanism or application; instead, it is an overall methodology that enables customers to build a converged network and adapt quickly to the ever-changing demands placed on that network. The requirements for IP-based voice and video, for example, may be different from the requirements for the next *x*-over-IP requirement.

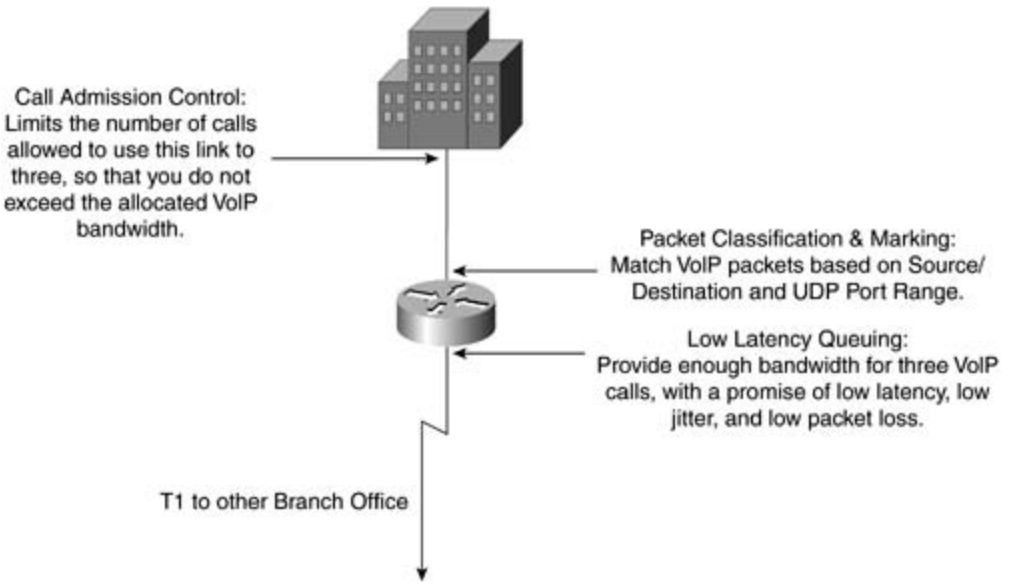
QoS in the AVVID Environment

The foundation for the AVVID architecture is the assumption that all services (including VoIP) use a common infrastructure. The network requirements of VoIP traffic differ from those of a regular data flow (such as FTP). An FTP flow, for instance, requires a large amount of bandwidth, is very tolerant to delay and packet loss, and couldn't care less about jitter. Conversely, VoIP takes a relatively tiny amount of bandwidth, is very sensitive to packet loss, and requires low delay and jitter. By treating these two flows the same on your network, neither would be likely to get ideal service, and the FTP traffic could ultimately dominate the link, causing poor call quality for your VoIP.

For this reason, QoS is one of the cornerstones of the Cisco AVVID. Without QoS applied to the converged links in a network, all packets receive the same treatment and real-time applications suffer. Many QoS considerations exist in a Cisco AVVID environment, but the primary things that all QoS mechanisms are concerned with are constant: bandwidth, delay, jitter, and packet loss.

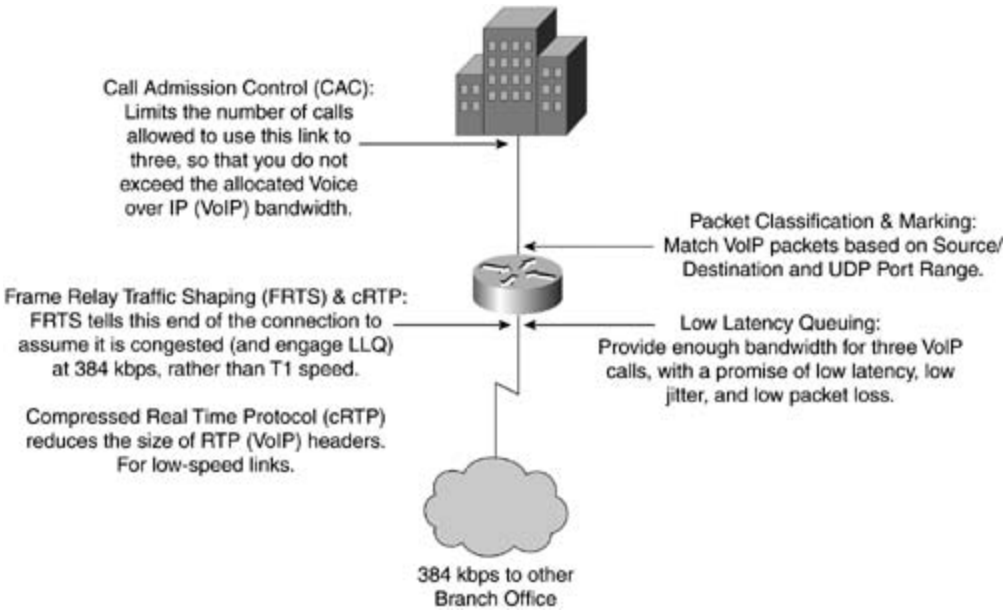
VoIP environments have multiple requirements. Assume that there is a T1 link between two branch offices, and you have determined that you can spare enough of that link for three concurrent VoIP calls. Figure 1-3 shows the minimum QoS mechanisms that you would configure.

Figure 1-3. Minimum QoS Mechanisms for VoIP in a Specific AVVID Environment



If you modify that assumption, only slightly, the required mechanisms change. The changes are not dramatic, but call quality will certainly suffer if they are not made. Assume that the topology now is Frame Relay, rather than a point-to-point connection, with one end at full T1 speed and the other end at 384 kbps. Figure 1-4 shows the additional mechanisms that would be used.

Figure 1-4. QoS Mechanisms for VoIP in a Mixed-Bandwidth Environment


Call Admission Control (CAC):
Limits the number of calls
allowed to use this link to
three, so that you do not
exceed the allocated Voice
over IP (VoIP) bandwidth.
Packet Classification & Marking:
Match VoIP packets based on Source/
Destination and UDP Port Range.
Frame Relay Traffic Shaping (FRTS) & cRTP:
FRTS tells this end of the connection to
assume it is congested (and engage LLQ)
at 384 kbps, rather than T1 speed.
Low Latency Queuing:
Provide enough bandwidth for three VoIP
calls, with a promise of low latency, low
jitter, and low packet loss.
Compressed Real Time Protocol (cRTP)
reduces the size of RTP (VoIP) headers.
For low-speed links.
384 kbps to other
Branch Office

Overview of Integrated and Differentiated Services

QoS standards fit into three major classifications: integrated services, differentiated services, and best effort.

Integrated services and differentiated services are discussed individually, but *best effort* (BE) is not. BE is just the treatment that packets get when no predetermined treatment is specified for them. When there is no QoS at all, for example, all traffic is treated as BE. BE can also be used to refer to that traffic that is not given special (or defined) treatment with integrated services or differentiated services.

Integrated Services Versus Differentiated Services

Several models have been proposed to provide QoS for the Internet. Each has advantages and drawbacks, with regard to the Internet, but the model that has been more generally accepted recently is the differentiated services model. In an enterprise environment, however, both models can prove very useful. Note that you can also use these models in combination to achieve end-to-end QoS, taking advantage of the strengths of each model. At this time, only the differentiated services model is fully supported on the Catalyst 6500.

Definition of Integrated Services

Integrated services (IntServ) is the name given to QoS signaling. QoS signaling allows an end station (or network node, such as a router) to communicate with its neighbors to request specific treatment for a given traffic type. This type of QoS allows for end-to-end QoS in the sense that the original end station can make a request for special treatment of its packets through the network, and that request is propagated through every hop in the packet's path to the destination. True end-to-end QoS requires the participation of every networking device along the path (routers, switches, and so forth), and this can be accomplished with QoS signaling.

In 1994, RFC 1633 first defined the IntServ model. The following text, taken from RFC 1633, provides some insight as to the original intent of IntServ:

> We conclude that there is an inescapable requirement for routers to be able to reserve resources, in order to provide special QoS for specific user packet streams, or "flows". This in turn requires flow-specific state in the routers, which represents an important and fundamental change to the Internet model.

As it turns out, the requirement was not as inescapable as the engineers who authored RFC 1633 originally thought, as evidenced by the fact that the Internet still relies almost entirely on BE delivery for packets.

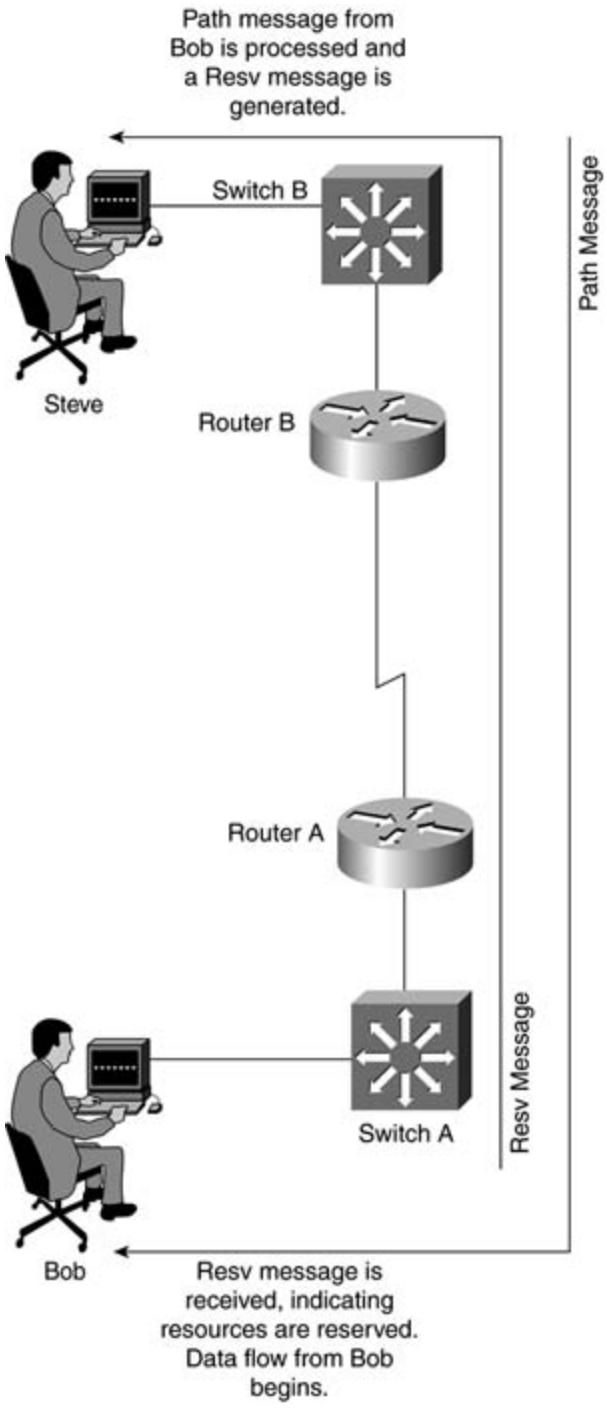
IntServ Operation

Resource Reservation Protocol (RSVP), defined by RFC 2205, is a resource reservation setup protocol for use in an IntServ environment. Specifics of operation are covered shortly, but the general idea behind RSVP is that Bob wants to talk to Steve, who is some number of network hops away, over an *IP video conferencing* (IPVC) system. For the IPVC conversation to be of acceptable quality, the conversation needs 384 kbps of bandwidth. Obviously, the IPVC end

stations don't have any way of knowing whether that amount of bandwidth is available throughout the entire network, so they can either assume that bandwidth is available (and run the risk of poor quality if it isn't) or they can ask for the bandwidth and see whether the network is able to give it to them. RSVP is the mechanism that asks for the bandwidth.

The specific functionality is probably backward from what you would guess, in that the receiver is the one who actually asks for the reservation, not the sender. The sender sends a Path message to the receiver, which collects information about the QoS capabilities of the intermediate nodes. The receiver then processes the Path information and generates a *Reservation* (Resv) request, which is sent upstream to make the actual request to reserve resources. When the sender gets this Resv, the sender begins to send data. It is important to note that RSVP is a unidirectional process, so a bidirectional flow (such as an IPVC) requires this process to happen once for each sender. Figure 1-5 shows a very basic example of the resource reservation process (assuming a unidirectional flow from Bob to Steve).

Figure 1-5. Path and Resv Messages for a Unidirectional Flow from Bob to Steve



The other major point to note about RSVP is that RSVP doesn't actually manage the reservations of resources. Instead, RSVP works with existing mechanisms, such as Weighted Fair Queuing, to request that those existing mechanisms reserve the resources.

Although RSVP has some distinct advantages over BE and, in some cases, over differentiated services, RSVP implementations for end-to-end QoS today are predominantly limited to small implementations of video conferencing. That said, RSVP is making a strong comeback and some very interesting new things (beyond the scope of this book) are on the horizon for RSVP. If you're interested in a little light reading on the subject, have a look at RFCs 3175, 3209, and 3210.

Definition of DiffServ

To define *differentiated services* (DiffServ), we'll defer to the experts at the IETF. The following excerpt is from the "Abstract" section of RFC 2475:

> This document defines architecture for implementing scalable service differentiation in the Internet. This architecture achieves scalability by aggregating traffic classification state which is conveyed by means of IP-layer packet marking using the DS field [DSFIELD]. Packets are classified and marked to receive a particular per-hop forwarding behavior on nodes along their path. Sophisticated classification, marking, policing, and shaping operations need only be implemented at network boundaries or hosts. Network resources are allocated to traffic streams by service provisioning policies which govern how traffic is marked and conditioned upon entry to a differentiated services-capable network, and how that traffic is forwarded within that network. A wide variety of services can be implemented on top of these building blocks.

To make that definition a little less verbose: The differentiated services architecture is designed to be a scalable model that provides different services to different traffic types in a scalable way. It must be possible to tell packets of one type from another type to provide the DiffServ, so techniques known as packet classification and marking are used. After packets of different types have been marked differently, it's possible to treat them differently based on that marking at each hop throughout the network, without having to perform additional complex classification and marking.

DiffServ Operation

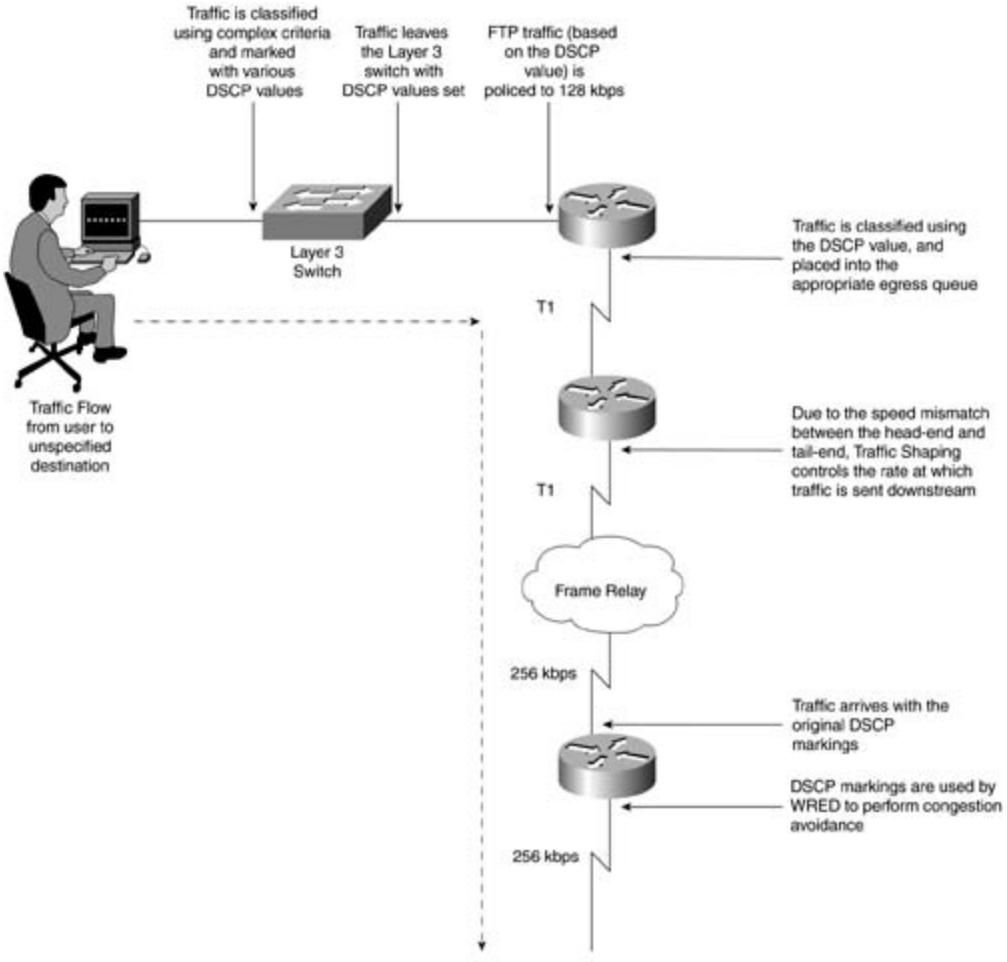
DiffServ is a complicated architecture, with many components. Each of these components has a different purpose in the network and, therefore, each component operates differently. The major components of the DiffServ architecture perform the following tasks:

- Packet classification
- Packet marking
- Congestion management
- Congestion avoidance
- Traffic conditioning

These mechanisms can be implemented alone or in conjunction with each other. Figure 1-6 shows a possible implementation of these mechanisms in a production network.

Figure 1-6. An Example of the Implementation of Various DiffServ Components

[View full size image]



The sections that follow describe the five major components of the DiffServ architecture in greater detail.

Packet Classification

Packet classification can be simple classification, based on Layer 2 or Layer 3 information, and is, as the name implies, a set of mechanisms that can distinguish one type of packet from other. An example of a simple packet classification mechanism is matching against an access list that looks for packets with a specific source and destination IP address. This packet classification can also be far more complex, looking at things such as destination URL and MIME type. An example of a more complex packet classification mechanism available in Cisco routers is *network-based application recognition* (NBAR). NBAR is capable of matching on a variety of Layer 4 through Layer 7 characteristics, such as those listed previously. NBAR is also capable of stateful packet inspection, which dramatically increases the potential functionality. Whatever the actual classification capability of a specific mechanism, packet classification is typically performed as close as possible to the traffic source and is usually used in conjunction with packet marking. The words*typically* and *usually* were used intentionally here, because your particular setup may necessitate performing these functions at other places in your network.

Packet Marking

Packet marking is a function that allows a networking device to mark packets differently, based on their classification, so that they may be distinguished more easily at future network devices. Consider this analogy: Many states have smoking-prohibited sections in restaurants, but restaurants in North Carolina (where all of this book's authors live and work) still have smoking sections. Therefore, every time we walk into a restaurant, we have to state our personal preference about whether we want to sit in the smoking or non-smoking section. It seems like a lot of wasted time to ask and answer that question over and over again—wouldn't life be easier if I could just walk into a restaurant and they knew where to seat me? This is a loose analogy to the function of packet marking in the sense that a packet only requires complex classification (Do you prefer smoking or nonsmoking, sir?) to happen once. After the packet has been classified at the first router hop, a marking is applied so that all future network hops can just look at the marking and know what to do with that packet. Packet marking is, as previously mentioned, generally deployed in conjunction with packet classification as close to the source as possible. One of the reasons that the DiffServ model is so scalable is that the complex packet classification and packet marking are both recommended for deployment on only the first-hop Layer 3-capable device. In a typical enterprise network deployment, this means a branch office device (which serves a small subset of the total user community) performs the complex operations for that branch. Then that marking is carried with the packet throughout the network, limiting the burden on the core of the network to very simple classification of packets (based on the markings that were applied at the edge of the network) and the switching of those packets to the appropriate egress interface.

Congestion Management

Congestion management has many subcomponents (discussed in more detail later in this chapter, but the overall function of congestion management is to isolate various classes of traffic (based either on complex classification at the first hop or based on packet marking at nonedge devices), protect each class from other classes, and then prioritize the access of each class to various network resources. Typically, congestion management is primarily focus on re-ordering packets for transmission. This impacts the overall bandwidth given to each class, however, and also impacts the delay and jitter characteristics of each class. Because of limited queue lengths, the delay experienced by packets in a given class could impact the packet loss experienced by the traffic in that class. Stated another way, if a large amount of delay exists for a given traffic type and the queue fills, packets for that class will be tail dropped. Congestion management in Cisco routers is an egress function and includes mechanisms such as CBWFQ and LLQ. Congestion management is typically used at all network layers (access, distribution, and core) in a real-time environment, but no strict rules dictate where you must use congestion management in your network.

Congestion Avoidance

Congestion avoidance is specifically designed to discard packets to avoid congestion. The concept behind congestion avoidance is based on the operation of TCP. The details of TCP operation are beyond the scope of this text, but the basic concept is that when TCP traffic is sent, the receiver of the traffic is expected to acknowledge the receipt of said traffic by sending an *acknowledgment* (ACK) message to the sender. If the sender doesn't receive this ACK in a given amount of time, it assumes that the receiver didn't receive the transmission. In response to this assumption, the sender will reduce its TCP window size (which essentially reduces the rate of transmission) and retransmit the traffic for which it did not receive an ACK. Note that all of this happens without the sender ever actually being told by the receiver that packets weren't received. A good analogy is two people having a conversation; one person asks the other a question, but receives no answer. After some reasonable amount of time, the person probably assumes that the other person didn't hear the question and repeats the question. Congestion

avoidance is implemented in Cisco routers as *Weighted Random Early Detection* (WRED) and is the process of monitoring the depth of a queue, and randomly dropping packets of various flows to prevent the queue from filling completely. This section covers the concept of congestion avoidance, however, not the WRED implementation specifics. Chapter 2, "End-to-End QoS: Quality of Service at Layer 3 and Layer 2," contains a more thorough discussion of WRED. By randomly discarding packets of various flows, two things are prevented. First, you prevent the queue from becoming completely full—if allowed to fill completely, "tail drop" would occur (that is, all incoming packets would be dropped). Tail drop is not good, because multiple packets from the same flows can be dropped, causing TCP to reduce its window size several times, thereby causing suboptimal link utilization. Second, it is possible to add intelligence to the decision-making process for which packets are "randomly" dropped; in the case of WRED, IP precedence or*DiffServ codepoint* (DSCP) markings can be used to influence which packets are dropped and how often.

Traffic Conditioning

Traffic conditioning contains two major components:

- Policers— Policers perform traffic policing, which means dropping packets that exceed a defined rate to control the rate at which traffic passes through the policer. Examples of policers implemented by Cisco are the *committed access rate* (CAR) policer and the class-based policer. The goal of policing is to rate limit traffic; an example of a practical application for a policer is to limit the amount of FTP traffic allowed to go out of a given interface to 1 Mbps. Traffic that exceeds this 1-Mbps rate limit is dropped. TCP retransmits dropped packets, UDP does not, which is a consideration when deciding whether a policer is right for a given situation.

- Shapers— Shapers perform traffic shaping, which, in Cisco equipment, takes many forms; *generic traffic shaping* (GTS), *Frame Relay Traffic shaping* (FRTS), class-based traffic shaping, and so on. The specific shaper that is used is not of consequence, because the concept is the same for all of them. The goal of the shaper is to limit the rate at which packets pass through the shaper by buffering packets that exceed a defined rate and sending those packets later. The goal is that, over time, the rate of transmission will be smoothed out to the defined rate. This is in contrast to the operation of a policer, where traffic is dropped if the defined rate is exceeded. Both policers and shapers have benefits and drawbacks, and each situation must be evaluated to determine which mechanism is best. For example, FTP traffic (which is TCP based and very tolerant of packet drops) can be policed without negatively impacting the usability of the application. Because FTP doesn't mind some delay in the transmission of its packets, in many cases FTP can also be shaped without any negative impact to the application. VoIP traffic, on the other hand, does not tolerate delay very well at all, so it is desirable to drop (police) a VoIP packet rather than delay (shape) it.

Differentiated Services: A Standards Approach

As you have just seen, many components comprise the DiffServ architecture, and those components can be used in many different ways. Of course, there are also different implementations of these mechanisms, which have been given different names by different vendors. The key to the DiffServ architecture's successful implementation in a multivendor environment, however, is that the entire architecture is standards-based. Regardless of what name each vendor uses to market a given feature, all the features that comprise the DiffServ architecture are standardized and should, therefore, interoperate between vendors with very predictable results. The idea of being able to provide predictable service to packets through the network is fundamental to being able to provide good QoS. This is especially critical when dealing with real-time interactive traffic, such as VoIP, but is also important for consistent data handling across multiple network nodes.

RFC 2475: Terminology and Concepts

The treatment given to a packet at each of these nodes, or hops, is called a *per-hop behavior* (PHB). PHBs are defined by RFC 2475 as "the externally observable forwarding behavior applied at a DS-compliant node to a DS behavior aggregate." For clarification, RFC 2475 also defines a DS behavior aggregate (or BA, which isn't nearly as complicated as it sounds) as, "a collection of packets with the same DS codepoint crossing a link in a particular direction." This concept was introduced earlier in this chapter, but RFC 2475 basically says that all packets with the same DSCP marking *must* be treated the same when passing through a given interface, in a given direction. It is possible to define a BA based on multiple criteria. Although not explicitly defined in the terminology of RFC 2475, the permissibility of such a BA definition can be inferred from the definition of the *multifield* (MF) classifier, "which selects packets based on the content of some arbitrary number of header fields; typically some combination of source address, destination address, DS field, protocol ID, source port and destination port." In other words, it is possible to group packets together, to receive a PHB, based on criteria other than the DSCP marking of those packets.

The question that naturally follows the definitions given by RFC 2475 is "what is a 'forwarding behavior?'" RFC 2475 states the following:

> "Forwarding behavior" is a general concept in this context. For example, in the event that only one behavior aggregate occupies a link, the observable forwarding behavior (i.e., loss, delay, jitter) will often depend only on the relative loading of the link (i.e., in the event that the behavior assumes a work-conserving scheduling discipline). Useful behavioral distinctions are mainly observed when multiple behavior aggregates compete for buffer and bandwidth resources on a node. The PHB is the means by which a node allocates resources to behavior aggregates, and it is on top of this basic hop-by-hop resource allocation mechanism that useful differentiated services may be constructed.

Simply stated, it's something that you can distinctly measure (that is, bandwidth, delay, jitter, loss), and observing different forwarding behaviors is typically only possible when multiple traffic types compete for resources on a congested link. Further, this basic ability to provide different resource allocations to behavior aggregates on a hop-by-hop basis is the foundation for DiffServ.

RFC 2475 also gives a great example:

> The most simple example of a PHB is one which guarantees a minimal bandwidth allocation of X% of a link (over some reasonable time interval) to a behavior aggregate. This PHB can

be fairly easily measured under a variety of competing traffic conditions. A slightly more complex PHB would guarantee a minimal bandwidth allocation of X% of a link, with proportional fair sharing of any excess link capacity.

RFC 2474: Terminology and Concepts

As previously discussed, traffic can be grouped into behavior aggregates only by DSCP, or based on multiple fields (for example, DSCP *and* source IP address). As you read this section, keep in mind that the DiffServ RFCs focus mainly on the classification by DS behavior aggregate.

RFC 2475 states the following:

> Nodes within the DS domain select the forwarding behavior for packets based on their DS codepoint, mapping that value to one of the supported PHBs using either the recommended codepoint → PHB mapping or a locally customized mapping [DSFIELD].

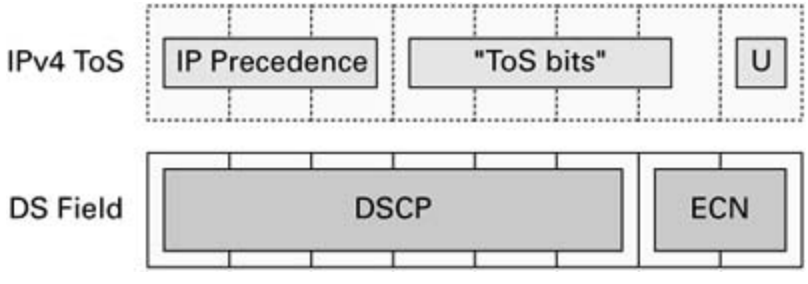
In more basic terms, a networking device decides which PHB to give a packet based on the DSCP value assigned to the packet. This value can either be one that is mapped to a PHB as defined by an RFC, or it can be a marking that is not standardized but has local significance. For example, DSCP value 26 (decimal) is mapped to a standardized PHB (called AF31, which is discussed later in this chapter), but DSCP value 27 (decimal) is not mapped to any standardized PHB. Therefore, when a network device receives packets marked with DSCP 26, certain assumptions can be made about the expected forwarding behavior. On the other hand, no assumptions can be made about the expected forwarding behavior for packets marked with DSCP 27, because that codepoint has no standardized PHB.

Because codepoint markings are very important in the DiffServ architecture, it seems prudent to discuss the origins of the DSCP values: RFC 2474. RFC 2474 obsoletes RFC 791, which defined the IPv4 *type of service* (ToS) octet, more commonly discussed as IP precedence. The Abstract section of RFC 2474 states the purpose of the document:

> This document defines the IP header field, called the DS (for differentiated services) field. In IPv4, it defines the layout of the TOS octet; in IPv6, the Traffic Class octet. In addition, a base set of packet forwarding treatments, or per-hop behaviors, is defined.

Figure 1-7 shows the DS field format.

Figure 1-7. DS Field Format



The addition of *explicit congestion notification* (ECN) to IP (RFC 2481) has since redefined bits 6 and 7 for use; however, a discussion of ECN is beyond the scope of this text.

RFC 2474 goes on to require that codepoint-to-PHB mappings must be configurable and that the default configuration should include recommended codepoint-to-PHB mappings. The exception to the requirement that codepoint-to-PHB mappings be configurable is for codepoints xxx000, because these codepoints, called class selector codepoints, are reserved for backward compatibility with IP precedence. The specifics of this backward compatibility are beyond the scope of this text, but the concept is that when bits 3, 4, and 5 of this byte are set to 0, and any of the first 3 bits of the byte (which were previously used for IP precedence) are not 0, the packet is handled as if it is marked with IP precedence. In other words, 001000 and 010000 are treated as if they are IP precedence 1 and IP precedence 2, respectively. No attempt is made in RFC 2474 to maintain any backward compatibility with bits 3 through 6 or 7 (previously used as the ToS bits and the *must be zero* [MBZ] bit, respectively).

Note as well that RFC 2474 documents the use of the codepoint 000000, which is the codepoint for BE treatment. BE is the treatment that packets get when no QoS is enabled in a network, and essentially means that the networking devices make every effort possible to get the packet to the destination, but ultimately there are absolutely no guarantees with regard to delay, jitter, packet loss, or bandwidth. In fact, there is no guarantee that the packet will even be transmitted. For example, a router might have three classes of traffic, each guaranteed 1/3 of the total link bandwidth. If all three classes are sending traffic equal to 1/3 of the total link bandwidth, the link is going to be totally congested. In a situation like this, BE traffic, which by definition is not classified into one of these three classes, has either very limited resources or none at all (depending on the implementation specifics). This situation could lead to significant delay and, ultimately, packet loss.

Assured Forwarding Versus Expedited Forwarding

Assured forwarding (AF, RFC 2597), and *expedited forwarding* (EF, RFC 2598) are PHBs that have recommended codepoints. EF has only 1 recommended codepoint, whereas AF has 12 recommended codepoints.

RFCs 2597 and 2598 both describe PHBs; beyond that, they are not specifically related, but the discussion of each has been combined in this section to contrast the two functions. The point of contrasting these two PHBs is to demonstrate how the DiffServ architecture allows for a great deal of flexibility in the PHBs that fall under the DiffServ umbrella. No relation or interdependency exists between AF and EF, but because the EF PHB is the easier of the two to understand, it is covered first.

RFC 2598, "An Expedited Forwarding PHB"

RFC 2598 defines "An Expedited Forwarding PHB" and states the following:

> The EF PHB can be used to build a low loss, low latency, low jitter, assured bandwidth, end-to-end service through DS domains.

This means that when a network node receives a packet, it should be transmitted as soon as possible (adding as little delay as possible). This sounds a lot like voice service, right? Although RFC 2598 doesn't explicitly discuss the intended use of this PHB, it can probably be safely assumed that the primary intention for this PHB is real-time traffic, including voice and, in some cases, interactive video. That's not, of course, to say that this PHB isn't ever used for any other kind of traffic. I suppose that's one of the great things about the DiffServ architecture implementation in Cisco equipment; it is very flexible and entirely user configurable. When I teach QoS classes, my joke is that Cisco enables users to configure QoS any way they want to … even if it's wrong. RFC 2598 is actually one of the few things in the DiffServ architecture that

provides an opportunity to get into real trouble with misconfiguration.

Consider the fact that the EF PHB's main purpose is to provide a forwarding behavior that introduces as little delay and jitter as possible. If more traffic is received for transmission than an interface can transmit, a queue begins to form. When a device queues traffic, by definition it introduces delay. As such, it can be inferred that building a queue is undesirable when implementing the EF PHB.

Said another way, if a device queues traffic, it introduces delay; so to provide the EF PHB, a device should not queue (or queue very little) traffic. Because this requirement means that the EF traffic will be given strict priority for transmission (to minimize delay), there is a risk that the receipt of too much EF traffic could cause starvation for other traffic on the link. One might envision a situation in which there is a steady stream of EF traffic, received at the line rate of the egress interface, and a stream of other traffic, received at the line rate of the egress interface. In this situation, without a mechanism to prevent starvation, the strict priority of the EF traffic on that link means that all the EF traffic is transmitted and none of the other traffic is transmitted. That may very well be the intent of a particular user, but to prevent this situation from unintentionally occurring, RFC 2598 calls for a measure of protection.

The measure that is called for to prevent the starvation of other traffic, just because a device receives too much EF traffic, is the following provision:

> The departure rate of the aggregate's packets from any DiffServ node must equal or exceed a configurable rate.

This behavior can be represented by stating that the rate of arrival must be less than or equal to the rate of departure, for packets receiving the EF PHB. To ensure that this is the case, the RFC suggests that a policer be used to rate limit the EF traffic, which is what is used in most Cisco devices. Specifically, if the configuration requires EF for all packets marked DSCP 46, and if 128 kbps is the configured egress rate for traffic receiving the EF PHB, and 129 kbps of traffic marked DSCP 46 is offered, 1k of traffic marked DSCP 46 is dropped to ensure that the rate of arrival into the EF queue does not exceed the rate of departure. This example, by the way, assumes link congestion is present. When no congestion exists, queuing does not become active and, therefore, the policing mechanism would also not become active. Stated another way, when there is no congestion, there is no policing of traffic that would, in the event of congestion, be placed in the priority queue for strict priority scheduling.

RFC 2597, "Assured Forwarding PHB Group"

In part, the Abstract of RFC 2597 states the following:

> The AF PHB group provides delivery of IP packets in four independently forwarded AF classes. Within each AF class, an IP packet can be assigned one of three different levels of drop precedence.

Until now, this text has only discussed single PHBs, so new terminology needs to be clarified before discussing AF in detail. The original RFC (2597) calls AF, as a whole, a PHB group. RFC 3260 later clarifies this definition and states that AF is actually a type of PHB group, which actually contains four separate PHB groups.

If you understand the implications of that correction, great! If you're scratching your head at the moment, don't despair. Before you finish reading this section, you'll understand the distinction and its importance.

RFC 2597 defines 12 DSCPs, which correspond to 4 AF classes, each class having 3 levels of "drop precedence." Visualize four totally separate buckets, each having three compartments, and

you have the general idea. Each AF class (referred to as AF1x, AF2x, AF3x, and AF4x) is completely independent of the other classes. Within each class, however, there are three levels of drop precedence (for example, AF1x contains AF11, AF12, and AF13). These drop precedence levels, within each class, have relativity to the other drop precedence values in their class, but no relativity at all to the other classes. Note that the first number is always the AF class to which the packets belong, and the second number is always the packet's drop precedence value. For clarification, there is no such thing as AF10 or AF14; there are only three levels of drop precedence per AF class.

Stated more directly, each AF class is totally independent of the other three and no assumptions can be made regarding the treatment of packets belonging to one class when compared with the treatment of packets belonging to another class. Within a class, however, assumptions may be made regarding the treatment of packets with different drop precedence values.

All the classes and their drop precedence values are represented by the DSCP markings that are given to packets. Table 1-2 lists the 12 recommended codepoints defined by RFC.

Table 1-2. Codepoints Recommended by RFC 2597

Class	Low Drop Precedence	Medium Drop Precedence	High Drop Precedence
AF1	001010 (AF11)	001100 (AF12)	001110 (AF13)
AF2	010010 (AF21)	010100 (AF22)	010110 (AF23)
AF3	011010 (AF31)	011100 (AF32)	011110 (AF33)
AF4	100010 (AF41)	100100 (AF42)	100110 (AF43)

RFC 2597 dictates that packets with a low drop preference must be dropped with a probability less than or equal to the probability with which medium drop precedence packets would be dropped and that packets with a medium drop precedence must be dropped with a probability less than or equal to the probability with which high drop precedence packets would be dropped. The "or equal to" part allows for the equal treatment of all packets within an AF class but, assuming that the drop precedence is going to be used to treat packets differently, high drop precedence packets are going to be dropped first in an AF class. Again, there can be no assumptions made about the probability of a packet from AF1x being dropped, with respect to the probability of a packet from AF2x being dropped.

Recall, from the beginning of this section, the seemingly pointless clarification of the AF PHB standard as one that defines multiple PHB groups, rather than one large PHB group? The reason that distinction is so critical is that a PHB group has components that are somehow relative to each other. With the redefinition of the AF standard as one that defines four distinct PHB groups, the point is now very clear that AF1x is a PHB group by itself and is, therefore, *totally separate* from the other AF classes. This total separation makes sure that everyone implementing the AF standard knows that absolutely no relativity exists between the drop probabilities of packets in different AF classes.

RFC 2597 and RFC 2598: Practical Application in a Cisco Network

All the RFCs in the world don't do anyone any good until they are actually implemented in a network, so it seems prudent to look at the Cisco implementation of these two RFCs. Cisco

routers provide EF or AF treatment to implement these RFCs. Later chapters detail the implementation specifics on the various Catalyst platforms, but it's important to first understand these RFCs as they are implemented on Cisco routers so that you can use the Catalyst information provided later to develop an end-to-end QoS strategy for your network.

EF Treatment

First, the EF PHB recommends DSCP 46 (101110) be used to mark packets that should received EF treatment. The mechanism for performing this marking is beyond the scope of RFC 2598, but many mechanisms can be used to perform this task, as the list that follows demonstrates:

- CAR
- Policy-based routing
- Dial peers
- Class-based marking
- Class-based policer

In almost every case, class-based marking and class-based policer are the recommended methods for performing packet marking in Cisco routers.

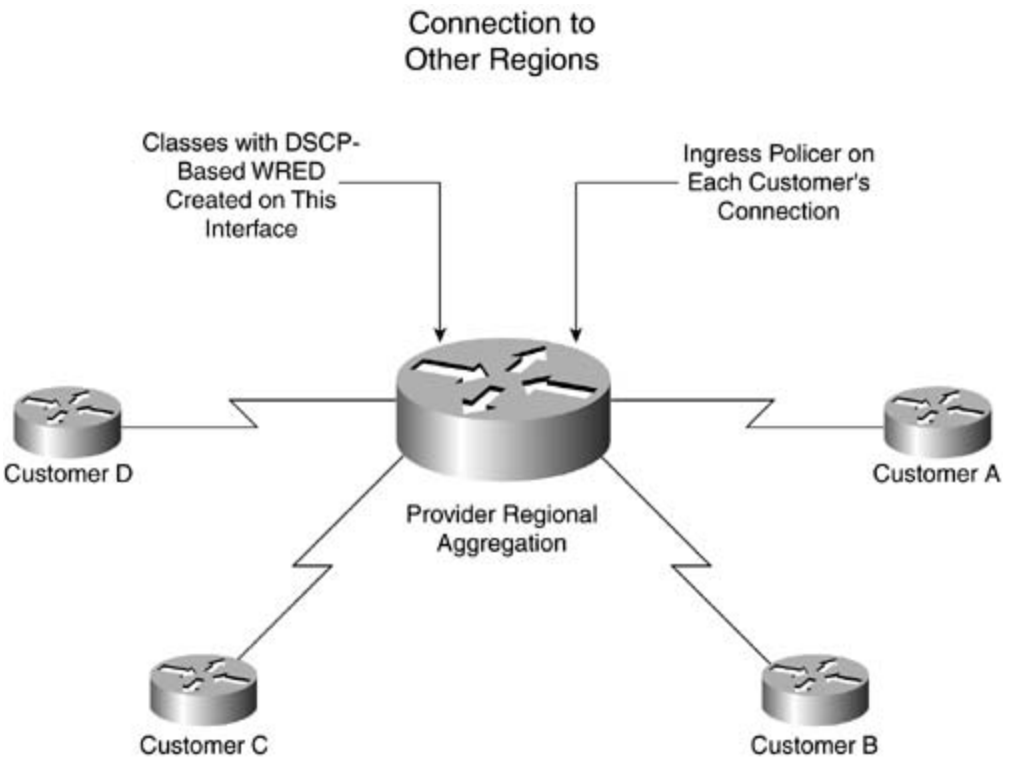
The mechanism in Cisco routers that actually provides the EF treatment to packets is called *Low Latency Queuing* (LLQ). A lengthy discussion of LLQ's operation is beyond the scope of this text, but the basic configuration defines a rate of departure for packets classified into the LLQ for EF treatment. This definition of the rate of departure activates a policer for the rate of arrival for packets into the LLQ. As such, a definition of 128 kbps of bandwidth with LLQ treatment means that, if 129 kbps of traffic is marked for EF, and offered to the LLQ, 1 kbps will be dropped. LLQ extends the CBWFQ model by introducing a single, strict priority queue.

AF Treatment

With regard to the same packet marking tools previously discussed, 12 codepoints are recommended for use with AF. These 12 codepoints, listed earlier in the chapter, are divided into 4 classes, each with 3 codepoints.

Figure 1-8 shows a possible implementation of AF.

Figure 1-8. Possible Implementation of Assured Forwarding



In this example

- A class called Customer-A will get AF11, AF12, and AF13 traffic.
- A class called Customer-B will get AF21, AF22, and AF23 traffic.
- A class called Customer-C will get AF31, AF32, and AF33 traffic.
- A class called Customer-D will get AF41, AF42, and AF43 traffic.

For each customer, an ingress policer marks the traffic based on the rate of the traffic. Up to 256 kbps is marked AFx1, above 256 kbps but below 768 kbps is marked AFx2, and above 768 kbps is marked AFx3. During periods of congestion, this traffic is placed into the classes described earlier as it egresses the aggregation router toward other regions. Because each class of traffic is independently forwarded, and each class is guaranteed a certain amount of bandwidth (using CBWFQ), the classes do not interfere with each other in any way.

Assume congestion and that Customers A, B, and C are all sending traffic at their CIR, but Customer D is sending traffic at 1.5 Mbps. It wouldn't be fair to punish the other customers because Customer D is not "behaving," so traffic is queued in the class called Customer-D and, when that queue begins to fill, WRED begins to discard packets. Per the RFC that defines AF, packets marked AF43 (those that were over 768 kbps) are discarded first, then packets marked AF42 (those that were between 256 kbps and 768 kbps), and packets marked AF41 are discarded only if no packets are marked AF43 or AF42.

It's not difficult to see how this could provide a service similar to Frame Relay where there is a CIR with burst and extended burst capabilities. However, no hard rule mandates the purpose for which AF must be used in your network.

Summary

This chapter defined QoS as the ability to create predictable service levels for various traffic types in the network, and also as "managed unfairness" (that is, the ability to provide different traffic types with unequal treatment while giving each type the treatment that it requires).

Integrated services and differentiated services have been discussed in detail, and you should now understand where you might find each of these useful in your network. The thorough discussion of some of the key DiffServ RFCs has provided the background information that you need in future chapters, which discuss Cisco's specific implementations of these standards. As these mechanisms are discussed, you should be able to relate their behavior to the standardized behaviors explained by the various RFCs that have been discussed in this chapter.

Chapter 2. End-to-End QoS: Quality of Service at Layer 3 and Layer 2

Chapter 1, "Quality of Service: An Overview," introduced several of the fundamental concepts of *quality of service* (QoS), various key RFCs, and explained various parts of the *differentiated services* (DiffServ) architecture. This chapter builds upon Chapter 1 by continuing to explore some DiffServ components and looking at mechanisms for traffic conditioning, link efficiency, and classification at Layer 3. This chapter also introduces the concept of Layer 2 packet marking for QoS and explains how Layer 2 markings relate to Layer 3 markings to facilitate end-to-end QoS. Although this chapter doesn't focus specifically on QoS for IP telephony applications, you will find examples of how Layer 2 QoS is used to allow voice traffic to traverse a switched infrastructure with limited delay and jitter. This chapter also looks at various traffic-conditioning components of QoS.

This chapter primarily focuses on QoS from a Cisco IOS router perspective. In addition, all the examples of this chapter were demonstrated on a Cisco IOS router. The last two sections of this chapter introduce QoS on the Catalyst platforms.

QoS Components

As discussed in Chapter 1, QoS is comprised of various components. This chapter discusses the following QoS components in detail:

- Congestion Management
- Congestion Avoidance
- Traffic Conditioning
- Link Efficiency

Each QoS component plays an important role in building an overall QoS strategy for your network, and each component offers unique characteristics that add value to your strategy.

In addition to discussing the various components, this chapter introduces you to some Cisco implementations of these components. After discussing the concepts in general terms, this chapter provides configuration examples and details how to verify your configurations using show commands.

Simply defined, *congestion* exists when an interface is offered more traffic for transmission than it is capable of transmitting. Congestion is usual in several common network design scenarios:

- LAN-to-WAN connections— When traffic is flowing from a LAN, where bandwidth is typically at least 10 Mbps, to a WAN, where bandwidth is typically less than 10 Mbps, the possibility of congestion is very real.
- Aggregation— When several links are aggregated and their traffic transmitted over a single link, the possibility exists that a situation may cause congestion. If ten 10-Mbps links were being aggregated into a Gigabit Ethernet connection, congestion would not be possible, because the single link's bandwidth would exceed that of the aggregated links. However, it is more common to see 24 or even 48 100-Mbps ports aggregated into a single Gigabit Ethernet uplink. In this situation, the sum of traffic from the aggregated links has the potential to exceed the capacity of the gigabit port.
- Speed mismatch— It is also possible to have congestion when traffic flows downstream, from higher-speed aggregation links to lower-speed access links. For example, traffic coming from the core of the network, through a Gigabit Ethernet link, to a PC that is connected by a 100-Mbps connection may experience congestion. In this example, the congestion point would occur at the 100-Mbps Ethernet port, as traffic tries to egress that port toward the PC.

Congestion Management

QoS involves many components and features, but the component that is most typically associated with the term QoS is congestion management. Congestion management is the key component for QoS on Catalyst switches. The congestion management component of QoS itself is made up of many different features in Cisco IOS and CatOS. All Catalyst switches that support QoS features support congestion management or congestion avoidance. The next section looks at these features in detail, but the purpose of this section is to define congestion management, in general.

As the name implies, congestion management enables you to manage the congestion that is experienced by packets at a given point in the network. Congestion management involves three main steps:

1. Queues are created at the interface where congestion is expected. Depending on the specific feature or mechanism being used to provide QoS and the platform on which the QoS is being configured, there could be only two queues or there could be several hundred (although there is currently no practical application for this many queues on any Catalyst platform).

2. Packets (this could also be frames, but for the sake of simplicity, the word *packets* is used) are then assigned to these queues, based on classification characteristics such as *DiffServ codepoint* (DSCP) value. The classification of packets by characteristics is typically user-defined, and packets are placed into queues by these predetermined characteristics. Some examples of packet characteristics that are typically used in classification are the values in the packet for IP precedence, DSCP, and Layer 2 *class of service* (CoS). It is also common to use extended access lists to match packets based on more complex criteria, such as port numbers.

3. Packets are then scheduled for transmission. Determining the scheduling of packets is specific to every congestion management mechanism, but it can generally be said that there will be a way to provide for the transmission of more packets out of some queues than some others. That is, you will be able to give some packets better treatment than others, with regard to the amount of bandwidth that those packets receive.

Congestion Management Mechanisms

As previously mentioned, several mechanisms in Cisco devices fall under the definition of congestion management. Although this chapter does not get deep into the specifics of these mechanisms, because there are several other books from Cisco Press that already cover these in detail, an overview of the various mechanisms will help explain the overall function of congestion management as part of an end-to-end QoS strategy.

The preceding section alluded to the need to queue packets, but did not explicitly discuss that need. When dealing with a situation where more traffic needs to be transmitted than an interface is capable of transmitting, some traffic must be queued while waiting for the resources necessary to transmit that packet. Cisco devices utilize four main types of queuing, as follows:

- First In, First Out

- Priority Queuing
- Custom Queuing
- Weighted Fair Queuing

First In, First Out

First In, First Out (FIFO) Queuing has no concept of priority for packets. FIFO is the most basic form of queuing, in that there is a default queue depth (64 packets in most cases) and that number of packets will be buffered, in the order that they were received, and transmitted in the order that they were received. A good analogy for FIFO is flying an airline where the first person to show up at the counter gets to board the plane first, and the last person to arrive boards last; there is no priority boarding for frequent flyers or other special treatment for certain passengers.

FIFO Queuing is the default behavior of most Catalyst switches where QoS is disabled and all Catalyst switches that do not support QoS.

Priority Queuing

Priority Queuing (PQ), on the other hand, is all about priority. With PQ, there are four queues to which packets are assigned. The queues are called High, Medium, Normal, and Low, and packets are serviced in that order, with all packets from the High queue being transmitted first. You can draw an analogy between PQ and the way that most airlines board their flights. All of the first-class passengers board first, then all of the elite frequent fliers, then the passengers in the rear of the main cabin, and then the passengers in the front of the main cabin. Unlike an airline, which eventually allows everyone with a ticket to board, PQ would continue to service "elite" packets to the point that other packets would not be able to get through. This condition is called *queue starvation*, because the applications with packets in lower-level queues can actually be starved of bandwidth by a high volume of traffic in higher-level queues.

Current Catalyst switches support PQ as an optional method for applying strict priority to *Voice over IP* (VoIP) traffic. Each platform that supports PQ varies slightly in implementation and architecture. Refer to the corresponding platform chapters for a discussion these variations.

Custom Queuing

Custom Queuing (CQ) does not have a strict priority scheduling mechanisms like PQ; instead, CQ tries to be fair by providing all classes with some ability to transmit packets in each interval of time. With CQ, there can be up to 16 queues and the packets in those queues are serviced in a round-robin fashion. That is, some packets from each queue are transmitted in each time interval. This method is similar to an airline deciding that in every 5-minute period, they will board 1 first-class passenger, 5 elite frequent fliers, 10 people seated in the rear of the main cabin, and 7 people seated in the front of the main cabin. If it seems like this method might become a little confusing at the airport, rest assured that it's fairly confusing to configure in a router, too.

Weighted Fair Queuing

Weighted Fair Queuing (WFQ) is a dynamic process that divides bandwidth among queues based on weights. The process is designed to be fair, such that WFQ ensures that all traffic is treated

fairly, with regard to its weight.

There are several forms of WFQ, including *Class-based Weighted Fair Queuing* (CBWFQ) and *Low Latency Queuing* (LLQ).

CBWFQ is probably the form of WFQ that is most commonly being deployed these days. CBWFQ works quite a bit like CQ, but the algorithm is more efficient and the configuration is quite a bit easier to understand. With CBWFQ, classes are created and traffic is assigned to those classes, as explained earlier in this chapter. Bandwidth is then assigned to those classes, and the amount of bandwidth assigned to a given class determines the amount of scheduling that class receives. In other words, the bandwidth statement on a given class determines the minimum amount of bandwidth that packets belonging to that class receive in the event of congestion.

In the recent past, a PQ was added to the CBWFQ mechanism, specifically to handle VoIP traffic. This addition was necessary because, although CBWFQ did an excellent job of dividing up the available bandwidth, CBWFQ did not give any specific regard to the delay or jitter being introduced by queuing packets.

The LLQ mechanism is CBWFQ with a single PQ, which receives strict scheduling priority. To go back to airline analogies, this is the equivalent of preboarding courtesies that are often offered to persons with special needs or those traveling with small children. In spite of the fact that these people may not be in first class, or elite frequent fliers, they are moved directly to the front of the line and put on the plane first because they have special needs. In the case of VoIP traffic, it may not be the most important traffic on your network, but it has very specific requirements for delay and jitter and, therefore, must be moved to the front of the line for transmission.

Catalyst switches use classification to appropriate queuing frames for transmission. Although Catalyst switches only support the Cisco IOS features WFQ, CBWFQ, and LLQ on WAN interfaces, Ethernet interfaces use similar forms of queuing but vary in configuration and behavior.

Scheduling

Scheduling allows resource sharing, specifically bandwidth, among classes of traffic or queues; this scheduling becomes more important as congestion increases. On Cisco switching platforms, frames are scheduled in several ways. The most commonly discussed is *weighted round-robin* (WRR). Because the functionality and configuration of WRR on each specific platform is discussed in the chapter for that platform, this chapter does not attempt to explain the functionality of WRR. Instead, this section previews what is to come in future chapters. For additional information about WRR and its implementation on Catalyst switches, refer to Chapter 6, "QoS Features Available on the Catalyst 2950 and 3550 Family of Switches," Chapter 7, "QoS Features Available on the Catalyst 4000 IOS Family of Switches and the Catalyst G-L3 Family of Switches," and Chapter 8, "QoS Support on the Catalyst 6500."

Congestion Avoidance

In contrast to congestion management, which deals with congestion that already exists, congestion avoidance mechanisms are designed to prevent interfaces or queues from becoming congested in the first place. The actual methods used to avoid congestion are discussed shortly, but keep in mind that the entire concept of congestion avoidance is based on the presence of TCP traffic.

To understand why the benefits of congestion avoidance are only realized when the majority of network traffic is TCP, it is important to understand the fundamental behavior of TCP. This overview is intended to provide just such an understanding, and is not intended to be a comprehensive discussion of TCP's behavior.

Unlike UDP, which has no acknowledgment mechanism, with TCP, when a packet is received, the receiver acknowledges receipt of that packet by sending a message back to the sender. If no acknowledgment is received within a period of time, the sender first assumes that data is being sent too rapidly and reduces the TCP window size, and then the sender resends the unacknowledged packet(s). Gradually, the sender increases the rate at which packets are being sent by increasing the window size.

Without congestion avoidance, when a particular queue becomes completely full, something called*tail drop* occurs. Tail drop is the term used to describe what happens when the (n +1) packet arrives at a queue that is only capable of holding n packets. That packet is dropped and, consequently, no acknowledgment is sent to the sender of that packet. As previously mentioned, this causes a reduction in the window size and a retransmission.

Tail drop presents several problems, however. Most troublesome is the fact that tail drop does not use any intelligence to determine which packet(s) should be dropped; rather the (n + 1) packet is just dropped, regardless of what type of packet it is. This could mean that a packet of the most important traffic type in your network is dropped, whereas a packet from the least important traffic type is transmitted. Another problem associated with tail drop is global synchronization, which reduces the overall throughput of a link with many flows (but is beyond the scope of this discussion).

All Catalyst switches experience tail drop when transmit queues become full. However, several Catalyst switches support configuration of tail-drop thresholds among various queues to minimize full conditions with higher-priority traffic. Chapters 6 and 8 discuss tail-drop thresholds and configuration options available on the Catalyst 3550 Family and Catalyst 6500 Family of switches, respectively.

Random Early Detection (RED)

The problems of tail drop and global synchronization can both be addressed with congestion avoidance. Congestion avoidance is sometimes called *active queue management*, or *Random Early Detection* (RED). The Introduction section of RFC 2309 defines the need for active queue management as follows:

> The traditional technique for managing router queue lengths is to set a maximum length (in terms of packets) for each queue, accept packets for the queue until the maximum length is reached, then reject (drop) subsequent incoming packets until the queue decreases because a packet from the queue has been transmitted. This technique is known as "tail drop", since the packet that arrived most recently (i.e., the one on the tail of the queue) is dropped when the queue is full. This method has served the Internet well for years, but it

has two important drawbacks.

1. Lock-Out

In some situations tail drop allows a single connection or a few flows to monopolize queue space, preventing other connections from getting room in the queue. This "lock-out" phenomenon is often the result of synchronization or other timing effects.

2. Full Queues

The tail drop discipline allows queues to maintain a full (or, almost full) status for long periods of time, since tail drop signals congestion (via a packet drop) only when the queue has become full. It is important to reduce the steady-state queue size, and this is perhaps queue management's most important goal.

The naive assumption might be that there is a simple tradeoff between delay and throughput, and that the recommendation that queues be maintained in a "non-full" state essentially translates to a recommendation that low end-to-end delay is more important than high throughput. However, this does not take into account the critical role that packet bursts play in Internet performance. Even though TCP constrains a flow's window size, packets often arrive at routers in bursts [Leland94]. If the queue is full or almost full, an arriving burst will cause multiple packets to be dropped. This can result in a global synchronization of flows throttling back, followed by a sustained period of lowered link utilization, reducing overall throughput.

The point of buffering in the network is to absorb data bursts and to transmit them during the (hopefully) ensuing bursts of silence. This is essential to permit the transmission of bursty data. It should be clear why we would like to have normally-small queues in routers: we want to have queue capacity to absorb the bursts. The counter-intuitive result is that maintaining normally-small queues can result in higher throughput as well as lower end-to-end delay. In short, queue limits should not reflect the steady state queues we want maintained in the network; instead, they should reflect the size of bursts we need to absorb.

Cisco IOS Software devices implement RED as *WeightedRED* (WRED), which serves the same purpose as RED, but does so with preferential treatment given to packets based on their IP precedence or DSCP value. If it is not desirable in your network to give different treatment to packets of different IP precedence or DSCP values, it is possible to modify the configuration so that all packets are treated the same.

The basic configuration for WRED in a Cisco IOS router is fairly simple:

```
Router(config)#interface Serial 6/0

Router(config-if)#random-detect
```

With only that configuration, IP precedence-based WRED is enabled, and all the defaults are accepted. The configuration can be verified with the show queueing interface command as demonstrated in Example 2-1.

Example 2-1. Verifying the WRED Configuration on a Cisco IOS Router Interface

```
Router#show queueing interface serial 6/0

Interface Serial6/0 queueing strategy: random early detection (WRED)

    Exp-weight-constant: 9 (1/512)

    Mean queue depth: 0


class     Random drop      Tail drop    Minimum Maximum  Mark

          pkts/bytes       pkts/bytes    thresh  thresh  prob

0            0/0              0/0           20      40  1/10

1            0/0              0/0           22      40  1/10

2            0/0              0/0           24      40  1/10

3            0/0              0/0           26      40  1/10

4            0/0              0/0           28      40  1/10

5            0/0              0/0           31      40  1/10

6            0/0              0/0           33      40  1/10

7            0/0              0/0           35      40  1/10

rsvp         0/0              0/0           37      40  1/10
```

There are several values shown in the show queueing output that have not yet been explained.

- Minimum threshold— When the average queue depth exceeds the minimum threshold, packets start to be discarded. The rate at which packets are dropped increases linearly until the average queue depth hits the maximum threshold.
- Maximum threshold— When the average queue size exceeds the maximum threshold, all packets are dropped.
- Mark probability denominator— This number is a fraction and represents the fraction of

packets that are dropped when the queue size is at the maximum threshold. In the preceding example, the mark probability denominator indicates that all IP precedence levels will have 1 of every 10 packets dropped when the average queue depth equals the maximum threshold.

In the show queueing output, notice that the minimum threshold is different for each IP precedence level. It was previously mentioned that it is possible to modify the configuration so that all IP precedence or DSCP values are treated the same. Example 2-2 shows one way to do this.

Example 2-2. Configuring WRED Parameters on a Cisco IOS Router Interface

```
Router(config)#interface s6/0

Router(config-if)#random-detect precedence 1 ?

  <1-4096>  minimum threshold (number of packets)


Router(config-if)#random-detect precedence 1 20 ?

  <1-4096>  maximum threshold (number of packets)


Router(config-if)#random-detect precedence 1 20 40 ?

  <1-65535>  mark probability denominator

  <cr>


Router(config-if)#random-detect precedence 1 20 40 10

Router(config-if)#random-detect precedence 2 20 40 10

Router(config-if)#random-detect precedence 3 20 40 10

Router(config-if)#random-detect precedence 4 20 40 10

Router(config-if)#random-detect precedence 5 20 40 10

Router(config-if)#random-detect precedence 6 20 40 10

Router(config-if)#random-detect precedence 7 20 40 10

Router(config-if)#exit

Router(config)#exit
```

```
Router#show queueing interface s6/0
Interface Serial6/0 queueing strategy: random early detection (WRED)
    Exp-weight-constant: 9 (1/512)
    Mean queue depth: 0

class     Random drop      Tail drop    Minimum Maximum  Mark
          pkts/bytes       pkts/bytes    thresh  thresh  prob
0            0/0              0/0           20      40  1/10
1            0/0              0/0           20      40  1/10
2            0/0              0/0           20      40  1/10
3            0/0              0/0           20      40  1/10
4            0/0              0/0           20      40  1/10
5            0/0              0/0           20      40  1/10
6            0/0              0/0           20      40  1/10
7            0/0              0/0           20      40  1/10
rsvp         0/0              0/0           37      40  1/10
```

As you can see from the show queueing output, the minimum threshold, maximum threshold, and mark probability denominator are now the same for all IP precedence values. This is not necessarily recommended; instead, it is shown to illustrate that the treatment of packets is entirely user configurable.

It is also possible to use DSCP-based WRED, and the configuration options for that differ slightly, as demonstrated in Example 2-3.

Example 2-3. Configuring DSCP-based WRED on a Cisco IOS Router Interface

```
Router(config)#interface s6/0
Router(config-if)#random-detect ?
  dscp-based  Enable dscp based WRED on an interface
prec-based  Enable prec based WRED on an interface
```

```
  <cr>
Router(config-if)#random-detect dscp-based
Router(config-if)#random-detect ?
  dscp                            parameters for each dscp value
  dscp-based                      Enable dscp based WRED on an interface
exponential-weighting-constant  weight for mean queue depth calculation
  flow                            enable flow based WRED
  prec-based                      Enable prec based WRED on an interface
  precedence                      parameters for each precedence value
  <cr>

Router(config-if)#random-detect dscp ?
  <0-63>   Differentiated services codepoint value
  af11     Match packets with AF11 dscp (001010)
  af12     Match packets with AF12 dscp (001100)
  af13     Match packets with AF13 dscp (001110)
  af21     Match packets with AF21 dscp (010010)
  af22     Match packets with AF22 dscp (010100)
  af23     Match packets with AF23 dscp (010110)
  af31     Match packets with AF31 dscp (011010)
  af32     Match packets with AF32 dscp (011100)
  af33     Match packets with AF33 dscp (011110)
  af41     Match packets with AF41 dscp (100010)
  af42     Match packets with AF42 dscp (100100)
  af43     Match packets with AF43 dscp (100110)
  cs1      Match packets with CS1(precedence 1) dscp (001000)
  cs2      Match packets with CS2(precedence 2) dscp (010000)
  cs3      Match packets with CS3(precedence 3) dscp (011000)
```

```
  cs4      Match packets with CS4(precedence 4) dscp (100000)
  cs5      Match packets with CS5(precedence 5) dscp (101000)
  cs6      Match packets with CS6(precedence 6) dscp (110000)
  cs7      Match packets with CS7(precedence 7) dscp (111000)
  default  Match packets with default dscp (000000)
  ef       Match packets with EF dscp (101110)
  rsvp     rsvp traffic

Router(config-if)#random-detect dscp af11 ?
  <1-4096>  minimum threshold (number of packets)

Router(config-if)#random-detect dscp af11 20 ?
  <1-4096>  maximum threshold (number of packets)

Router(config-if)#random-detect dscp af11 20 40 ?
  <1-65535>  mark probability denominator
  <cr>

Router(config-if)#random-detect dscp af11 20 40 10 ?
  <cr>

Router(config-if)#random-detect dscp af11 20 40 10
Router(config-if)#exit
Router(config)#exit
Router#show queueing interface s6/0
Interface Serial6/0 queueing strategy: random early detection (WRED)
    Exp-weight-constant: 9 (1/512)
    Mean queue depth: 0
```

dscp	Random drop	Tail drop	Minimum	Maximum	Mark
	pkts/bytes	pkts/bytes	thresh	thresh	prob
af11	0/0	0/0	20	40	1/10
af12	0/0	0/0	28	40	1/10
af13	0/0	0/0	24	40	1/10
af21	0/0	0/0	33	40	1/10
af22	0/0	0/0	28	40	1/10
af23	0/0	0/0	24	40	1/10
af31	0/0	0/0	33	40	1/10
af32	0/0	0/0	28	40	1/10
af33	0/0	0/0	24	40	1/10
af41	0/0	0/0	33	40	1/10
af42	0/0	0/0	28	40	1/10
af43	0/0	0/0	24	40	1/10
cs1	0/0	0/0	22	40	1/10
cs2	0/0	0/0	24	40	1/10
cs3	0/0	0/0	26	40	1/10
cs4	0/0	0/0	28	40	1/10
cs5	0/0	0/0	31	40	1/10
cs6	0/0	0/0	33	40	1/10
cs7	0/0	0/0	35	40	1/10
ef	0/0	0/0	37	40	1/10
rsvp	0/0	0/0	37	40	1/10
default	0/0	0/0	20	40	1/10

Theshow queueing output for af21, af31, and af41 indicates that the minimum threshold for af11 would have been 33 by default. As shown, the minimum threshold is 20, based on commands entered.

NOTE

The af*xx* values shown in the show queueing output are related to the *assured forwarding per-hop behavior* (AF PHB) discussed in Chapter 1. From the show queueing output, you can see that afx3 packets will be dropped before afx2 packets, and afx2 packets will be dropped before afx1 packets. This behavior was explained in detail in Chapter 1.

Incidentally, the term *average queue depth* has been used several times during this discussion of WRED and, although a detailed explanation of this formula is beyond the scope of this chapter, the formula used to calculate average queue depth is as follows:

Average = (old_average * (1 – 2^{-n})) + (current_queue_size * 2^{-n}), where *n* is the exponential weight factor, which is user configurable.

CAUTION

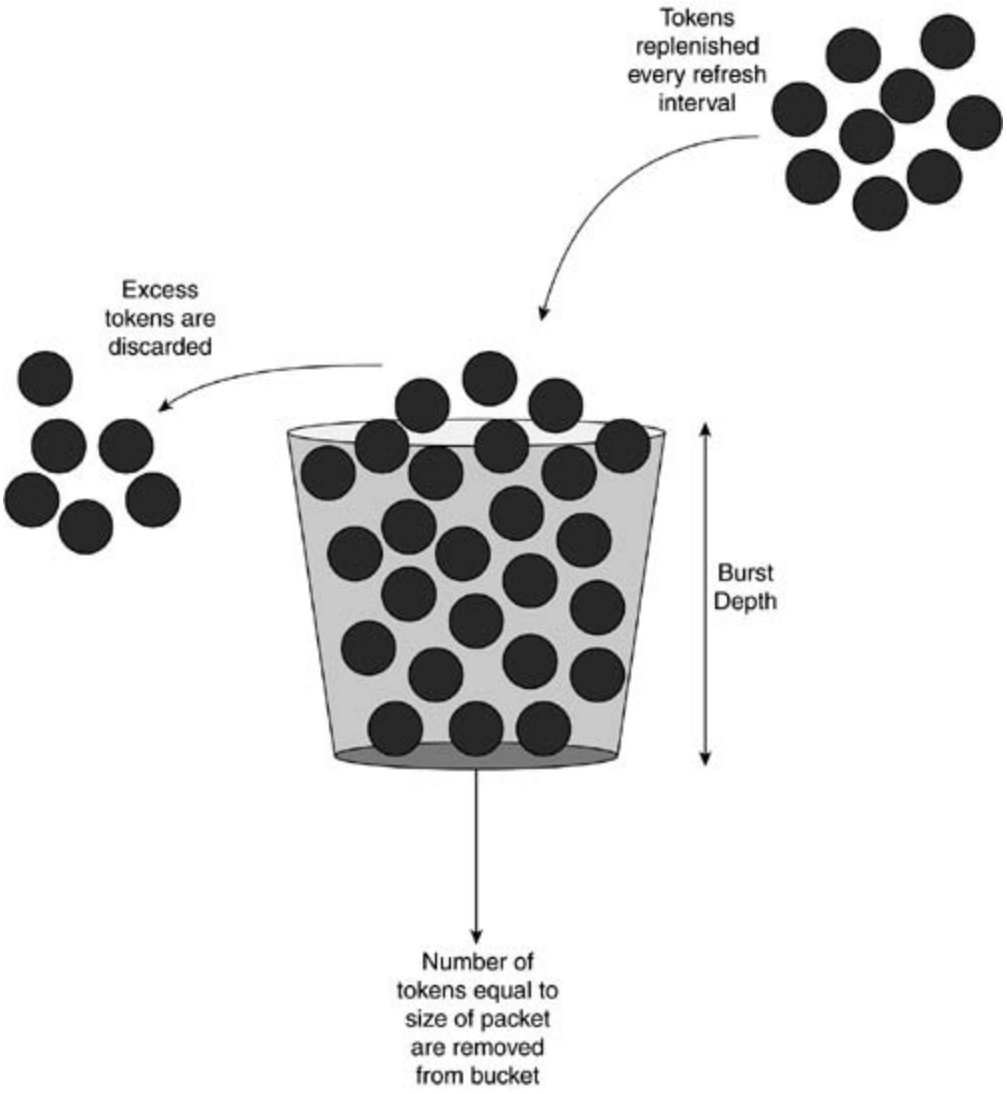
If the value for *n* is set too high, WRED will not work properly and the net impact will be roughly the same as if WRED were not in use at all.

At time of publication, the Catalyst 3550 Family and Catalyst 6500 Family of switches support the WRED congestion management feature. Refer to the congestion management section in each product chapter for the Catalyst 3550 Family and Catalyst 6500 Family of switches for more details.

Token Bucket Mechanism

A*token bucket* is a formal definition of a rate of transfer. For this discussion, assume the token bucket starts full. This implies the maximum amount of tokens is available to sustain incoming traffic. Assume a bucket, which is being filled with tokens at a rate of *x* tokens per refresh interval. Each token represents 1 bit of data. To successfully transmit a packet, there must be a one-to-one match between bits and tokens. As a result, when a packet or frame arrives at the port or interface, and enough tokens exist in the bucket to accommodate the entire unit of data, the packet conforms to the contract, and therefore is forwarded. When the packet is successfully transmitted, the number of tokens equal to the size of the transmitted packet is removed from the bucket. Figure 2-1 illustrates the token bucket mechanism.

Figure 2-1. Token Bucket Mechanism



If the actual ingress traffic rate exceeds the configured rate, and there are insufficient tokens in the token bucket to accommodate the arriving traffic, the excess data is considered out-of-profile and can be dealt with in one of two ways:

- Re-assign QoS values to appropriate header
- Drop packet

If the decision is to mark down the nonconforming traffic, the DSCP value is derived from the mapping tables.

The token bucket mechanism has three components: a burst size, a mean rate, and a time interval (Tc). Although the mean rate is generally represented as bits per second, any two values may be derived from the third by the relation shown as follows:

Mean rate = burst size / time interval

Here are some definitions of these terms:

- Mean rate— Also called the *committed information rate* (CIR), it specifies how much data can be sent or forwarded per unit time on average.
- Burst size— Also called the *committed burst* (Bc) size, it specifies in bits (or bytes) per burst how much traffic can be sent within a given unit of time to not create scheduling concerns. (For a shaper, such as *generic traffic shaping* (GTS), it specifies bits per burst; for a policer, such as *committed access rate* (CAR), it specifies bytes per burst.)
- Time interval— Also called the *measurement interval*, it specifies the time quantum in seconds per burst.

By definition, over any integral multiple of the interval, the bit rate of the interface will not exceed the mean rate. The bit rate, however, may be arbitrarily fast within the interval.

A token bucket is used to manage a device that regulates the data in a flow. For example, the regulator might be a traffic policer, such as CAR, or a traffic shaper, such as *Frame Relay traffic shaping* (FRTS) or GTS. A token bucket itself has no discard or priority policy. Rather, a token bucket discards tokens and leaves to the flow the problem of managing its transmission queue if the flow overdrives the regulator. (Neither CAR nor FRTS and GTS implement either a true token bucket or true leaky bucket.)

In the token bucket metaphor, tokens are put into the bucket at a certain rate. The bucket itself has a specified capacity. If the bucket fills to capacity, newly arriving tokens are discarded. Each token is permission for the source to send a certain number of bits into the network. To send a packet, the regulator must remove from the bucket a number of tokens equal in representation to the packet size.

If not enough tokens are in the bucket to send a packet, the packet either waits until the bucket has enough tokens (in the case of GTS) or the packet is discarded or marked down (in the case of CAR). If the bucket is already full of tokens, incoming tokens overflow and are not available to future packets. Thus, at any time, the largest burst a source can send into the network is roughly proportional to the size of the bucket.

Note that the token bucket mechanism used for traffic shaping has both a token bucket and a data buffer, or queue; if it did not have a data buffer, it would be a policer. For traffic shaping, packets that arrive that cannot be sent immediately are delayed in the data buffer.

For traffic shaping, a token bucket permits burstiness but bounds it. It guarantees that the burstiness is bounded so that the flow never sends faster than the capacity of the token bucket plus the time interval multiplied by the established rate at which tokens are placed in the bucket. It also guarantees that the long-term transmission rate does not exceed the established rate at which tokens are placed in the bucket.

All Catalyst switches that support policing utilize the token bucket algorithm for bandwidth-limiting traffic flows. Chapters 6,7, and 8 discuss the token bucket algorithm for the Catalyst 2950 and 3550, 4000 IOS, and 6500 Family of switches, respectively.

Traffic Shaping

Cisco devices utilize various types of traffic shaping, but this section is intended only to introduce the general concept of traffic shaping, not the specific implementations. For information about implementation on Cisco platforms, refer to Cisco.com.

The purpose of traffic shaping is to control the rate at which packets are sent out of an interface, while preventing packet loss. You might choose to use traffic shaping for any number of reasons, but the primary reasons are as follows:

- Data rate mismatch— In Frame Relay networks, it is common to have a main location with a large amount of bandwidth and remote locations with smaller amounts of bandwidth. Because the main location has no idea that the other end of the link has less bandwidth, traffic shaping must be used to control the rate at which traffic is sent to the remote locations.
- Controlling access to bandwidth— You might want to control access to bandwidth for various reasons, even when plenty of bandwidth is available. A good example is tiered-pricing or "burstable" Internet connections that have become popular in recent years. These connections deliver a circuit with a high possible speed (for example, a DS3) but charge based on average utilization of the circuit. So, if your company wants the flexibility to quickly move to a higher rate, but wants to control expenses, ordering this type of circuit and using traffic shaping to make sure that you do not exceed a certain subrate might be a good choice.

The key benefit of traffic shaping is that packet buffering, rather than packet drop, is used to achieve rate control. This is a crucial difference when dealing with drop-sensitive applications. There is also a potential downside to shaping, however, because shaping is accomplished through buffering, which introduces delay. Delay-sensitive traffic should not be shaped, unless the potential impact is fully understood and acceptable in your environment.

The decision of whether a packet should be sent immediately or buffered until the next time interval is based on the token bucket scheme that was introduced in the preceding section.

Policing

The purpose of the policer is to control the amount of bandwidth consumed by an individual microflow, or an aggregate of flows. Do not confuse policing with traffic shaping. Although traffic shaping limits flows to a specified rate, it buffers nonconforming traffic to ensure the flows remain within the confines of the configured contract. Policing, on the other hand, does not buffer nonconforming traffic. As a result, policing does not introduce additional jitter or delay, which could impact the timely delivery and performance of real-time voice or video applications.

In contrast to traffic shaping, policing accomplishes rate control by dropping packets. The two primary mechanisms for traffic policing in Cisco IOS are CAR and class-based policer. Cisco Catalyst platforms do not support CAR, but do support a policing function. Depending on the platform and supervisor engine, policing may be supported ingress only or both ingress and egress. In addition, policing may be aggregate or per-flow. On non-Catalyst platforms, policing is currently aggregate policing, although there are rumors that per-flow policing will be available sometime in the future.

It is important to understand that policing leads to packet loss, and to understand the impact of dropped packets on the application being policed. In the case of an application such as FTP, which is TCP-based and very tolerant of dropped packets, policing has very little impact other than the obvious impact of slowing the traffic rate. With an application such as IP-based video conferencing, however, dropped packets may cause poor picture and sound quality, so policing should be used with caution.

Networks use policing for dozens of reasons, but several of the most common reasons are as follows:

- Subrate delivery— Local Internet service providers may sell something less than a T1 connection (for example, 384 kbps), but are forced to deliver that over a full T1. Some providers choose to use a Frame Relay service that enables them to handle this problem another way, but it is often less expensive for the ISP to just use a point-to-point T1 and police the traffic coming into the provider's network. This also allows for an extremely simple bandwidth upgrade if the customer calls and requests more bandwidth.

- Rate control of recreational traffic— It is not uncommon these days for networks with no controls to be overrun with peer-to-peer file-sharing services such as Napster and Kazaa. Many networks have chosen to implement policing to limit the impact that these nonbusiness applications can have on their network.

- Network security— QoS mechanisms should never be viewed as an alternative to network security devices but, when used to compliment network security devices, QoS mechanisms can add to the overall security of the network. For example, policing ping traffic to a reasonable amount (what is considered reasonable depends on the network) will not stop someone from launching *Internet Control Message Protocol* (ICMP)-based attacks on your network, but will be able to drop a good portion of the attack traffic as it enters your network, thereby limiting the impact that the attack has on internal resources.

- Safety mechanism— If your network is implementing *connection admission control* (CAC) to make sure that only two concurrent video sessions can be established over a single link, there should not ever be more than two sessions in progress. As a safety mechanism, to prevent rogue video traffic from dominating the link in the event of a failure of your CAC device, a policer might be used to limit the total amount of video to equal the bandwidth required for two concurrent sessions. It is important to note that, if the CAC mechanism

failed, policing would police the aggregate of the video traffic, which would likely impact video quality. Although poor video quality is not ideal, other network traffic would be protected.

As mentioned earlier, Cisco IOS has two main mechanisms for policing. CAR is a legacy mechanism and, although a lot of networks use CAR today, future development efforts will be put into class-based policer. Class-based policer was originally released to correspond to RFC 2697, which defines a single-rate three-color policer. The rate that is defined is the rate to which traffic should be policed. The three colors refer to the capability to use conform-action, exceed-action, and violate-action to specify different packet markings or actions (such as, drop) that can be configured when packets conform to, exceed, or violate the specified rate. If only the conform-action and exceed-action are specified, a single token bucket is used to perform the policing. If the violate-action is specified, a second token bucket is also used.

With the release of RFC 2698, which defines a two-rate, three-color policer, Cisco IOS version 12.2(4)T added to the functionality of the policing function. With the introduction of two-rate policing, it is now possible to specify both a CIR and a *peak information rate* (PIR).

CAR is not supported on Catalyst switches for shaping traffic between VLANs or Ethernet interfaces. CAR is supported for shaping of traffic across WAN interfaces. Instead, Catalyst switches use policers defined by policies to shape traffic across interfaces.

QoS Signaling

Resource Reservation Protocol (RSVP) is an IP service that enables hosts to make a request for reserved bandwidth along the path to the destination host. Allowing an explicit reservation of bandwidth end-to-end means that the traffic will have a guaranteed QoS.

It is important to understand that RSVP itself does not actually provide the bandwidth; it just signals the router that there is a request. In Cisco devices, RSVP works in conjunction with WFQ and WRED to provide the requested QoS.

Generally speaking, reserving bandwidth is not necessary for data flows other than real-time applications, such as VoIP or IP-based video conferencing. For these traffic types, latency and bandwidth are both critical to performance and it makes sense to have an explicit reservation for those characteristics along the network path.

RSVP works for unicast or multicast, and there are two types of reservations to be concerned with:

- Distinct reservation— This is a flow that originates from one and only one source. Distinct reservations are given a distinct reservation for each source on each link that the traffic crosses. Unicast traffic and "one-to-many" multicast would likely both use distinct reservations.
- Shared reservation— This is a flow that originates from one or more sources. A shared reservation is used by a "many-to-many" multicast application, and there is a single reservation, defined with a wildcard filter, such that any sender in the specified group has access to the reservation.

Subnet Bandwidth Manager (SBM), defined by RFC 2814, is a signaling protocol that deals with RSVP-based admission control on 802-based networks. Today, SBM is primarily used in switched Ethernet environments for LAN-based admission control for RSVP flows. Simply stated, SBM is responsible for handling admission control for resource reservations.

On each managed segment, an election process happens dynamically, and one device becomes the*Designated Subnet Bandwidth Manager* (DSBM). The DBSM is responsible for the admission control on that specific segment. There can be more than one SBM on a segment, but only one is elected as the DSBM.

The actual process of making a reservation, when a DBSM is present, includes the following steps:

1. When a DBSM sends an RSVP Path message out an interface that belongs to a managed segment, the message is sent to the DBSM. If the segment were not managed, this message would have gone directly to the RSVP session destination.
2. The DBSM builds a Path state for the message, and retains information about the node from which it received the message.
3. The DBSM forwards the message to the destination.
4. The DBSM then gets the RSVP Resv message and processes the message to determine what action to take next.

5. If there is bandwidth available for the reservation, the Resv message is sent to the previous hop. In this case, the local Path state is used for this session.

6. If bandwidth isn't available for the reservation, the DBSM returns a Resverr message to the requesting node.

On Cisco IOS routers, you can use the following command to configure SBM:

```
Router(config-if)#ip rsvp dsbm

candidate [priority]
```

Theshow ip rsvp sbm detail command can be used to verify the configuration of SBM.

Link Efficiency

Cisco IOS includes more and more features with every release. Some of the features in Cisco IOS a very specific, and may never need to be used in your network. However, link-efficiency mechanism likely to be of use in any network where there is a mixture of voice and data running over slower-s links (T1 speed or less).

In general, link-efficiency mechanisms are designed to help maximize efficiency when voice is pres the network. Although quite a few mechanisms would be considered part of the group of link-efficie features, only a few are discussed here; primarily, *Real Time Protocol Header Compression* (cRTP) *Link Fragmentation and Interleaving* (LFI).

Link Fragmentation and Interleaving (LFI)

Before discussing the specifics of link-efficiency mechanisms, it seems wise to discuss the need for efficiency in the first place. Primarily, when conversations take place about using VoIP, everyone ju up and screams about the need for QoS or, more specifically, prioritization of traffic. Although prioritization is certainly necessary in a VoIP environment, lower-speed links (those under 768 kbp create problems for VoIP that simple prioritization cannot solve. In these cases, link-efficiency mechanisms are necessary to ensure proper QoS for the VoIP traffic.

Serialization delay is the time that it takes to place bits on to the circuit. Table 2-1 shows the serialization delay for packets of various sizes, by link speed.

Table 2-1. Serialization Delay of Various Packet Sizes on Links of Vario Speeds

Link Speed	Packet Size					
	64 bytes	128 bytes	256 bytes	512 bytes	1024 bytes	1500 byt
56 kbps	9 ms	18 ms	36 ms	72 ms	144 ms	214 ms
64 kbps	8 ms	16 ms	32 ms	64 ms	128 ms	187 ms
128 kbps	4 ms	8 ms	16 ms	32 ms	64 ms	93 ms
256 kbps	2 ms	4 ms	8 ms	16 ms	32 ms	46 ms
512 kbps	1 ms	2 ms	4 ms	8 ms	16 ms	23 ms
768 kbps	0.640 ms	1.28 ms	2.56 ms	5.12 ms	10.24 ms	15 ms

As you can see from this table, the serialization delay for a 1500-byte packet on a link slower than kbps is quite high. The "break point" for a 1500-byte packet happens at 768 kbps or higher speeds When the link speed is 768 kbps, or greater, the serialization delay does not have a significant neg impact, so fragmentation is not necessary.

If the serialization delay would otherwise be greater than 15 ms, however, fragmentation is neede Specifically, LFI is needed. All forms of LFI have the same function, but there are several types of L

- MLP Interleaving— LFI on multilink PPP links
- FRF.12— LFI for Frame Relay data *permanent virtual circuits* (PVCs)
- FRF.11 Annex C— LFI for Frame Relay *Voice over Frame Relay* (VoFR) PVCs

The function of all of these types of LFI is to fragment large data packets and interleave the much smaller data packets. Figure 2-2 illustrates the basic concept.

Figure 2-2. Link Fragmentation and Interleaving

Configuring MLP Interleaving is as simple as using the following three commands:

```
Router(config-if)#encapsulation ppp
Router(config-if)#ppp multilink
Router(config-if)#ppp multilink interleave
```

Optionally, you can manually change the default fragmentation delay from 30 using the following command:

```
Router(config-if)#ppp multilink fragment-delay milliseconds
```

Table 2-2 shows the recommended settings for fragmentation

Table 2-2. Recommended Settings for Fragmentation

Link Speed	10 ms	20 ms	30 ms	40 ms	50 ms	100 ms	200 m
56 kbps	70 bytes	140 bytes	210 bytes	280 bytes	350 bytes	700 bytes	1400 b
64 kbps	80 bytes	160 bytes	240 bytes	320 bytes	400 bytes	800 bytes	1600 b
128 kbps	160 bytes	320 bytes	480 bytes	640 bytes	800 bytes	1600 bytes	3200 b
256 kbps	320 bytes	640 bytes	960 bytes	1280 bytes	1600 bytes	3200 bytes	6400 b
512 kbps	640 bytes	1280 bytes	1920 bytes	2560 bytes	3200 bytes	6400 bytes	12800 bytes
768 kbps	1000 bytes	2000 bytes	3000 bytes	4000 bytes	5000 bytes	10000 bytes	20000 bytes
1536 kbps	2000 bytes	4000 bytes	6000 bytes	8000 bytes	10000 bytes	20000 bytes	40000 bytes
Shaded values indicate that fragmentation is not needed.							

Configuring*Frame Relay Fragmentation* (FRF.12 or FRF.11 Annex C) requires only one command; r
that the following command is entered within a Frame Relay map class, not under the interface dir

```
Router(config-map-class)#frame-relay fragment fragment_size
```

You can verify the configuration of LFI for Frame Relay with the show frame-relay fragment
command. These commands are, at present, only applicable to the FlexWAN module for the Catalys
6500 Family of switches.

Real Time Protocol Header Compression (cRTP)

Real Time Protocol (RTP) is the standard protocol for the transport of real-time data, and is defined
RFC 1889. As defined, an RTP packet includes the payload plus an RTP/UDP/IP header. The total h

is 40 bytes and breaks down as follows: The RTP header is 12 bytes, the UDP header is 8 bytes, an IP header is 20 bytes. Depending on the application, the payload is typically between 20 bytes and bytes. That means that it's possible that the header would be twice the size of the payload! Clearly is not efficient, so the folks at the IETF came up with a way to take advantage of similarities betwe successive packets and reduce the 40-byte header to somewhere between 2 and 5 bytes. For VoIP, header compression can yield an overall reduction in packet size on the order of 200 percent or mc

Although the idea of using cRTP everywhere sounds good at first glance, the trade-off, in terms of performance, on link speeds higher than T1 are not desirable. Therefore, use of cRTP should be lim to links with speeds of T1 or less. Speaking of performance, prior to Cisco IOS 12.0(7)T, cRTP hapı in the process switching path, which often slowed the processing of packets to the point that it was desirable to use cRTP. As of 12.0(7)T, however, cRTP will use the *Cisco Express Forwarding* (CEF) switching path, or fast switching path (if CEF is not enabled). Only if both CEF and fast switching a disabled will cRTP packets be process switched.

The configuration of cRTP on a serial interface is simple, as shown in the following example:

```
Router(config-if)#ip rtp header-compression [passive]
```

Thepassive keyword tells the routers to compress outgoing RTP packets only if the incoming RTP packets on that interface are compressed.

For Frame Relay PVCs, the configuration of cRTP is as follows:

```
router(config-if)#frame-relay map ipip-address dlci [broadcast]rtp header-compress
└active | passive]
```

To verify the configuration and operation of cRTP, use the following commands for Frame Relay an Frame Relay interfaces, respectively:

```
Router#show frame-relay ip rtp header-compression [interface type number]

Router#show ip rtp header-compression [type number] [detail]
```

Classification and Marking at Layer 3

RFC 791 first defined Layer 3 packet marking in terms of IP precedence and *type of service* (ToS), whereas RFC 795 expanded on the service mappings of the TOS bits. The discussion of these RFCs provided here is for historical purposes, and it should be noted that both of these RFCs are obsolete, due to various RFCs that are part of the DiffServ architecture. Refer to Chapter 1 for a discussion of current RFCs that are part of the DiffServ architecture.

IP Precedence and the Type of Service Byte

RFC 791 defines the three most significant bits of the ToS byte as the IP precedence bits, and the next 3 bits as delay, throughput, and reliability. Table 2-3 shows the original definition of the ToS byte.

Table 2-3. Definition of ToS Byte

Bits	Precedence
Bits 0–2	Precedence
Bit 3	0 = normal delay, 1 = low delay
Bit 4	0 = normal throughput, 1 = high throughput
Bit 5	0 = normal reliability, 1 = high reliability
Bits 6–7	Reserved for future use

Keep in mind that the bits are numbered left to right, with the left bits (0–2) being the most significant.Table 2-4 shows the values assigned to the 3 bits used for IP precedence.

Table 2-4. IP Precedence Bits and Their Definitions

Bits	IP Precedence
111	Network control (precedence 7)
110	Internetwork control (precedence 6)
101	CRITIC/ECP (precedence 5)
100	Flash override (precedence 4)
011	Flash (precedence 3)
010	Immediate (precedence 2)
001	Priority (precedence 1)
000	Routine (precedence 0)

The concept of IP precedence became wildly popular and is in use in many networks today. The use of the next three most significant bits (defined for delay, throughput, and reliability) was not as well received for a variety of reasons. For the most part, these bits were not used due to confusion over the proper implementation to deliver the service levels specified. RFC 1349 redefined these 3 bytes and made use of 1 additional bit (bit 6). Table 2-5 shows how RFC 1349 defined what were now the four types of service bits.

Table 2-5. Definition of ToS Bits in RFC 1349

Bits	Precedence
1000	Minimize delay
0100	Maximize throughput
0010	Maximize reliability
0001	Minimize monetary cost
0000	Normal service

Unfortunately, this new definition was not implemented much more than the previous definition of these bits. Because these bits are not commonly used, and have been made obsolete due to later RFCs, they are not discussed here. You can find more information about the service mappings in RFCs 795 and 1349.

The Differentiated Services Field (DS Field)

As stated in RFC 2474, DiffServ enhancements to IP are designed to allow service differentiation on the Internet in a scalable manner, primarily because per-flow classification will not be necessary at each hop. There is a great deal more explained in RFC 2474 than is covered here. The purpose of this section is only to discuss packet marking, using the DS field.

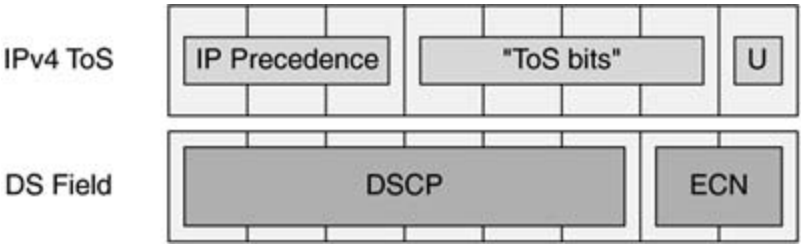
The idea behind DiffServ is that packets will be marked (much like with IP precedence) at the edge of a network and those markings will then be read by each router (hop) that a packet

passes through. Based on the specific rules that have been defined within that hop, the packet is assigned to a traffic class and that traffic class will have been given some amount of access to scheduling and other resources.

With IP precedence, only the 3 left-most bits were used for IP precedence, and the next 3 bits were rarely used. So RFC 2474 redefines the ToS byte to make use of the 6 left-most bits (bits 0–5) for packet marking. Doing so increases the number of possible packet markings from 8 to 64 and eliminates "wasted" bits (those that were rarely used). The 2 least significant bits of the ToS byte are not defined by RFC 2474, but have since been defined by RFC 2481. RFC 2481 is beyond the scope of this discussion, but is an interesting concept called *explicit congestion notification* (ECN). You can view RFC 2481 (and all other RFCs) at www.ietf.org, under the RFC Pages section.

Figure 2-3 illustrates the ToS byte with IP precedence and the redefined ToS byte with DiffServ:

Figure 2-3. ToS Byte: With IP Precedence and Differentiated Services



Per-Hop Behaviors

With the introduction of the DSCP markings, there were significantly more possible markings for packets (0–63 are the possible markings for packets). Because there were so many more possible markings, the IETF decided to standardize what some of the codepoints meant. In part, this is to provide backward compatibility to IP precedence and, in part, this is to facilitate certain types of behaviors that were seen as fundamental to the DiffServ architecture.

The following definition of a per-hop behavior is taken from Section 2.4 of RFC 2475:

> A per-hop behavior (PHB) is a description of the externally observable forwarding behavior of a DS node applied to a particular DS behavior aggregate … In general, the observable behavior of a PHB may depend on certain constraints on the traffic characteristics of the associated behavior aggregate, or the characteristics of other behavior aggregates.

For a definition of forwarding behavior, refer to Chapter 1.

For the purpose of backward compatibility, it is important to note that a router that understands IP precedence, but is not DiffServ-aware, looks only at the 3 most significant bits (left-most bits) in the ToS field, whereas a DiffServ-capable router looks at the 6 most significant bits. Therefore, the marking 000000 is defined in RFC 2474 as Best Effort. This means that a router that only understands IP precedence would read that field as 000, which is the definition of Best Effort in an IP precedence environment. Similarly, all markings in the format *xxx*000 are reserved by RFC 2474 to provide backward compatibility with IP precedence values, as shown in Table 2-6.

Table 2-6. Backward Compatibility with IP Precedence Values

Bits	Precedence
001000	Class selector 1; read as 001 by a non-DiffServ node and treated like IP precedence 1.
010000	Class selector 2; read as 010 by a non-DiffServ node and treated like IP precedence 2.
011000	Class selector 3; read as 011 by a non-DiffServ node and treated like IP precedence 3.
100000	Class selector 4; read as 100 by a non-DiffServ node and treated like IP precedence 4.
101000	Class selector 5; read as 101 by a non-DiffServ node and treated like IP precedence 5.
110000	Class selector 6; read as 110 by a non-DiffServ node and treated like IP precedence 6.
111000	Class selector 1; read as 111 by a non-DiffServ node and treated like IP precedence 7.

These backward-compatible markings were necessary to not break networks while the

conversion from IP precedence to DiffServ was taking place.

RFC 2597: The Assured Forwarding PHB

Other than those defined in RFC 2474, there are two main PHBs, RFC 2597 defines the first of these. It is called the *assured forwarding* (AF) PHB, and the concept behind the PHB is to provide a level of assurance as to a given packet's probability of being forwarded during congestion.

RFC 2597 defines four classes, and each class is completely independent of the other classes. In addition, each class has three level of "drop precedence" to which packets of that class can be assigned.

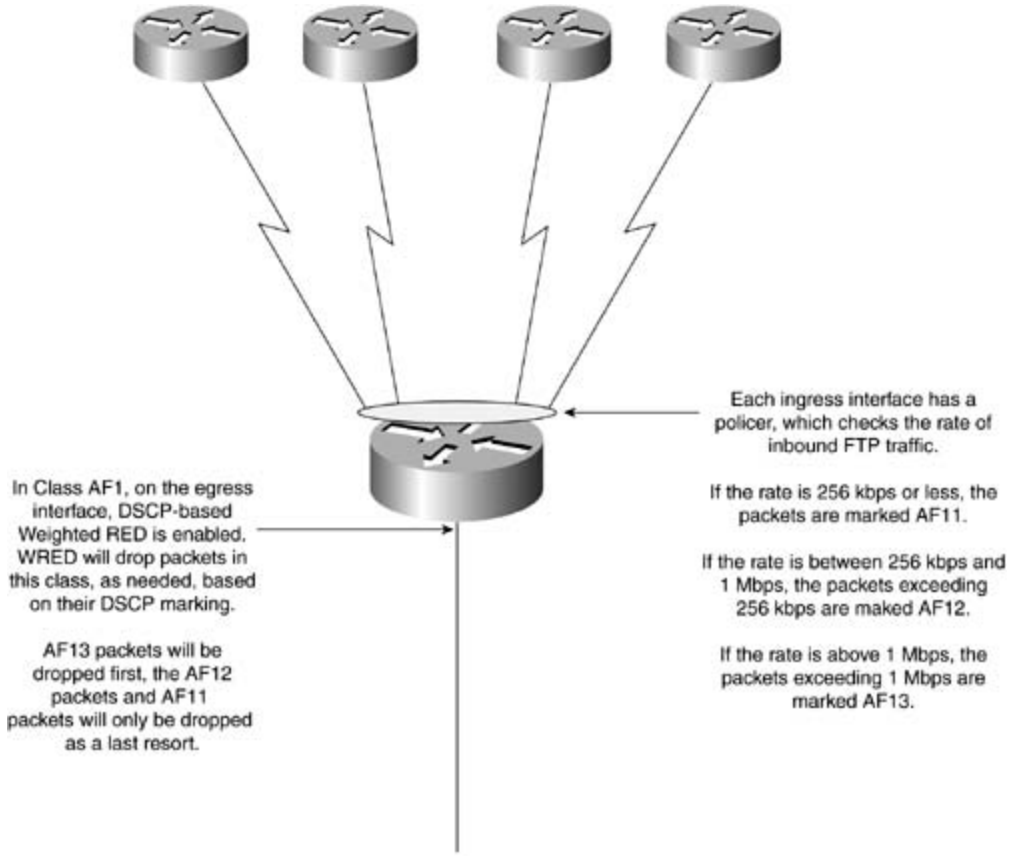
From a high level, the concept is that you can have four different classes of traffic and, within those classes, you can have three different levels of probability that a packet will be dropped if that class becomes congested. Table 2-7 shows the code point markings for the AF PHB and the decimal values associated with each.

Table 2-7. Codepoint Markings for the AF PHB

Drop Precedence	Class 1	Class 2	Class 3	Class 4
Low	AF11 (001010) 10	AF21 (010010) 18	AF31 (011010) 26	AF41 (100010) 34
Medium	AF12 (001100) 12	AF22 (010100) 20	AF32 (011100) 28	AF42 (100100) 36
High	AF13 (001110) 14	AF23 (010110) 22	AF33 (011110) 30	AF43 (100110) 38

To understand a possible use for the AF PHB, assume that you have four branch offices aggregating into a single router. Each branch has been told that they should not send more than 256 kbps of FTP traffic, but sometimes they do anyway. In fact, sometimes they send more than 1 Mbps of FTP traffic. The problem is that when one branch sends a lot of FTP traffic, it sometimes interferes with another branch's FTP traffic, even if they are sending at or below the 256-kbps limit. A possible solution for this problem can be found by using the AF PHB, as illustrated in Figure 2-4.

Figure 2-4. An Example of the AF PHB



In Cisco devices, this PHB is implemented on egress interfaces using a combination of CBWFQ and DSCP-based class-based WRED. Each site has contracted for 256 kbps of FTP traffic, so the bandwidth statement reflects a minimum reservation of 1024 kbps. The entire configuration of the egress interface seen in Figure 2-4 would be entered as shown in Example 2-4.

Example 2-4. Configuring CBWFQ and Class-based WRED

```
Router(config)#class-map match-any FTP
Router(config-cmap)#match ip dscp af11
Router(config-cmap)#match ip dscp af12
Router(config-cmap)#match ip dscp af13
Router(config-cmap)#exit
Router(config)#policy-map AF-PHB
Router(config-pmap)#class FTP
Router(config-pmap-c)#bandwidth 1024
```

```
Router(config-pmap-c)#random-detect dscp-based

Router(config-pmap-c)#exit

Router(config-pmap)#exit

Router(config)#exit

Router#
```

The running-config for this would then appear as in Example 2-5.

Example 2-5. Verifying the Configuration of CBWFQ and Class-based WRED

```
!
class-map match-any FTP
  match ip dscp af11
  match ip dscp af12
  match ip dscp af13
!
!
policy-map AF-PHB
  class FTP
   bandwidth 1024
   random-detect dscp-based
```

When attached to an interface, the functionality of this policy can be verified with the show policy-map interface command.

RFC 2598: The Expedited Forwarding PHB

The*expedited forwarding* (EF) PHB is designed to build a low-loss, low-latency, and low-jitter, assured bandwidth class. If all of those characteristics sound a bit like how you would want to treat VoIP traffic, you are already well on your way to understanding the purpose of this PHB.

Although RFC 2598 does not specifically say that this PHB should be used only for voice, this is the likely service for which this PHB would be used. Other types of interactive traffic could no doubt benefit from this type of treatment, however.

The idea behind the EF PHB is that traffic of this class should be forwarded with strict priority, as soon as that traffic is received. Note that the EF PHB should not normally be used for TCP traffic, due to the bursty nature of TCP. The reason behind this recommendation is that the EF PHB, in order not to starve other traffic classes of bandwidth, implements a policing function. This policing function ensure that the EF class receives all the bandwidth to which it is entitled, but is limited in the total amount of bandwidth that it can consume.

The EF PHB is implemented in Cisco devices as LLQ. Note that prior to Cisco IOS version 12.2, the policing function was implemented differently for different media types and, in some cases, the policing function happened regardless of congestion. In newer versions of Cisco IOS, however, this inconsistency has been resolved and the policing function of the LLQ only happens in the event of congestion on the interface.

If added to the configuration already shown for the AF PHB, the configuration for the EF PHB would be as shown in Example 2-6.

Example 2-6. Configuring LLQ (EF PHB)

```
Router(config)#class-map VOICE

Router(config-cmap)#match ip dscp ef

Router(config-cmap)#exit

Router(config)#policy-map AF-PHB

Router(config-pmap)#class VOICE

Router(config-pmap-c)#priority 56

Router(config-pmap-c)#exit

Router(config-pmap)#exit

Router(config)#exit

Router#
```

Notice that, in this example, the policy map is still named AF-PHB. Because this name was already being used for the interface, the EF configuration was just added to this policy. Of course, the name of the policy doesn't really matter.

The running-config for the entire policy would now appear as shown in Example 2-7.

Example 2-7. Verifying the Configuration of LLQ and WRED

```
class-map match-any FTP
  match ip dscp af11
  match ip dscp af12
  match ip dscp af13
class-map match-all VOICE
  match ip dscp ef
!
!
policy-map AF-PHB
  class FTP
   bandwidth 1024
   random-detect dscp-based
  class VOICE
    priority 56
```

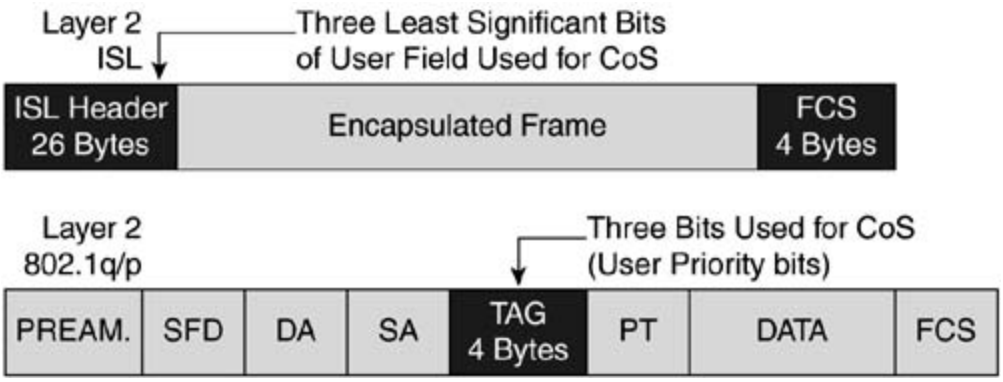
This example reserves 56 kbps of bandwidth for the LLQ, which means that, under congestion, there will also be a policer that limits the VOICE class to 56 kbps. Again, when applied to an interface, this policy can be verified with the show policy-map interface command. The use of this command is currently limited to the FlexWAN module.

Classification and Marking at Layer 2

Before the days of VoIP and IP-based video conferencing, Layer 3 packet marking was the only packet marking that was ever discussed. Seemingly, Layer 3 markings were all that would ever be needed; the reasoning behind that line of thought was that the links in the switched environment were so fast that there would never be a need to prioritize access to bandwidth. Of course, VoIP and other real-time traffic also have very low tolerance for delay or jitter. Therefore, a clear need exists to place bounds on the amount of delay and jitter that a given packet will experience in the switched network.

A method of marking frames at Layer 2 was developed to allow differentiation of frames in terms of the characteristics that those frames would receive as they traverse the switched infrastructure. However, the only way to mark frames at Layer 2 is in the ISL or 802.1Q header. The location of bits used for this purpose is different for ISL and 802.1Q, as illustrated in Figure 2-5.

Figure 2-5. The Layer 2 Class of Service Bits



Mapping Layer 2 to Layer 3 Values

As frames/packets move from the Layer 2 environment to a Layer 3 environment, the ISL or 802.1Q header is lost. To preserve end-to-end QoS, this loss creates a need for the ability to map Layer 2 CoS values to Layer 3 ToS values (either IP precedence or DSCP). As frames/packets move back from the routed network to the switched network, ISL or 802.1Q headers will again be able to carry Layer 2 QoS markings. This is especially important when Layer 2 devices that have no Layer 3 capabilities are involved, because they may understand CoS markings, but not ToS markings. Again, to preserve end-to-end QoS, it would be necessary to use the Layer 3 marking to mark a Layer 2 CoS value on these downstream packets.

The commands for handling these mappings differs between platforms, but the command for setting a CoS-to-DSCP map on the Catalyst 6500 Family of switches is as follows:

```
set qos cos-dscp-map {dscp1, dscp2, dscp3, dscp4, dscp5, dscp6, dscp7, dscp8}
```

The variables *dscp1* through *dscp8* represent the DSCP values that you would enter. The seven positions shown correspond to the seven nonzero CoS values.

Because there are only eight CoS values, it is easy to map CoS to DSCP, but slightly more complicated when you map DSCP to CoS, because it's not possible to map all 64 DSCP values to an individual CoS value. As such, groups of DSCP values must be mapped to a single CoS value. On the Catalyst 6500 Family of switches, that configuration is as follows:

```
set qos dscp-cos-map 20-25:7 33-38:3
```

This example maps DSCP values 20 to 25 to CoS value 7 and DSCP values 33 to 38 to CoS value 3. It is a little confusing at first, but makes more sense as you become familiar with the syntax.

There are also default mappings in most switching platforms, but you should consult the chapters in this book that deal directly with the platforms that you are working with to determine what those defaults are and whether they are suitable for your network.

A General View of QoS on the Catalyst Platforms

This chapter mostly focuses on very detailed descriptions of the exact functionality of QoS on specific platforms. This section, however, provides a general overview of QoS concepts and a glimpse into the material covered in the later chapters. This section purposely avoids details about the operation of specific mechanisms on Cisco Catalyst switches. Instead, this section introduces some ideas that you will learn about in more detail in later chapters. The later chapters discuss the following QoS concepts for each Catalyst switching platform in more detail than they are introduced here:

- Classification
- Marking
- Policing
- Congestion Management
- Congestion Avoidance

Not all of Catalyst switching families support all the listed features, and those features are omitted from discussion on nonapplicable switch families. The following sections briefly discuss these features and their relation to Catalyst switches.

Catalyst QoS Classification

Classification determines how a switch or router marks, processes, and schedules frames. Currently shipping Catalyst switches utilize an internal DSCP value to correctly schedule and mark frames for egress transmission. Chapter 6 first introduces the concept of internal DSCP.

Moreover, Catalyst switches classify frames based on a variety of ingress frame parameters such as CoS, DSCP, IP precedence, trust, ingress interface, or IP address. Trust, based either on platform-specific default or user configuration, is an indication of whether the network administrator trusts the QoS markings of ingress frames on a per-interface basis. Generally, network administrators do not trust user ports because operating systems enable users to set the CoS value on egress packets. This situation may yield a negative impact on the network because users determine the priority of their traffic. Conversely, network designs typically trust infrastructure connections such as switch-to-switch, switch-to-router, or switch-to-IP Phone connections. The assumption in this case is that the other switches are already configured properly for trusting, classification, and marking.

Trusted interfaces do not alter the QoS marking of ingress frames. Untrusted interfaces alter the QoS markings to a configurable value CoS or DSCP. This value is typically zero (Best Effort) for untrusted interfaces. The command-line configuration and restrictions associated with trusted or untrusted interfaces are per platform. Later chapters discuss these configurations and restrictions, when the focus is on platform-specific properties. This trust concept is discussed in more detail in the section "Cisco Catalyst QoS Trust Concept" later in this chapter.

Catalyst QoS Marking

Marking is the act of a switch or router rewriting the CoS, DSCP, or IP precedence fields of frames based on classification. Marking modifies the intended ingress frame behavior as set by the originating device. Catalyst switches use interface configurations or policers to define marking parameters.

Catalyst QoS Policing

Catalyst switches define policers for applying bandwidth limits to ingress and egress traffic. In addition, Catalyst switches use policers to mark traffic. For example, it is possible to configure a policing policy such that all traffic of a certain type below 1 Mbps is marked with a DSCP value of af31 and all traffic above 1 Mbps is marked with a DSCP value of af32.

As with several with Catalyst QoS features, the configuration syntax and actual behavior for policing on Catalyst platforms is platform-specific and is discussed in later chapters. However, common to all platforms is the use of a token bucket concept for the policing bandwidth function.

Catalyst QoS Congestion Management

Catalyst switches use scheduling and transmit queues to achieve congestion management. All the currently shipping switches in the Catalyst product line support a form of scheduling that is more advanced than FIFO. The specific mechanisms differ on a platform basis.

With regard to output scheduling, the QoS marking determines the scheduling and output queue. In the case of the Catalyst 6500 Family of switches, for example, there are different queue types and drop thresholds per line module. Other platforms, such as the Catalyst 3500 Family and the Catalyst 4000 IOS Family of switches, use a single queue type for all line modules and product families.

Congestion Avoidance

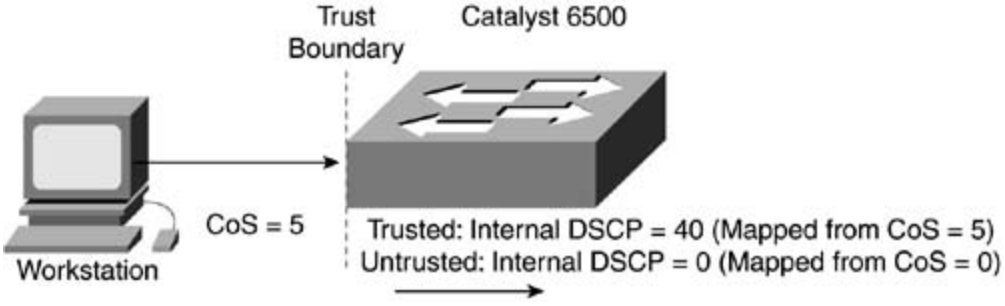
At time of publication, only the Catalyst 3550 Family and the Catalyst 6500 Family of switches support congestion avoidance. In brief, congestion avoidance attempts to prevent congestion by applying specific queuing parameters. The Catalyst 3550 Family and the Catalyst 6500 Family of switches utilize WRED and several other queuing configurations to support congestion avoidance. This book discusses the Catalyst 3500 Family and Catalyst 6500 Family of switches in Chapters 6 and 8, respectively.

Cisco Catalyst QoS Trust Concept

The trust concept is a classification configuration option supported on all Catalyst switches that support QoS classification. The trust state of a switch port or interface defines how ingress packets are classified, marked, and subsequently scheduled. For a Cisco Catalyst switch that bases QoS only on CoS values, a port that is configured as untrusted reclassifies any CoS values to zero or to a statically configured CoS value. The CoS values of packets arriving on an untrusted port are assumed not verifiable and deemed unnecessary by the system administrator of the switch. Depending on the platform, untrusted ports may be configured to reclassify or mark IP precedence, DSCP, or CoS values on any ingress frame based on an 802.1q tag or access list.

Figure 2-8 illustrates the QoS trust concept. A workstation attached to a Catalyst 6500 switch is sending 802.1q tagged frames to the Catalyst 6500 switch with a CoS value of 5. If the port is configured as untrusted, the switch sets an internal DSCP value associated with the frame to 0. The switch does not actually alter the CoS value of the frame until transmission. All untrusted ports set the internal DSCP to 0 by default. However, the overriding internal DSCP value is configurable on various platforms. If the switch port is configured for *Trust-CoS*, the CoS value is not altered on ingress. Figure 2-6 applies to trusting IP precedence and DSCP values as well.

Figure 2-6. Catalyst QoS Trust Concept



Switches that support classification of frames based on DSCP values derive an internal DSCP value for internal priority as the packet transits the switch. Several options exist to derive this internal DSCP value depending on platform. In brief, the general possible configurations for mapping an internal DSCP value are as follows:

- Trust-IPPrec— Internal DSCP is derived from the received IP precedence.
- Trust-DSCP— Internal DSCP is derived from the received DSCP.
- Trust-CoS— Internal DSCP is derived from the received CoS.
- Untrusted— Internal DSCP is derived from port configuration.

Additional configuration parameters are involved in determining an internal DSCP value, and caveats apply to this process. These caveats and configuration parameters are discussed on a per-platform basis throughout the book.

In summary, trusted ports are assumed to have ingress packets marked with IP precedence, DSCP, or CoS values that are valid. Untrusted ports are considered to have ingress frames marked with IP precedence, DSCP, or CoS values that are not deemed valid or desired by the system administrator of the switch.

The Cisco IP Phone

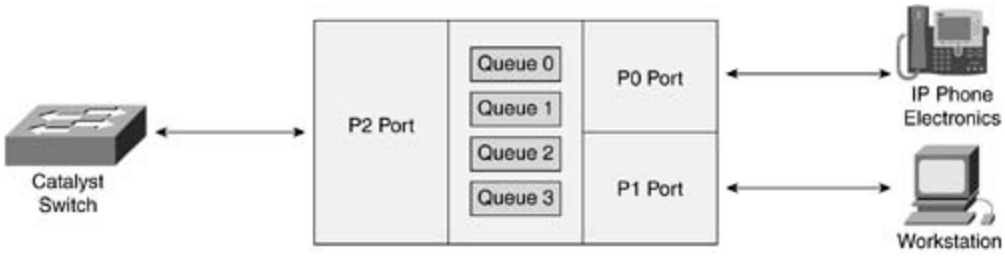
The Cisco IP Phone plays an important role in Cisco Catalyst QoS. A large majority of customers implement Cisco Catalyst QoS for the sole purpose of VoIP using Cisco IP Phones. Although a variety of IP Phone models exist, each phone uses a similar architecture. All current Cisco IP Phones include an internal three-port Layer 2 switch. As shown in Figure 2-7, the internal three-port switch enables customers to connect workstations through the IP Phone, which is in turn connected to a Catalyst switch using a single cable. Most customers actually use this daisy-chaining feature for both the phone and workstation to reduce the cable plant size and cost.

Figure 2-7. Cisco IP Phone Physical Network Example

Network administrators generally accept a Cisco IP Phone as a trusted device. Most QoS campus network designs suggest using the trust feature with Cisco IP Phones.

The internal switch ports of the Cisco IP Phone are referred to as P0, P1, and P2. P0 internally connects to the internal IP Phone appliance, P1 is an external 10/100-Mbps Fast Ethernet port that connects PCs and workstations, and P2 is an external 10/100-Mbps Fast Ethernet port that connects to the Catalyst switch. Figure 2-8 illustrates the integrated switch architecture in the Cisco IP Phone.

Figure 2-8. Cisco IP Phone Integrated Switch Architecture

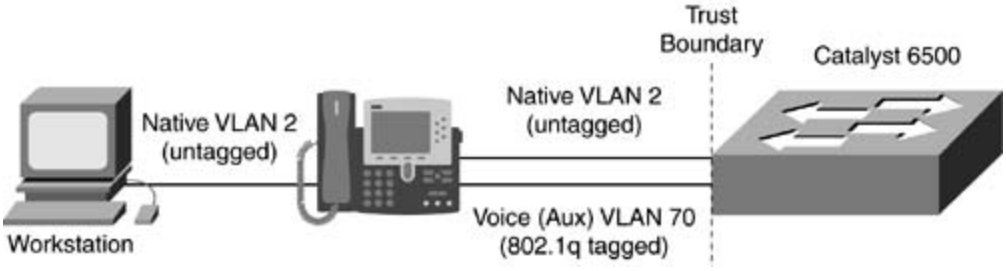


Each Cisco IP Phone port has four queues comprised of a single threshold statically configured at 100 percent, 4q1t. Queue 0 functions as a high-priority queue for traffic with a CoS value of 5. Spanning-tree*bridge protocol data units* (BPDUs) also use this queue. Voice traffic from the internal IP Phone appliance has a CoS value of 5 by default. All queues are serviced in a round-robin fashion. However, a timer maintains queue priority by limiting service to the low-priority queues when there is traffic in the high-priority queue. Subsequently, the Cisco IP Phone itself manages input and output scheduling for traffic traversing the integrated switch ports.

Voice VLANs and Extended Trust

Through the use of dot1q trunks, voice traffic from an IP Phone connected to an access port can reside on a separate VLAN and subnet. The workstation attached to the IP Phone might still reside on the access, or native, VLAN. This additional VLAN on an access port for voice traffic is referred to as a *voice VLAN* in Cisco IOS Software and *auxiliary VLAN* in CatOS. Subsequently, with the use of voice VLANs, all voice traffic is tagged to and from the Cisco IP Phone and Catalyst switch. The Catalyst switches use *Cisco Discovery Protocol* (CDP) to inform the IP Phone of the voice VLAN ID. By default, Cisco IP Phone voice traffic has a CoS value of 5. Figure 2-9 provides an example logical depiction of a voice VLAN. A common network design is to deploy both voice VLANs with trusting configurations for Cisco IP telephony applications (such as Cisco IP Phones).

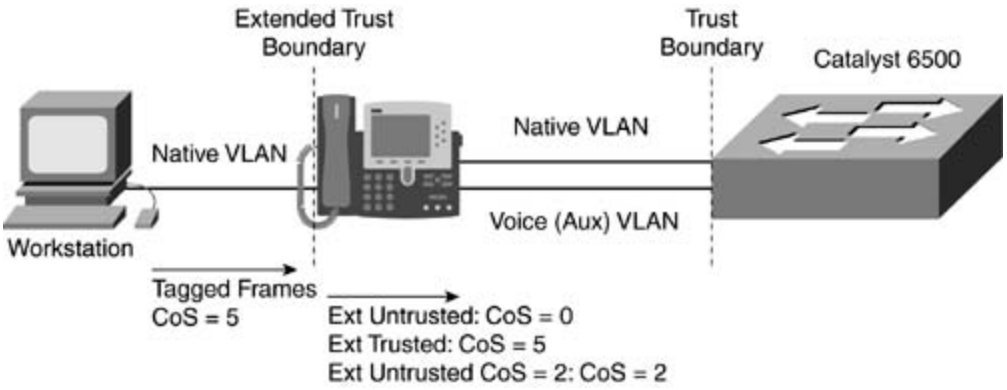
Figure 2-9. Example Logical Depiction of Voice VLAN



Another QoS option for IP Phones is extended trust. The switch can inform the IP Phone via CDP whether to trust ingress frames on its P1 port. The IP Phone may also be informed to overwrite the CoS value of the ingress frames on the P1 port with a specific CoS value. By default, the IP Phone does not trust frames arriving on the P1 port and rewrites the CoS value to 0 of any tagged frames. Untagged frames do not have CoS value.

Extended trust is a feature available to any device that can interpret the CDP fields describing the voice VLAN information. At the time of publication, Cisco IP Phones and other Cisco appliances are the only devices to use this feature.

Figure 2-10. Catalyst Extended Trust Concept


Extended Trust
Boundary
Trust
Boundary
Catalyst 6500
Native VLAN
Native VLAN
Voice (Aux) VLAN
Workstation
Tagged Frames
CoS = 5
Ext Untrusted: CoS = 0
Ext Trusted: CoS = 5
Ext Untrusted CoS = 2: CoS = 2

Summary

This chapter has given you a broad view of the QoS theories and mechanisms at Layer 2 and Layer 3, but hopefully left you hungry for more details. Those details are coming in each of the remaining chapters of this book. Now that you are familiar with the RFCs and overall concepts that drive the development of QoS mechanisms on Cisco devices, you will read about the very specific and often intricate details of these mechanisms on Cisco Catalyst platforms. You may find, as you read through some of the more detailed chapters, that you want to refer back to this chapter to see where a specific implementation detail fits into the end-to-end QoS strategy.

The next chapter includes a broad overview of QoS feature support on all Catalyst switches, and then focuses on QoS specifics on the Catalyst 2900XL, 3500XL, and 4000 CatOS Family of switches.

As you read the implementation specifics in the following chapters, keep an eye on the big picture of end-to-end QoS. From the Catalyst 3550 Family of switches to the Catalyst 4000 IOS Family of switches to the Catalyst 6500 Family of switches, think about how the configuration of each platform can be customized to fit the exact needs of your network. After all, you want to build a custom QoS policy that allows for the use of all the platforms in a single network, with a single end-to-end policy.

Chapter 3. Overview of QoS Support on Catalyst Platforms and Exploring QoS on the Catalyst 2900XL, 3500XL, and Catalyst 4000 CatOS Family of Switches

Previous chapters described the necessity for QoS in campus networks and the fundamentals behind QoS operation. This chapter explains the various platform QoS features available across the Cisco Catalyst product family. A group of concise tables in the beginning of this chapter provides a quick reference for QoS features available for each Catalyst platform. In addition, this chapter, along with subsequent chapters, begins the product tour of the access layer Catalyst switches with the fewest QoS features and continues with the high-end core Catalyst switches with industry-leading QoS features. Although the access layer switches support only a few QoS features, these switches provide an excellent foundation for exploring QoS fundamentals in the campus network.

Specifically, this chapter covers the following topics:

- Brief Per–Catalyst Platform QoS Features Table
- QoS Features Overview
- QoS Features on the Catalyst 2900XL and 3500XL Switches
- QoS Features on the Catalyst 4000 CatOS Switches

The Cisco Catalyst 2900XL, 3500XL, and 4000 Family of switches share Layer 2 QoS features needed on access layer switches. These features include the following:

- Classification
- Marking
- Congestion Management

This chapter covers these topics on the respective platforms with command references, examples, and case studies. Upon completion of this chapter, you will understand each Catalyst platform's supported QoS features and be able to configure the Catalyst 2900XL, 500XL, and 4000 CatOS Family of switches for packet classification, marking, and congestion management.

From a platform perspective, the Catalyst 4000 CatOS Family of switches must be distinguished from the Catalyst 4000 Cisco IOS Family of switches due to individual differences in QoS features and configuration. The supervisor engine model determines whether a Catalyst 4000 switch operates on CatOS or Cisco IOS. In addition, the Catalyst 4000 Layer 3 services module also has exclusive Layer 3 QoS features (discussed in Chapter 7, "Advanced QoS Features Available on the Catalyst 4000 IOS Family of Switches and the Catalyst G-L3 Family of Switches"). This chapter is only applicable to the Catalyst 4000 CatOS switches. Table 3-8 shows which Catalyst 4000 switches are applicable to this chapter.

Catalyst Feature Overview

Cisco Catalyst switches support a wide range of QoS features. Generally, the high-end platforms support more QoS features especially platforms that support Layer 3 IP routing. Tables 3-1 through3-5 provide a quick reference for QoS features for each platform. All platforms may have limitations and caveats per feature, and each QoS feature is discussed in the appropriate chapter of this book in additional detail.

Furthermore, QoS features are also dependent on whether the platform supports IP routing. The Catalyst 3550, Catalyst 4000 Cisco IOS Software family, Catalyst 5500 with *Route Switch Module* (RSM) or *Router Switch Feature Card* (RSFC), and the Catalyst 6000/6500 with Multilayer Switch Module (MSM) or *Multilayer Switch Feature Card* (MSFC) I/II support IP routing. Other platforms may support Layer 3 QoS features, such as classification based on *differentiated services codepoint* (DSCP) and marking of IP precedence; however, these platforms do not actually support routing of IP frames. As a result, network designs do not require platforms that support IP routing to classify, mark, police, or schedule traffic based on DSCP or IP precedence values. Therefore, network designers may choose lower-cost switches that do not support IP routing to enable Layer 3 QoS features.

The next sections provide quick reference tables for supported QoS features per platform. The tables only provide a glimpse into QoS feature support of each platform and do not indicate the benefits or restrictions of each feature. Refer to the appropriate chapters later in this book for thorough discussions of QoS feature support on each platform.

Specifically, the next sections highlight the following QoS features supported on each platform:

- Input Scheduling
- Policing
- Classification and Marking
- Output Scheduling

Table 3-1 indicates at a simplistic level, QoS feature support on a per-platform basis for most of the currently shipping Catalyst switches. The table only indicates at the fundamental level where a feature is supported and does not indicate the restrictions or caveats.

Table 3-1. QoS Feature Overview on Current Catalyst Switches

Product Family	Classification	Marking	Policing	Congestion Management	Congestion Avoidance
2950	Yes	Yes	Yes	Yes	No
3550	Yes	Yes	Yes	Yes	Yes
4000 IOS Family	Yes	Yes	Yes	Yes	No
6500 Family	Yes	Yes	Yes	Yes	Yes

Input Scheduling

Input scheduling is currently available only on the Catalyst 6000/6500. Input scheduling priorities and schedules packets out of ingress packet queues based on several QoS values including CoS and DSCP. However, most of Catalyst switches can deliver packets to the switching fabric at line rate or a specified rate. This specific rate defines the maximum throughput of the switch. If the input rate is not exceeded, input scheduling is not crucial in implementing QoS architecture. Furthermore, ingress policing is an option on many Catalyst switches that aids in preventing oversubscription of the switch fabric by limiting ingress traffic. Table 3-2 summarizes Catalyst platform support for input scheduling. The Comments column also denotes any switch capable of ingress policing.

Table 3-2. Catalyst Platform QoS Input Scheduling Support

Catalyst Switch	Input Scheduling	Ingress Policing	Comments
Catalyst 2900XL	No	No	Switching fabric is capable of 1.6 Gbps ingress.
Catalyst 2948G-L3/4912G-L3/4232-L3	No	Yes	
Catalyst 2950	No	Yes	
Catalyst 3500XL	No	No	Switching fabric is capable of 5.0 Gbps ingress.
Catalyst 3550	No	Yes	
Catalyst 4000 CatOS Family	No	No	Nonblocking line cards can deliver ingress traffic at line rate to switching fabric
Catalyst 4000 Cisco IOS Family (Supervisor III and IV)	No	Yes	Non-blocking linecards can deliver ingress traffic at line rate to switching fabric.
Catalyst 5500	No	No	
Catalyst 5500 w/NFFC[*] II	No	No	
Catalyst 6000/6500	Yes	Yes	Based on Layer 2 CoS[**]; option for ingress Priority Queue.

[*] NFFC = NetFlow Feature Card

[**] CoS = class of service

Classification and Marking

Classification and marking support and features vary per switch. Table 3-3 indicates which platforms support specific classification and marking features. All switches that support QoS also support classification based on CoS values. Current generation switches that support IP routing also support classification and marking using IP precedence or DSCP values in addition to classification and marking of CoS values.

Table 3-3. Catalyst Platform QoS Classification and Marking Support

Catalyst Switch	Classification Marking of Untagged Frames	Marking CoS on Tagged Frames	Marking DSCPon Tagged Frames	Classification Based onDSCPof Ingress Frames
Catalyst 2900XL	Yes	No	No	No
Catalyst 2948G-L3/4912G-L3/4232-L3	No	No	No	No, IP precedence only
Catalyst 2950	No, IP precedence only	Yes	Yes	Yes
Catalyst 3500XL	Yes	Yes, on specific models	No	No
Catalyst 3550	Yes	Yes	Yes	Yes
Catalyst 4000 CatOS Family	Yes	Yes	No	No
Catalyst 4000 Cisco IOS Family (Supervisor III and IV)	Yes	Yes	Yes	Yes
Catalyst 5500	Yes, requires NFFC II	Yes, requires NFFC II	Yes, requires NFFC II	Yes, requires NFFC II
Catalyst 6500	Yes	Yes	Yes	Yes

Policing

Table 3-4 indicates which Catalyst platforms support policing. Feature support and platform implementation of policing varies between each Catalyst switch. Three types of policing exist for Catalyst platforms:

- Individual policing
- Aggregate policing
- Microflow policing

Individual policing applies the bandwidth limit of a policer per interface. For example, an individual policer configured to constrain ingress traffic to 32 kbps limits each applicable interface to 32 kbps on ingress. An aggregate policer configured for the same bandwidth constraint limits the bandwidth collectively among all interfaces. Microflow policing is available on the Catalyst 6500, and it applies bandwidth limits to each *access-control entry* (ACE) of a defined policer. Chapter 8, "QoS Support on the Catalyst 6500," discusses ACEs and microflow policing in more detail.

Each platform has unique support, restrictions, and requirements surrounding policing. Refer to each product chapter for specifics.

Table 3-4. Catalyst Platform QoS Policing Support

Cisco Catalyst Platform	Ingress Policing	Egress Policing	Individual Policing	Aggregate Policing	Microflow Policing
Catalyst 2900XL	No	No	No	No	No
Catalyst 2948G-L3/4912G-L3/4232-L3	Yes, per-port rate-limiting	Yes, per port rate-limiting and traffic shaping	No	No	No
Catalyst 2950	Yes	No	Yes	No	No
Catalyst 3500XL	No	No	No	No	No
Catalyst 3550	Yes	Yes	Yes	Yes	No
Catalyst 4000 CatOS Family	No	No	No	No	No
Catalyst 4000 Cisco IOS Family (Supervisor III and IV)	Yes	Yes	Yes	Yes	No
Catalyst 5500 w/NFFC II	No	No	No	No	No
Catalyst 6500	Yes	No	No	Yes	Yes

Congestion Management

Congestion management is supported on all Catalyst switches that support QoS features. Congestion avoidance and management is achieved via the use of output scheduling using the tail-drop and *Weighted Random Early Detection* (WRED) queuing mechanisms. Chapter 2, "End-to-End QoS: Quality of Service at Layer 3 and Layer 2," explains the difference between congestion management and congestion avoidance, and later chapters explain the tail-drop and WRED queuing mechanisms in the congestion avoidance section of each chapter where applicable. Moreover, only the Catalyst 3550, Catalyst 4000 IOS Family of switches, and the Catalyst 6500 support congestion avoidance.

The nomenclature for output scheduling queues is a follows:

*X*p*Y*q*Z*t

- *X* indicates the number of strict-priority queues.
- *Y* indicates the number of queues other than strict-priority queues.
- *Z* indicates the configurable thresholds per queue.

For example, 1p3q2t indicates that a switch has an egress output queue with one strict-priority queue and three normal-priority queues each with two configurable thresholds per queue.

Table 3-5 indicates the available output queues per platform.

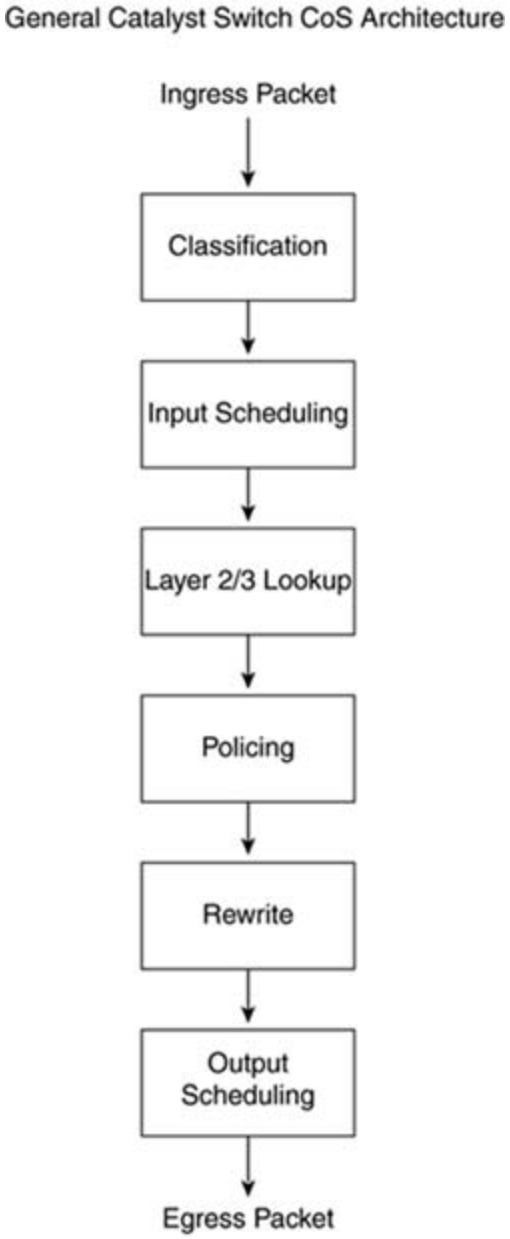
Table 3-5. Catalyst Platform Congestion Management Support

Cisco Catalyst Platform	Output Scheduling	Scheduling Queues
Catalyst 2900XL	Yes	Global 2q1t
Catalyst 2948G-L3/4912G-L3/4232-L3	Yes	4q
Catalyst 2950	Yes	4q
Catalyst 3500XL	Yes	Global 2q1t
Catalyst 3550	Yes	1p3q2t, 4q4t
Catalyst 4000 CatOS Family	Yes	2q1t
Catalyst 4000 Cisco IOS Family (Supervisor III and IV)	Yes	1p3q1t, 4q1t
Catalyst 5500 w/NFFC II	Yes	1q4t
Catalyst 6500	Yes	Ingress: 1q4t, 1p1q4t, 1p1q, 1p2q1t Egress: 2q2t, 1p2q2t, 1p3q1t, 1p2q1t, 1p1q8t, and 1p1q0t

Material Presentation for Catalyst Switching Platforms

Figure 3-1 shows the general QoS packet-flow architecture for Cisco Catalyst switches. The architecture presents only the standard model for QoS features on each Catalyst platform. However, support for each feature of the architecture is platform dependent and varies significantly for each Catalyst switch.

Figure 3-1. General Catalyst QoS Packet-Flow Architecture



For each Catalyst platform, this book discusses QoS features using the following flowchart:

- QoS Architecture Overview
- Input Scheduling
- Classification and Marking
- Policing
- Congestion Management and Avoidance
- Sample Configurations and Case Studies
- Summary

As indicated in the Tables 3-1 through 3-5, not every Cisco Catalyst platform supports all the QoS components and features. For those platforms, the QoS component is omitted or discussed as an unsupported feature. Chapter 10, "End-to-End QoS Case Studies," concludes with comprehensive case studies using several Cisco Catalyst switches and QoS features.

QoS Support on the Catalyst 2900XL and 3500XL

Specific models of the Catalyst 2900XL and 3500XL support QoS classification and congestion management. Ingress packet CoS values and configured port priorities exclusively determine classification of ingress frames for placement into either a low-priority or high-priority global trans queue. The two global transmit queues with priority scheduling create the congestion management mechanism. The following sections discuss these QoS features with detailed overviews, configuratio guidelines, and examples.

Catalyst 2900XL Product Family Delineation

QoS support on the Catalyst 2900XL and 3500XL platforms is software and model dependent. For t 2900XL, the original-edition models do not support QoS features, including the uplink modules for Catalyst 2916M. All standard- and enterprise-edition models do support QoS features. Table 3-6 indicates which 2900XL models support QoS features. All models of the 3500XL support QoS featur In addition, the Catalyst 3524XL-PWR-XL and the 3548XL support CoS reclassification. This QoS fe is not available on other 3500XL platforms.

Table 3-6. QoS Support by Model of 2900XL

Catalyst 2900XL/3500XL Model	Description	QoS Support
WS-C2908-XL	8-port 10/100BASE-TX switch	No
WS-C2912-XL-A/EN	12-port 10/100BASE-TX switch	Yes
WS-C2912MF-XL	12-port 100BASE-FX switch	Yes
WS-C2916M-XL	16-port 10/100BASE-TX switch + 2 uplink slots	No
WS-C2924-XL	24-port 10/100BASE-TX switch	No
WS-C2924C-XL	22-port 10/100BASE-TX + 2-port 100BASE-FX switch	No
WS-C2924-XL-A/EN	24-port 10/100BASE-TX switch	Yes
WS-C2924C-XL-A/EN	22-port 10/100BASE-TX switch + 2-port 100BASE-FX switch	Yes
WS-C2924M-XL-A/EN	24-port 10/100BASE-TX switch + 2 uplink slots	Yes
WS-C2924M-XL-EN-DC	24-port 10/100BASE-TX switch + 2 uplink slots (DC power)	Yes

Catalyst 2900XL and 3500XL QoS Architectural Overview

The Catalyst 2900XL and 3500XL switches are limited to QoS features that suit access layer switche These features include classification, marking, and congestion management via the use of output scheduling. Because of these features, the 2900XL and 3500XL fit well into an end-to-end QoS desi

with core switches. Figure 3-2 shows a sample network deploying access layer QoS features with comprehensive QoS features in the core.

Figure 3-2. Network Topology Using Catalyst 3524XLs

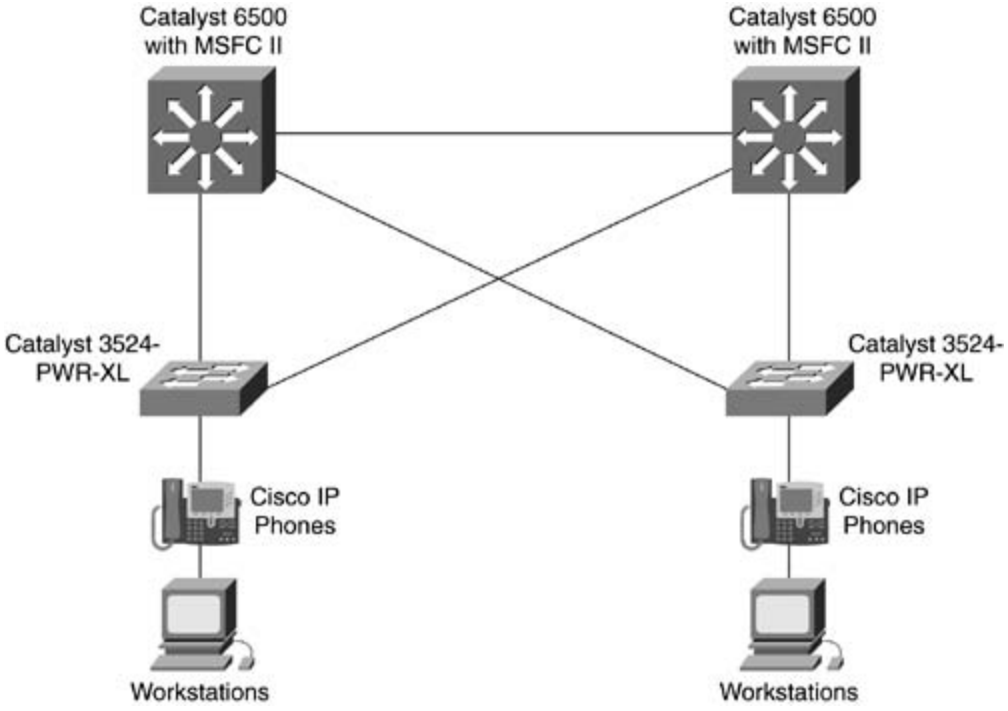
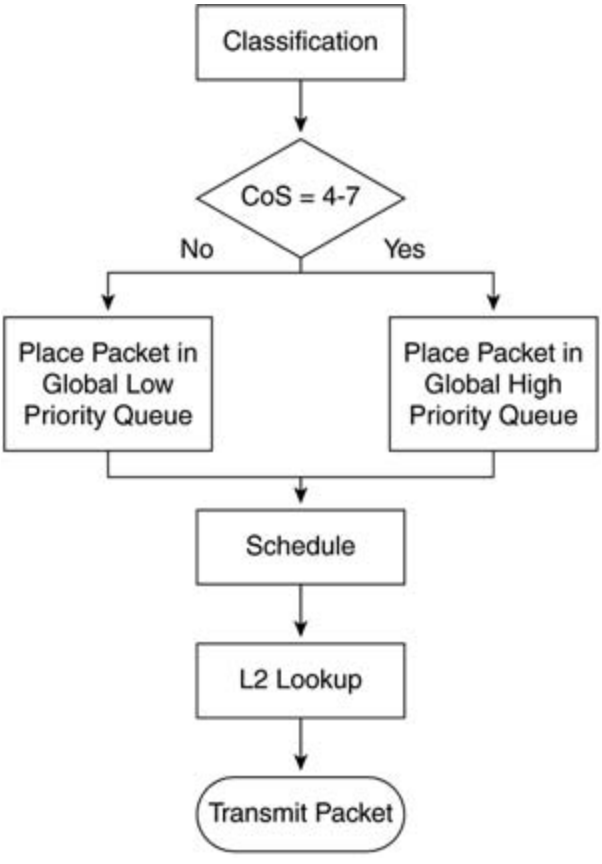


Figure 3-3 shows the basic QoS architecture model for the Catalyst 2900XL and 3500XL discussed the following sections.

Figure 3-3. Basic QoS Architecture for the Catalyst 2900XLs and 3500X Switches



Software Requirements

For QoS feature support, the 2900XL and 3500XL require Cisco IOS Software Release 12.0(5)XP or higher. The 3524-PWR-XL and the 3548XL require 12.0(5)XU or higher for the reclassification of cl of CoS values in frames.

Input Scheduling

The Catalyst 2900XL and 3500XL do not perform input scheduling as ingress packets are immediat copied to a global, shared memory buffer. As long as the packet-forwarding rate of the switch is nc exceeded, input congestion is not critical to implementing QoS. The packet-forwarding rates of the Catalyst 2900XL and the Catalyst 3500XL are 1.6 Gbps and 5.0 Gbps, respectively.

Classification/Reclassification

The Catalyst 2900XL and 3500XL switches both support classification of untagged frames. Two mo of the Catalyst 3500XL switch, the Catalyst 3524-PWR-XL and the Catalyst 3548XL switches, suppc marking of ingress tagged frames. Classification and marking is configurable only on a per-port ba and each port may be configured with a unique CoS value to be classified.

To configure a Catalyst 2900XL and 3500XL for classification or marking of frames, use the followir interface command:

```
switchport priority {default default-priority-id | extend {cos value | none | trust}
override}
```

- The *default-priority-id* parameter is the CoS value to be assigned to untagged ingress frames.
- The **extend** option is to configure the 802.1p trust configuration of the connected appliance or P1 port of the IP Phone. For example, a Cisco IP Phone can be configured to trust or reclassify frames received on its P1 port.
- The **override** option is used to mark tagged frames with the *default-priority-id*. Only the Catal 3524-PWR-XL and the Catalyst 3548XL switches support this marking feature.

Example 3-1 shows a Catalyst 3548XL switch port configured to classify untagged frames with a Co value of 2.

Example 3-1. Catalyst 3548XL Switch Port Configured to Classify Untagg Frames

```
Switch#show running-config
Building configuration...

Current configuration:
(text deleted)
!
interface FastEthernet0/48
 switchport priority default 2
 spanning-tree portfast
 (text deleted)
end
```

Example 3-2 shows a Catalyst 3524-PWR-XL switch port configured for a voice VLAN. In Cisco IOS Software Release 12.0.x, voice VLAN ports must be configured as trunk ports.

Example 3-2. Catalyst 3548XL Switch Port for Voice VLANs

```
Switch#show running-config
Building configuration...

Current configuration:
(text deleted)
!
interface FastEthernet0/24
 switchport trunk encapsulation dot1q
 switchport trunk native vlan 2
 switchport mode trunk
 switchport voice vlan 70
 spanning-tree portfast
(text deleted)
end
```

> **NOTE**
>
> The Catalyst 2900XL and 3500XL software configuration for voice VLANs differs from the Catalyst switches that run Cisco IOS Software Release 12.1, such as the Catalyst 4000 Supervisor III and IV. Cisco IOS Software Release 12.1 does not require voice VLAN ports to be configured as trunks.

Congestion Management

The 2900XL and 3500XL switches use a shared memory buffer system because each individual port does not have its own output queue. This shared memory buffer is divided into two global transmit queues. Each ingress packet is placed into one of two global transmit queues based on CoS value f tagged frames and CoS classification for untagged frames. One of the transmit queues is designate packets with a CoS value of 0 to 3, and the other transmit queue is reserved for packets with a CoS value of 4 to 7. The queues use a 100-percent threshold value. These queues are not configurable t different CoS values or thresholds. This queue scheme creates a logical high-priority and low-priori queuing mechanism. Priority scheduling is applied such that the high-priority queue is consistently serviced before the low-priority queue. The use of two global transmit queues based on CoS value default behavior and cannot be altered. As a result, no global configuration is required to enable Q output scheduling.

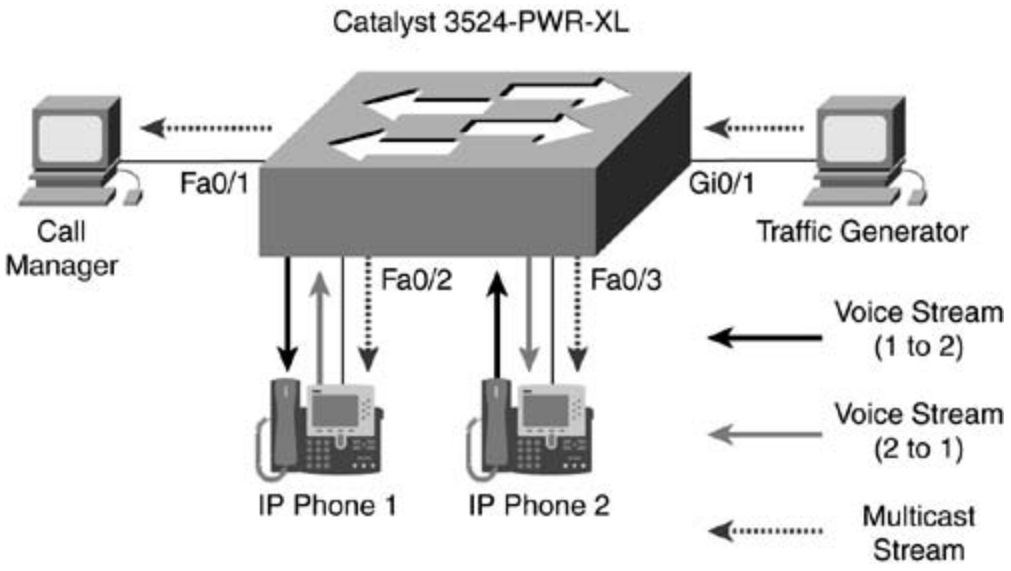
NOTE

Untagged packets that are classified with a CoS value transmitted on trunk ports are appropriately tagged with an 802.1q header with the respective CoS. For packets transmitted on nontrunk ports, the untagged classification only determines which queue the frame is placed in for egress transmission.

Case Study: Classification and Output Scheduling on Cisco Catalyst 3500 Switches

To demonstrate classification and output scheduling on the Catalyst 3500XL series, a Catalyst 3524 PWR-XL was set up with two Cisco 7960 IP Phones, a Call Manager, and a traffic generator connect to three Fast Ethernet ports and a Gigabit Ethernet port, respectively. Figure 3-4 shows this topolo Two trials were conducted taking voice quality statistical measurements from each IP Phone based 1-minute, G7.11 voice call between IP Phone 1 and 2. To create traffic congestion, the traffic gener attached to Gigabit Ethernet port was sending multicast at line rate with a CoS value of 0. The multicast traffic was flooded to all ports, including the Fast Ethernet IP Phones, causing output congestion.

Figure 3-4. Catalyst 3500XL Case Study Network Diagram



The Catalyst 3524XL switch was running software version 12.0(5)WC5 for the trial. The configurati only included voice VLANs on the Cisco IP Phone ports. The remaining port configuration of the swi was default. Example 3-3 shows the relevant configuration.

Example 3-3. Catalyst 3548XL Switch Port Configuration for Case Study

```
Switch#show running-config
Building configuration...

Current configuration:
(text deleted)

interface FastEthernet0/1
 switchport access vlan 70
!
interface FastEthernet0/2
 switchport trunk encapsulation dot1q
 switchport trunk native vlan 2
 switchport mode trunk
```

```
 switchport voice vlan 70
 spanning-tree portfast
!
interface FastEthernet0/3
 switchport trunk encapsulation dot1q
 switchport trunk native vlan 2
 switchport mode trunk
 switchport voice vlan 70
 spanning-tree portfast
!
(text deleted)


interface GigabitEthernet0/1
 switchport access vlan 70
(text deleted)
end
```

The variant in the two trials was the CoS value placed on the telephony frames between the IP Pho With a CoS value of 0, the telephony stream was treated with a low priority (the same priority as t multicast traffic). With a CoS value of 5, the telephony stream was treated with a high priority. Tab 7 summarizes the number of frames transmitted and lost as well as jitter from each trial.

Table 3-7. QoS Trial Results on Catalyst 3524-PWR-XL

Trial	Total Frames Transmitted (Phone 1/2)	No. of Receive Lost Frames (Phone 1/2)	Maximum Recorded Jitter (Phone1/2)
CoS = 0 on voice stream	3100/3110	1551/1549	51/49 ms
CoS = 5 on voice stream	3104/3106	0/0	51/49 ms

As indicated in Table 3-7, the Catalyst 3524XL did not drop a single frame due to output congestio the IP Phone ports for packets with a CoS value of 5. Similar results are achievable with multiple C IP Phones in a campus network using the Catalyst 2900XL and 3500XL. The jitter did not vary betw the trials because all Catalyst switches drop frames under congestion and only buffer a few frames maximum recorded jitter is around 50 ms, which is above the recommended 30 ms for *Voice over* (VoIP). Only the first few frames of the IP flow recorded jitter near 50 ms.

Summary

The Catalyst 2900XL and 3500XL suit basic QoS needs for an access layer switch. If additional feat such as policing and classification based on DSCP are required, network designers should consider Catalyst 2950 and 3550 switches for use as an access layer switch. You can summarize QoS featur support on the Catalyst 2900XL and 3500XL switches as follows:

- No support for input scheduling.
- Classification based on CoS only; no support for classification based on IP precedence or DSC
- Two global queues for high-priority and low-priority traffic.
- No configurable CoS mapping to queues or queue threshold.
- Ports are trusted by default.
- Untagged frames are mapped to high-priority or low-priority queues based on configured classification CoS value.
- The Catalyst 3548XL and 3524-PWR-XL support reclassification of tagged frames.

QoS Support on the Catalyst 4000 CatOS Family of Switches

Catalyst 4000 CatOS switches provide for QoS classification and congestion management solely based on CoS values. The Catalyst 4000 IOS switches, discussed in Chapter 7, support a wider range of QoS. For the Catalyst 4000 CatOS switches, a high- and low-priority transmit port queue with round-robin scheduling accomplish congestion management. The Catalyst 4000 CatOS switches do not support policing or input scheduling. The following sections discuss the Catalyst 4000 CatOS QoS features with detailed overviews, configuration guidelines, and examples.

Catalyst 4000 Product Family Delineation

This section covers the Catalyst 4000 CatOS Family of switches. As discussed in the introduction to this chapter, the Catalyst 4000 Cisco IOS switches, the Catalyst 4000 CatOS switches, and the Layer 3 services module each have unique QoS feature support. The Catalyst 4000 Cisco IOS switches and the Layer 3 services module are covered in Chapter 7.Table 3-8 summarizes the Catalyst 4000 switches into the CatOS or IOS category. This chapter applies to the Catalyst 4000 switches that run CatOS Software.

Table 3-8. Catalyst 4000 CatOS Versus Cisco IOS Software Platform Support

Catalyst 4000 Model	Family	Description	Software
Catalyst 2948G	Catalyst 4000	48-port 10/100BASE-TX switch ports + 2 1000BASE-X GBIC[*] switch ports	CatOS
Catalyst 2980G	Catalyst 4000	80-port 10/100BASE-TX switch ports + 2 1000BASE-X GBIC switch ports	CatOS
Catalyst 2980G-A	Catalyst 4000	80-port 10/100BASE-TX switch ports + 2 1000BASE-X GBIC switch ports	CatOS
Catalyst 2948G-L3	Catalyst G-L3	48-port 10/100BASE-TX + 2 1000BASE-X GBIC Layer 3 switch	IOS
Catalyst 4003 + WS-X4012 Supervisor I Engine	Catalyst 4000	3-slot modular chassis + 2 1000BASE-X GBIC ports on Layer 2 supervisor	CatOS
Catalyst 4006 + WS-X4013 Supervisor II Engine	Catalyst 4000	6-slot modular chassis + 2 1000BASE-X GBIC ports on Layer 2 supervisor	CatOS
Catalyst 4006 + WS-X4014 Supervisor III Engine	Catalyst 4000	6-slot modular chassis + 2 1000BASE-X GBIC ports on Layer 2/3 supervisor	IOS

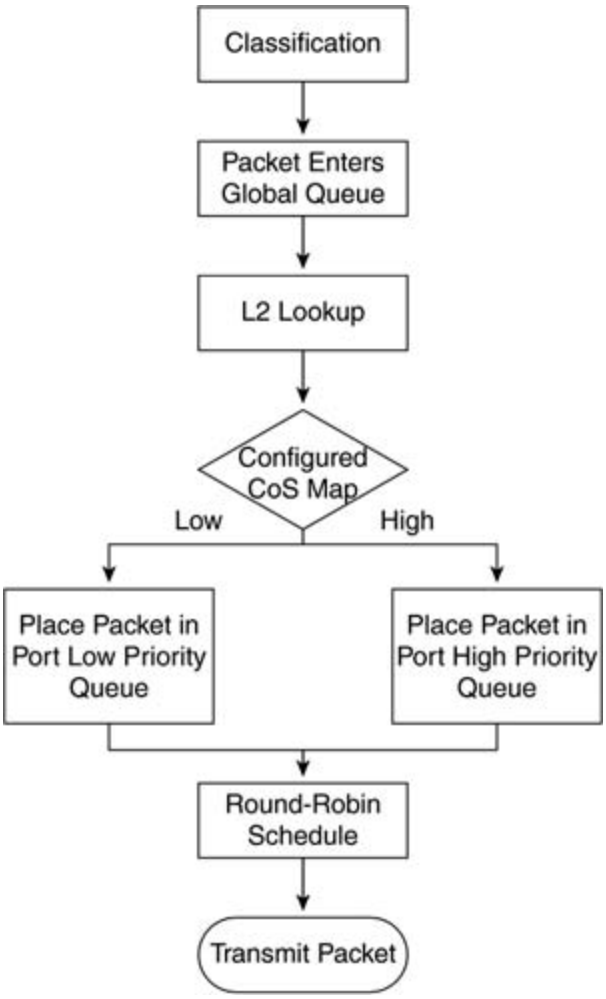
Catalyst 4006 + WS-X4515 Supervisor IV Engine	Catalyst 4000	6-slot modular chassis + 2 1000BASE-X GBIC ports on Layer 2/3 supervisor	IOS
Catalyst WS-X4232-L3 Layer 3 Services Module	Catalyst G-L3	Layer 3 router module for Catalyst 4003 and 4006 chassis with Supervisor I or II Engine	IOS
Catalyst 4503 + WS-X4013 Supervisor II Engine	Catalyst 4000	3-slot modular chassis + 2 1000BASE-X GBIC ports on Layer 2 supervisor	CatOS
Catalyst 4503 + WS-X4014 Supervisor III Engine	Catalyst 4000	3-slot modular chassis + 2 1000BASE-X GBIC ports on Layer 2/3 supervisor	IOS
Catalyst 4503 + WS-X4515 Supervisor IV Engine	Catalyst 4000	3-slot modular chassis + 2 1000BASE-X GBIC ports on Layer 2/3 supervisor	IOS
Catalyst 4506 + WS-X4013 Supervisor III Engine	Catalyst 4000	6-slot modular chassis + 2 1000BASE-X GBIC ports on Layer 2 supervisor	CatOS
Catalyst 4506 + WS-X4014 Supervisor III Engine	Catalyst 4000	6-slot modular chassis + 2 1000BASE-X GBIC ports on Layer 2/3 supervisor	IOS
Catalyst 4506 + WS-X4515 Supervisor IV Engine	Catalyst 4000	6-slot modular chassis + 2 1000BASE-X GBIC ports on Layer 2/3 supervisor	IOS
Catalyst 4507R + WS-X4515 Supervisor IV Engine	Catalyst 4000	7-slot modular chassis + 2 1000BASE-X GBIC ports on Layer 2/3 supervisor	IOS
Catalyst 4840G	Catalyst G-L3	40-port 10/100BASE-TX + 1000BASE-X GBIC Layer 3 server load-balancing switch	IOS
Catalyst 4908G-L3	Catalyst 4000	8 1000BASE-X GBIC Layer 3 switch	CatOS
Catalyst 4912G-L3	Catalyst G-L3	8 1000BASE-X GBIC switch ports	IOS

[*] GBIC Gigabit Interface Converter

Catalyst 4000 CatOS Family of Switches QoS Architectural Overview

The Catalyst 4000 CatOS switches support only QoS classification, marking, and congestion management. Classification and marking is based on the CoS value of 802.1q frames and port trust. Using two transmit queues for output scheduling achieves congestion management of egress traffic. Input scheduling is limited to *first-in, first-out* (FIFO) ingress queuing only. Figure 3-5 shows the basic QoS model for the Catalyst CatOS switches.

Figure 3-5. Basic QoS Architecture for the Catalyst Cat4000 CatOS Switches



Software Requirements

The Catalyst 4000 CatOS switches require CatOS Software version 5.2(1) or higher for QoS feature support.

Enabling QoS Features on Catalyst 4000 CatOS Switches

QoS must be globally enabled on CatOS switches before classification, marking, and output scheduling configurations are applied. To enable QoS on the Catalyst 4000 CatOS switches, enter the following command:

```
set qos {enable | disable}
```

Example 3-4 shows a user enabling QoS on a CatOS switch.

Example 3-4. Enabling QoS Features on a Catalyst 4000 CatOS Switch

```
Console> (enable) set qos enable

QoS is enabled.

Console> (enable)
```

Input Scheduling

Similar to other access layer switches, the Catalyst 4000 CatOS switches performs only FIFO Queuing of ingress packets. For line-module ports that are nonblocking, FIFO Queuing does not pose an issue because nonblocking line-module ports can deliver traffic to the switching fabric at line rate. Ports that are oversubscribed to the switching fabric are also referred to as *blocking ports*. Oversubscribed ports share bandwidth and data transmit contention in groups of two to eight ports depending on line module. Campus network design must consider oversubscribed ports very carefully on the Catalyst 4000 because of the lack of input scheduling. Furthermore, when using the nonblocking modules, consider aligning the front panels to minimize oversubscription. For example, avoid placing workstations utilizing real-time voice and video applications on the same group of ports that share oversubscribed bandwidth with high-traffic servers and network appliances. The product release notes contain detailed information on which ports share bandwidth. Moreover, all line-module ports support 802.1x flow control for constraining host traffic. 802.1x flow control is useful in limiting traffic for hosts connected to oversubscribed ports.

Table 3-9 lists the available line modules at the time of publication and denotes whether the ports are blocking or nonblocking. Several line modules are both nonblocking and blocking depending on the front-panel port. The table also describes how the ports are subscribed to the switching fabric.

Table 3-9. Catalyst 4000 Line Modules Architecture

Module	Ethernet Ports (Media Type)	Architecture to Switch Fabric
WS-U4504-FX-MT	4 100BASE-FX (MTRJ)	Nonblocking.
WS-X4012	2 1000BASE-X (GBIC)	Nonblocking.
WS-X4013	2 1000BASE-X (GBIC)	Nonblocking.
WS-X4014	2 1000BASE-X (GBIC)	Nonblocking.
WS-X4515	2 1000BASE-X (GBIC)	Nonblocking.
WS-X4124-FX-MT	24 100BASE-FX (MTRJ)	Nonblocking.
WS-X4148-FX-MT	48 100BASE-FX (MTRJ)	Nonblocking.
WS-X4148-RJ21	48-port 10/100BASE-TX (RJ21)	Nonblocking.
WS-X4148-RJ45	48-port 10/100BASE-TX (RJ45)	Nonblocking.
WS-X4148-RJ45V	48-port 10/100BASE-TX with Inline Power (RJ45)	Nonblocking.
WS-X4306-GB	6 1000BASE-X (GBIC)	Nonblocking.
WS-X4232-GB-RJ	32-port 10/100BASE-TX (RJ45) + 2 1000BASE-X (GBIC)	Nonblocking.
WS-X4232-L3	32, L2 10/100BASE-TX L2 (RJ45) + 2 L3 1000BASE-X (GBIC)	32 10/100BASE-TX ports are nonblocking.
WS-X4412-2GB-T	12-port 1000BASE-T (RJ45) + 2 1000BASE-X (GBIC)	The 2 1000BASE-X ports are nonblocking. The 1000BASE-T ports are group 3 front-panel ports to a 1-gigabit switch fabric connection.
WS-X4418-2GB	18 1000BASE-X (GBIC)	Front-panel ports 1 and 2 are nonblocking. Ports 3 through 18 are grouped 4 front-panel 1000BASE-X ports to a 1-gigabit switch fabric connection.
WS-X4424-GB-RJ45	24-port 10/100/1000BASE-T (RJ45)	Each consecutive group of 4 ports is connected to a 1-gigabit switch fabric connection.
WS-X4448-GB-LX	48-port 1000BASE-LX (SFP)	Each consecutive group of 8 ports is connected to a 1-gigabit switch fabric connection.
WS-X4448-GB-RJ45	48-port 10/100/1000BASE-T (RJ45)	Each consecutive group of 8 ports is connected to a 1-gigabit switch fabric connection.

Classification, Marking, and Trusting

The Catalyst 4000 CatOS switches are unable to differentiate between trusted and untrusted ports. As a result, the Catalyst 4000 CatOS switches consider all ports trusted, and the switch does not alter the CoS value for any Ethernet 802.1q tagged frames. System administrators need to be aware of servers, network appliances, or workstations that may be inappropriately marking CoS values in transmitted 802.1q tagged frames because the incorrectly marked frames could effect high-priority traffic such as voice or video.

Classifying Untagged Frames

The Catalyst 4000 CatOS switch may mark untagged frames with a default CoS value. The default CoS value is a global parameter applied to all ports for untagged frames received by the switch. This default CoS value marking technique cannot be applied to selective ports or selective frames. Marking is strictly a global parameter for untagged frames. To configure the default CoS value for untagged frames, enter the following command:

set qos defaultcos*default-cos-value*

default-cos-value indicates the CoS value to be marked on untagged frames.

Example 3-5 shows a user configuring a global default CoS value.

Example 3-5. Defining Default CoS Value on Catalyst 4000 CatOS Switch

```
Console> (enable) set qos defaultcos 5

qos defaultcos set to 5
```

> **NOTE**
>
> Extended trust configuration is not supported on the Catalyst 4000 CatOS switches.
>
> The Catalyst 4000 CatOS switches support only 802.1q trunking; *Inter-Switch Link* (ISL) trunking is not supported. The Catalyst 4000 Supervisor III and IV Engine both support ISL on existing linecards with a few exceptions.

Congestion Management

Congestion management is handled through the use of output scheduling. The Catalyst 4000 CatOS Software manages output scheduling by the use of a per-port, two-queues, one threshold (2q1t) system. Packets are mapped to a logical high- or low- priority output queue depending on the switch QoS configuration and CoS value in the frame. There is only one threshold setting, 100 percent; therefore, the only threshold configuration is to tail drop packets when a queue is full. Packets are removed from the queues round-robin with each queue getting serviced 1:1. Because packet flows with higher CoS values of less bandwidth are generally mapped to one specific queue, those packets are less likely to be dropped due to output congestion with the lower-priority, high-bandwidth packet flows.

To configure the CoS values to map to specific queues and verify the configuration, enter the following commands:

set qos map*port_type q# threshold#***cos***cos_list*

show qos info [**runtime** | **config**]

For the Catalyst 4000 CatOS switches, the *port_type* is always 2q1t with a *threshold#* of 1. *q#* identifies the queue to map the CoS value to, and the *cos_list* identifies what queue frames of specific CoS values are mapped. The *cos_list* must be configured in pairs: 0-1, 2-3, 4-5, and 6-7. Because QoS configurations are saved to the *nonvolatile random-access memory* (NVRAM) configuration at run time, the `runtime` and `config` options have no significance and both display the current and saved configuration. Not mapping CoS values after enabling QoS may result in unexpected performance because all CoS values map to the same transmit queue by default when QoS is enabled. Example 3-6 shows a user configuring and verifying the CoS mapping.

Example 3-6. Configuring Catalyst 4000 QoS CoS Mapping

```
Console> (enable) set qos map 2q1t 2 1 cos 4-7

Qos tx priority queue and threshold mapped to cos successfully.


Console> (enable) show qos info runtime
```

```
Run time setting of QoS:

QoS is enabled

All ports have 2 transmit queues with 1 drop thresholds (2q1t).

Default CoS = 0

Queue and Threshold Mapping:

Queue Threshold CoS

----- --------- ---------------

1     1         0 1 2 3

2     1         4 5 6 7
```

The Catalyst 4000 CatOS switch records the number of frames tail dropped as a result of the transmit port queue being full. The counters record the tail-drop frames as txQueueNotAvailable in the show counters*mod/port*. In addition, both the *out-lost* counter from the show mac [*mod*[/*port*]] command and the *Xmit-Err* counter from the show port [*mod*[/*port*]] command include the txQueueNotAvailable counter. Note that the *out-lost* and *Xmit-Err* are not inclusively counters for txQueueNotAvailable and increment for other packet counters as well. Example 3-7 shows some extrapolated output from the show counters [*mod*[/*port*]],show port [*mod*[/*port*]], and show mac [*mod*[/*port*]] commands from the QoS case study later in the chapter.

Example 3-7. show counters, show port, and show mac Command Output Excerpts

```
Console> (enable) show counters 5/1

(text deleted)

23 txQueueNotAvailable    = 19422994

(text deleted)
```

```
Console> (enable) show mac 5/1

(text deleted)

MAC      Dely-Exced MTU-Exced  In-Discard Lrn-Discrd In-Lost    Out-Lost

-------- ---------- ---------- ---------- ---------- ---------- ----------

 5/1             0          0          0          0          0    19422994
```

```
(text deleted)
```

```
Console> (enable) show port 5/1
(text deleted)
Port Align-Err    FCS-Err     Xmit-Err    Rcv-Err     UnderSize
----- ---------- ---------- ---------- ---------- ---------
 5/1           -          0    19422994          0         0
(text deleted)
```

Auxiliary VLANs

For VoIP appliances, such as the Cisco IP Phone, the 2q1t system works well. IP Phones should be configured in conjunction with auxiliary VLANs. Through the use of *Cisco Discovery Protocol* (CDP) packets, the IP Phone is informed of the auxiliary VLAN ID to use in sending tagged frames.

To configure a port for an auxiliary VLAN for tagged traffic, use the following command:

set port auxiliaryvlan*mod* [*/ports*] {**vlan** | **untagged** | **none**}

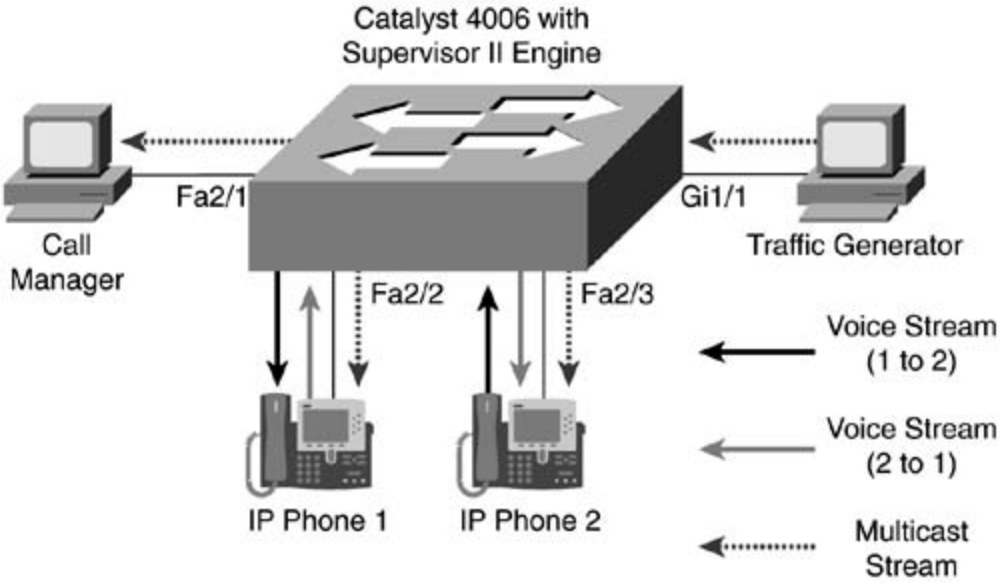
Thevlan option specifies the VLAN ID of the auxiliary VLAN. The untagged option tells the port to use untagged frames for the auxiliary VLAN, and none disables the auxiliary VLAN configuration on the port.

A LAN IP Phone conversation based on *pulse code modulation* (PCM) (G.711) compression uses only 83 kbps, far below the output rate of an Ethernet port operating at 10 Mbps. Cisco IP Phones connect at 100 Mbps full-duplex by default. Mapping only VoIP frames exclusively to a single queue based on CoS value allows voice traffic to flow egress from the output queue without packet loss even under output port loads above line rate.

Case Study: Output Scheduling on the Catalyst 4000 Series Switches

To illustrate the output scheduling behavior on the Catalyst 4000 Family, a Catalyst 4006 with a Supervisor II Engine running CatOS Software version 6.3.7 is connected to two Cisco 7960 IP Phones, a Cisco Call Manager server, and a traffic generator connected to three Fast Ethernet ports and a Gigabit Ethernet port as shown in Figure 3-6.

Figure 3-6. Catalyst 4000 Case Study Network Diagram



Two trials were conducted taking voice quality statistical measurements from each IP Phone based on a 1-minute, G7.11 voice call between IP Phone 1 and 2. To create traffic congestion, the traffic generator attached to the Gigabit Ethernet port was sending multicast at line rate with a CoS value of 0. The multicast traffic was flooded to all ports, including the Fast Ethernet ports connected to the IP Phones. This flooding of traffic led to output congestion.

The switch port configuration only included auxiliary VLANs on the Cisco IP Phone ports. The remaining port configuration of the switch was default. Example 3-8 outlines the relevant configuration.

Example 3-8. Catalyst 4000 CatOS Switch Configuration for Case Study

```
begin

(text deleted)


#qos

set qos enable
```

```
(text deleted)

_____________________________________________________________________

#module 1 : 2-port 1000BaseX Supervisor

set vlan 70  1/1

set trunk 1/1 off dot1q 1-1005

set spantree portfast  1/1 enable

set port channel 1/1 mode off

(text deleted)

_____________________________________________________________________

#module 2 : 48-port Inline Power Module

set vlan 70  2/1

set port auxiliaryvlan 2/2 70

set port auxiliaryvlan 2/3 70

set trunk 2/1 off dot1q 1-1005

set trunk 2/2 off dot1q 1-1005

set trunk 2/3 off dot1q 1-1005

set spantree portfast  2/1-3 enable

set port channel 2/1-3 mode off

(text deleted)
```

QoS was enabled for both trials. In the first trial, however, QoS was enabled but the CoS mapping was left as default. As a result, frames of CoS values 0 through 7 mapped to the same output queue. In the second trial, frames with a CoS value of 4 to 7 were mapped to queue 1, and the remaining frames were mapped to queue 0. Example 3-9 shows the QoS configuration for each trial.

Example 3-9. Catalyst 4000 QoS CoS Mapping Configuration for Each Trial of Case Study

```
! Trial 1:
```

```
Console> (enable) show qos info runtime

Run time setting of QoS:

QoS is enabled

All ports have 2 transmit queues with 1 drop thresholds (2q1t).

Default CoS = 0

Queue and Threshold Mapping:

Queue Threshold CoS

----- --------- ---------------

1     1         0 1 2 3 4 5 6 7

2     1
```

```
! Trial 2:

Console> (enable) show qos info runtime

Run time setting of QoS:

QoS is enabled

All ports have 2 transmit queues with 1 drop thresholds (2q1t).

Default CoS = 0

Queue and Threshold Mapping:

Queue Threshold CoS

----- --------- ---------------

1     1         0 1 2 3

2     1         4 5 6 7
```

As Table 3-10 indicates, the voice stream statistical measurements clearly showed significant frame loss and poor voice quality when all frames shared the same output queue. When the frames were output scheduled appropriately, no loss of frames occurred and voice quality was excellent. The maximum jitter was well within the recommended boundary of less than 30 ms.

Table 3-10. QoS Trial Results on Catalyst 4006 with Supervisor II Engine

Trial	Total Frames Transmitted (Phone 1/2)	No. of Receive Lost Frames (Phone 1/2)	Maximum Recorded Jitter (Phone1/2)
1 queue	3245/3300	2536/2459	20/22
2 queues	3130/3240	0/0	15/9

Summary

The Catalyst 4000 CatOS Family of switches suits basic QoS needs for an access layer switch. If additional features such as policing and classification based on DSCP are required, network designers need to consider the Catalyst 4000/4500 Cisco IOS Family of switches using the Supervisor Engine III or IV. The Catalyst 4000/4500 Cisco IOS Family switches support classification based on DSCP or CoS, ingress and egress policing, and output scheduling based on a 1p3q1t or 4q1t port queuing system. You can summarize the QoS feature support on the Catalyst 4000 CatOS Family of switches as follows:

- No support for input scheduling.
- Classification based on CoS only; no support for classification based on IP precedence or DSCP.
- Extended trust options are not supported.
- Output ports have two queues with one threshold (2q1t).
- Frames are tail dropped when queue is full.
- Tail dropped frames are recorded as txQueueNotAvailable in the show counters*mod/port*.
- CoS mapping to queues are configurable in pairs: 0-1, 2-3, 4-5, and 6-7.
- Ports are trusted by default irrespective of the QoS global configuration.
- The queue threshold is not configurable.
- Untagged frames can be mapped to the queue based on configured CoS value.
- Tagged frames cannot have CoS values rewritten.
- Layer 3 services module can be added to Catalyst 4000 CatOS switch for policing of IP routed traffic between VLANs. However, the Layer 3 services module rewrites ingress CoS to zero.

In summary, the Catalyst 2900XL, 3500XL, and Catalyst 4000 CatOS Family of switches only support a subset of QoS features compared to the Catalyst 2950, 3550, 4000 IOS, and 6500 Family of switches.

The available QoS features depend on the platform; they also depend on whether the platform supports IP routing. The Catalyst 3550, Catalyst 4000 Cisco IOS Software Family, Catalyst 5500 with RSM or RSFC, and the Catalyst 6000/6500 with MSM or MSFC I/II support IP routing. Other platforms may support Layer 3 QoS features, such as classification based on DSCP and marking of IP precedence; however these platforms do not actually support routing of IP frames.

For a list of the QoS features supported by each platform, see Tables 3-1 through 3-5.

Chapter 4. QoS Support on the Catalyst 5000 Family of Switches

This chapter discusses QoS feature support on the Catalyst 5000 Family of switches. The Catalyst 5000 Family of switches supports only a small subset of QoS features. Furthermore, QoS feature support on these switches has important hardware and configuration restrictions. This chapter discusses all the restrictions and configuration requirements for implementing QoS on the Catalyst 5000 Family of switches and provides command references, examples, and a case study. Specifically, this chapter covers the following topics:

- QoS Architectural Overview
- Enabling Qos Features
- Input Scheduling
- Classification
- Marking
- Congestion Avoidance
- Case Study

Upon completion of this chapter, you will understand the restrictions surrounding QoS feature support on the Catalyst 5000 Family of switches and be able to configure the switches for packet classification, marking, and congestion avoidance.

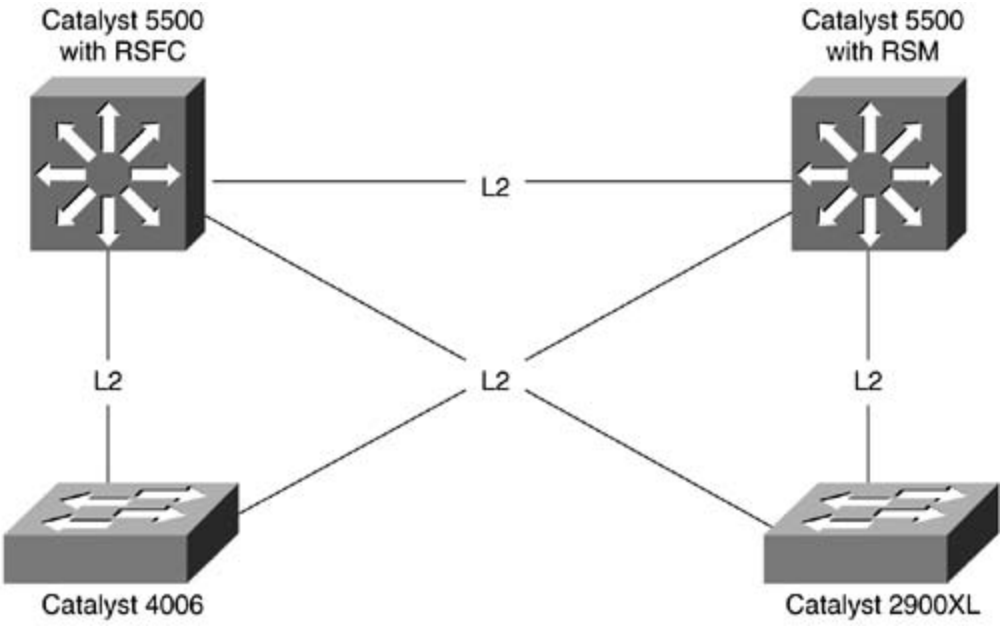
Catalyst 5000 Family of Switches QoS Architectural Overview

The Catalyst 5000 Family of switches forwards frames strictly based on MAC address and VLAN ID. To route frames with a Catalyst 5000, the switch requires a router module, either a *Route Switch Module* (RSM) or a *Router Switch Feature Card* (RSFC). Catalyst 5000 switches with a *NetFlow Feature Card* or *NetFlow Feature Card II* (NFFC or NFFC II, respectively), router module, and specific line modules can perform *multilayer switching* (MLS). MLS enables the RSM or RSFC to offload the Layer 2 rewrite functionality of routers to hardware components on the supervisor engine and line cards. In relation to QoS functionality, MLS-enabled switches handle QoS packet rewrites slightly different than do switches without MLS enabled. Because of the system architecture deployed on the Catalyst 5000 Family of switches, these switches support the following QoS features in specific hardware configurations:

- Classification of untagged frames
- Trusting of tagged frames
- *Class of service* (CoS) rewrite based on destination MAC and VLAN ID or ingress port
- CoS and *type of service* (ToS) rewrite based on *access-control entries* (ACEs)
- Output scheduling using one queue with four thresholds

For supporting QoS in network designs, especially networks delivering *Voice over IP* (VoIP) or using *DiffServ codepoint* (DSCP) for differentiated service, the preferred switches are the Catalyst 6500 Family of switches, the Catalyst 4000 IOS Family of switches, the Catalyst 3550, or the Catalyst 2950. By today's standards, the Catalyst 5000 Family of switches is a legacy product and does not support the features necessary for end-to-end QoS implementations. Furthermore, Cisco Systems, Inc., no longer sells the Catalyst 5000 Family of switches. Nonetheless, a large number of networks still use the Catalyst 5000 Family of switches; therefore, network designs need to exploit the QoS features available on these switches. Figure 4-1 illustrates a sample network design using a Catalyst 5500 switch in the core layer. The Catalyst 5500 switches include a router module for Layer 3 routing and the RSMs utilize HSRP for router redundancy. The Catalyst 5500 switches connect Layer 2 to the Catalyst 4000 CatOS switches used as access layer switches. Generally, legacy networks deploying Catalyst 5500's only utilize Layer 3 routing in the core. As a result, these networks only connect Layer 2 to access layer switches and perform no routing of traffic on the access layer switches. Although not shown on the diagram, several VLANs are deployed in this topology to minimize broadcast domains.

Figure 4-1. Sample Network Topology Using Catalyst 5000 Switches



Software Requirements

The Catalyst 5000 Family of switches requires CatOS software version 5.1(1) or higher for QoS feature support. The Catalyst 5000 Family of switches runs CatOS software up to CatOS software version 6.4. The Catalyst 5000 Family of switches does not support Native IOS software or the CatOS software version 7 train.

MLS changes the behavior of QoS on the Catalyst 5000 Family of switches. MLS support on the Catalyst 5000 Family of switches requires a NFFC and a Cisco router running Cisco IOS Software version 11.3 WA, 12.0, or higher. The recommended routers for deploying MLS are the RSMs or RSFCs. The RSM is a line module for the Catalyst 5000 Family, and the RSFC is a daughter card for the Catalyst 5000 Supervisor IIG and IIIG engines. External routers including the Cisco 3600, 7200, and 7500 support MLS as well and are alternatives to using the RSM or RSFC for MLS.

Hardware Requirements

Only a limited subset of the Catalyst 5000 Family of switches supports QoS features. To enable these QoS features, an NFFC II must be present on the supervisor engine. EARL 3 and NFFC II are the same hardware component, and the names are used interchangeably. EARL is an acronym for the Layer 2 forwarding logic on the supervisor engine and exists in several versions across all the Catalyst supervisor engines of the Catalyst 5000 Family.

The NFFC II enables QoS support for classification, marking, and congestion avoidance. The Supervisor IIIG and IIG include an NFFC II, whereas specific models of the Supervisor Engine III include the NFFC II. Table 4-1 illustrates which supervisor engines include an NFFC II. The Supervisor Engine IIIs without an NFFC II are upgradeable to include an NFFC II.

Table 4-1. NetFlow Feature Card II to Supervisor Engine Matrix

Model Number	Supervisor Engine	NetFlow Feature Card	EARL Version	EARL Subtype Model	QoS Feature Support
WS-X5550	Supervisor IIIG	NFFC II	EARL 3	WS-F5531	Yes
WS-X5540	Supervisor IIG	NFFC II	EARL 3	WS-F5531	Yes
WS-X5534	Supervisor IIIF	n/a	EARL 1++	WS-F5520	No
WS-X5530-E3	Supervisor III (NFFC II)	NFFC II	EARL 3	WS-F5531	Yes
WS-X5530-E3A	Supervisor III (NFFC II-A)	NFFC II-A	EARL 3	WS-F5531A	Yes
WS-X5530-E2	Supervisor III (NFFC)	NFFC	EARL 2	WS-F5521	No
WS-X5530-E2A	Supervisor III (NFFC A)	NFFC	EARL 2	WS-F5521A	No
WS-X5530-E1	Supervisor III	n/a	EARL 1++	WS-F5520	No
WS-X5509	Supervisor II	n/a	EARL 1+	WS-F5511	No
WS-X5506	Supervisor II	n/a	EARL 1+	WS-F5511	No
WS-X5505	Supervisor II	n/a	EARL 1+	WS-F5511	No
WS-X5009	Supervisor I	n/a	EARL 1	WS-F5510	No
WS-X5006	Supervisor I	n/a	EARL 1	WS-F5510	No
WS-X5005	Supervisor I	n/a	EARL 1	WS-F5510	No
WS-C2926G	Supervisor II	n/a	EARL 1+	WS-F5511	No
WS-C2926T	Supervisor II	n/a	EARL 1+	WS-F5511	No
WS-C2926GS	Supervisor III	NFFC II	EARL 3	WS-F5531	Yes
WS-C2926GL	Supervisor III	NFFC II	EARL 3	WS-F5531	Yes
WS-C2902	Supervisor I	n/a	EARL 1	WS-F5520	No
WS-C2901	Supervisor I	n/a	EARL 1	WS-F5520	No

Example 4-1 demonstrates use of the show module command to determine the supervisor engine model number, the EARL version subtype model, and NFFC type for a Catalyst 5000 Family of switches.

Example 4-1 shows sample output from a Supervisor Engine III with an NFFC II.

Example 4-1. Sample Output from the show module Command

```
Console> (enable) show module

Mod Slot Ports Module-Type                Model                Sub Status

--- ---- ----- ------------------------- -------------------- --- --------

1   1    2     100BaseFX MMF Supervisor  WS-X5530             yes ok

(text deleted)


Mod Module-Name         Serial-Num

(text deleted)
```

```
Mod MAC-Address(es)                        Hw     Fw          Sw

--- -------------------------------------- ------ ---------- -----------------

(text deleted)


Mod Sub-Type Sub-Model Sub-Serial Sub-Hw

--- -------- --------- ---------- ------

1   NFFC II  WS-F5531  0012152468 1.0

1   uplink   WS-U5533  0010450920 1.0
```

The output from the show module command indicates not only supervisor engine type, NFFC, and EARL versions, but also includes software version, MAC addresses, hardware and firmware revisions, and serial numbers of all line modules in the chassis. For supervisor engines without an NFFC or NFFC II, the subtype indicates EARL version.

Moreover, QoS feature support requires traffic ingress and egress from specific Catalyst 5000 line modules. Table 4-2 summarizes the modules required for front-panel support of QoS features.

Table 4-2. Required Line Modules for QoS Support

Model No.	Description
WS-U5537-FETX	4-port 10/100BASE-TX uplink module
WS-U5538-FEFX-MMF	4-port 10/100BASE-FX uplink module
WS-X5234-RJ45	24-port 10/100BASE-TX RJ-45
WS-X5236-FX-MT	24-port 100BASE-FX MT-RJ
WS-X5239-RJ21	36-port 10/100BASE-TX Telco

To obtain supported features per line module from the *command-line interface* (CLI), use the following command:

show port capabilities*mod/ports*

Example 4-2 shows sample output from the show port capabilities command for the WS-X5224 and WS-X5234 line modules, respectively. The output from the commands indicate that the WS-X5224 does not support QoS features, whereas the WS-X5234 supports CoS and ToS rewrite as well as output scheduling.

Example 4-2. Sample Output from the show port capabilities Command

```
Console> (enable) show port capabilities 2/1

Model                     WS-X5224

Port                      2/1

Type                      10/100BaseTX

Speed                     auto,10,100

Duplex                    half,full

Trunk encap type          no

Trunk mode                no

Channel                   no

Broadcast suppression     pps(0-150000),percentage(0-100)

Flow control              no

Security                  yes

Dot1x                     yes

Membership                static,dynamic

Fast start                yes

QOS scheduling            rx-(none),tx-(none)

CoS rewrite               no

ToS rewrite               no

Rewrite                   no

UDLD                      yes

AuxiliaryVlan             no
```

```
SPAN                    source,destination


Console> (enable) show port capabilities 3/1
Model                   WS-X5234
Port                    3/1
Type                    10/100BaseTX
Speed                   auto,10,100
Duplex                  half,full
Trunk encap type        802.1Q,ISL
Trunk mode              on,off,desirable,auto,nonegotiate
Channel                 3/1-2,3/1-4
Broadcast suppression   percentage(0-100)
Flow control            receive-(off,on),send-(off,on)
Security                yes
Dot1x                   yes
Membership              static,dynamic
Fast start              yes
QOS scheduling          rx-(none),tx-(1q4t)
CoS rewrite             yes
ToS rewrite             IP-Precedence
Rewrite                 yes
UDLD                    yes
AuxiliaryVlan           1..1000,untagged,dot1p,none
SPAN                    source,destination
```

Enabling QoS Features on the Catalyst 5000 Family of Switches

QoS must be globally enabled on the Catalyst 5000 Family of switches before the switch enacts on classification, marking, and output scheduling configurations. To enable QoS on the Catalyst 5000 Family of switches, enter the following command:

```
set qos {enable | disable}
```

Example 4-3 illustrates a user enabling QoS on a Catalyst 5000 Switch.

Example 4-3. Enabling QoS Features on a Catalyst 5000 Switch

```
Console> (enable) set qos enable

QoS is enabled.
```

Input Scheduling

Similar to other access layer switches, the Catalyst 5000 Family of switches performs only FIFO queuing of ingress packets. Input scheduling is not supported, but this does not pose a significant issue if traffic does not exceed the backplane bandwidth. Backplane bandwidth depends on the chassis model and supervisor engine. All Supervisor I and II Engines utilize a single 1.2-Gbps system bus for the backplane architecture. The Supervisor III Engines utilize three 1.2-Gbps system buses for total of 3.6-Gbps bandwidth in a three-system bus chassis for the backplane architecture. The three-system bus chassis encompass the 5505, 5500, and 5509 models.

The three system buses on the 5505, 5500, and 5509 chassis aggregate at the supervisor engine. The supervisor engine is responsible for moving traffic between the three buses. All Ethernet line cards except the three-port, WS-X5403, and nine-port Gigabit Ethernet modules utilize a single bus connector. With the exception of the WS-X5403 and WS-X5410, all line cards attach to the first bus connector in a slot. The WS-X5403 attaches a single front-panel Gigabit Ethernet port to each bus connector on the backplane, whereas the WS-X5410 distributes traffic from all nine front-panel Gigabit ports to the three bus connectors on the backplane. The switch references the three buses as bus A, B, and C, respectively.

To distribute traffic across the buses for single bus connecting line cards, each chassis employs a different bus connector layout per slot. In this manner, the switch distributes traffic on a line card basis by placing the three backplane bus connectors in a different order depending on a chassis slot. In a Catalyst 5505 chassis, for instance, the first bus connector in slot 3 is A, whereas slots 4 and 5 utilize B and C, respectively, as the first bus connector. Placing three single bus line cards in slots 3 to 5 results in traffic from each line card occupying a different 1.2-Gbps system bus. Tables 4-3 to 4-6 illustrate the bus layout for each of the Catalyst 5000 Family of switches chassis.

Table 4-3. Catalyst 5000 Chassis Bus Layout

Slot	Backplane Connector
1	A
2	A
3	A
4	A
5	A

Table 4-4. Catalyst 5505 Chassis Bus Layout

Slot	Backplane Connector
1	A B C
2	A B C
3	A B C
4	B A C
5	C B A

Table 4-5. Catalyst 5509 Chassis Bus Layout

Slot	Backplane Connector
1	A B C
2	A B C
3	A B C
4	A B C
5	B A C
6	B A C
7	B A C
8	C B A
9	C B A

Table 4-6. Catalyst 5500 (13-Slot) Chassis Bus Layout

Slot	Backplane Connector
1	A B C
2	A B C
3	A B C
4	A B C
5	B A C
6	B
7	B
8	B
9	B or ATM
10	C or ATM
11	C or ATM

12	C or ATM
13	Reserved for ATM switch processor

The Catalyst 5500 chassis utilizes slots 9 through 12 for ATM or Ethernet line modules. This chassis only supports an ATM switch process in slot 13 and does not accept an Ethernet line module in this slot.

In addition to referencing the system architecture, use the **whichbus** {*mod/port*} command to determine which bus a line module utilizes.

Example 4-4 illustrates a user enabling the **whichbus** {*mod/port*} command.

Example 4-4. Example of Using the whichbus Command

```
Console> (enable) whichbus 4/1

B
```

Because the Catalyst 5000 Family of switches does not support input scheduling, maintaining backplane utilization within the 1.2 Gbps per system bus is important when implementing a QoS architecture. When oversubscribing the system bus, the switch tail drops frames from front-panel ports. Therefore, to avoid the arbitrary tail dropping of frames, do not oversubscribe the bus. Use the **show traffic** command to view the current and peak loads on each system bus as demonstrated in Example 4-5.

Example 4-5. Sample Output of the show traffic Command

```
Console> (enable) show traffic

Threshold: 100%


Switching-Bus Traffic Peak Peak-Time

------------- ------- ---- -------------------------

A               3%     10% Sun Mar 30 2002, 09:52:14

B               5%     11% Sun Mar 30 2002, 09:52:14

C               17%    63% Sun Mar 30 2002, 09:52:14
```

Use the show system command to view the collective utilization of all three buses. Example 4-6 displays sample output of the show system command.

Example 4-6. Sample Output of the show system Command

```
Console> (enable) show system

(text deleted)


Modem   Baud  Traffic Peak Peak-Time

------- ----- ------- ---- -------------------------

disable  9600   7%      13% Sun Mar 30 2002, 09:52:14

(text deleted)
```

As a generally accepted practice when using the Catalyst 5000 Family of switches, avoid utilizing more than 50 percent of a system bus's bandwidth. For networks requiring more bandwidth, use the Catalyst 3550 Family of switches, the Catalyst 4000 IOS Family of switches, or the Catalyst 6500 Family of switches.

For more information regarding the system architecture of the Catalyst 5000 Family of switches, refer to the following technical documents at Cisco.com:

- "Hardware Troubleshooting for Catalyst 5500/5000/2926G/2926 Series Switches" Document ID: 18810
- "Preparing to Troubleshoot Hardware for Catalyst 5500/5000/2926G/2926 Series Switches" Document ID: 18826

Classification and Marking

The Catalyst 5000 Family of switches bases classification solely on Layer 2 CoS values. Although th
rewrite Layer 3 IP precedence values in specific configurations, the switch does not use ToS values
precedence values or DSCP values to make any QoS classification, marking, or congestion avoidan
With regard to classification, the Catalyst 5000 Family of switches is similar to the Catalyst 4000 C
of switches and the Catalyst 2900XL and 3500XL switches.

Classification and Marking of Untagged Frames Based on Ingress Port

The Catalyst 5000 switch does not support any port-level rewriting of DSCP or ToS values, the swit
rewrites the ingress frames' CoS value and does not alter the DSCP or ToS bits even on untrusted p
respect, the Catalyst 5000 Family of switches, by default, does not trust CoS values but always tru
ToS values. Nevertheless, the Catalyst 5000 Family of switches can reclassify and mark untagged f
specific CoS value similar to the Catalyst 2900XL and 3500XL switches on line modules that suppor
features as indicated in Table 4-2. The switch must transmit the frame with a dot1q tag or *Inter-Sw*
(ISL) header for the respective CoS value to be present on egress. Use the following command to c
port to classify untagged ingress frames with a specific CoS value:

set port qos*mod/ports***cos***cos_value*

cos_value represents the overriding CoS value. To clear the configuration, use the clear port qos
cos command. Example 4-7 illustrates a user configuring a switch port to classify untagged ingress
a specific CoS value and then clearing the configuration.

Example 4-7. User Configuring Port to Classify Untagged Frames with Sp Value

```
Console> (enable) set port qos 3/1 cos 5

Port  3/1 qos cos set to 5


Console> (enable) clear port qos 3/1 cos

Port  3/1 qos cos setting cleared.
```

No options exist on the Catalyst 5000 Family of switches for trusting CoS, trusting DSCP, or trustin precedence.

Classification of Tagged Frames Based on Ingress Port

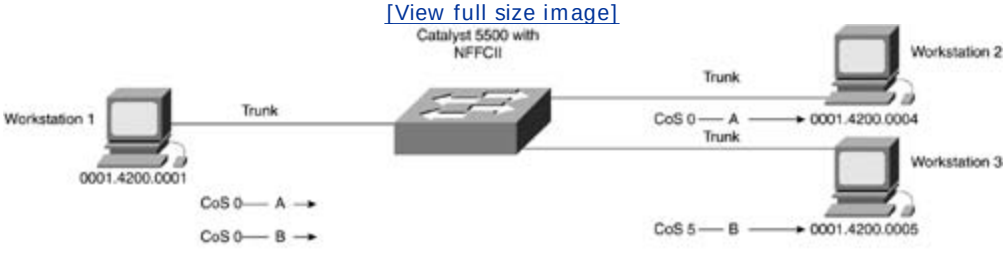
The Catalyst 5000 Family of switches trusts ISL- or dot1q-tagged frames' CoS value by default and support reclassification or marking of tagged frames based on port configuration.

Classification and Marking Based on Destination VLAN and MAC

The Catalyst 5000 Family of switches supports rewriting the CoS value of selected destination MAC VLAN basis. For a CoS value to be present on the egress frame, the switch must transmit the frame dot1q tag or ISL header on a port configured for trunking. Figure 4-2 illustrates an example of usir classification and marking based on destination VLAN and MAC. In this example, a Workstation 1 is traffic to Workstations 2 and 3 with a CoS value of 0. The configuration applied to the switch marks destined for MAC address 0001.4200.0005 with a CoS value of 5. The marking subsequently also e scheduling for the frames.

Figure 4-2. Topology Illustrating Classification and Marking Based on De MAC Address and VLAN

[View full size image]



Use the following command to configure marking based on a destination MAC address on a VLAN b

set qos mac-cos*dest_MAC_addr VLAN cos_value*

The parameters for this command are defined as follows:

- *dest_MAC_addr* represents the destination MAC address.
- *VLAN* represents the VLAN ID where the destination MAC address resides.
- *cos_value* symbolizes the CoS value to write on the frame.

Example 4-8 illustrates a user configuring classification and marking based on the destination MAC using the topology in Figure 4-2.

Example 4-8. User Configuring Classification and Marking Based on the Destination MAC Address and VLAN

```
Console> (enable) set qos mac-cos 00-01-42-00-00-05 5 4

CoS 4 is assigned to 00-01-42-00-00-05 vlan 5.
```

Classification and marking based on destination MAC address and VLAN does not scale in large top requires knowledge and continuous updates to MAC addresses. The preferable methods of classifica marking are to utilize ingress port configuration or ACEs. The next section discusses classification a based on ACEs.

Classification and Marking Based on ACE

Classification and marking based on ACE provides a method of classifying and marking IP version 4 traffic crossing a routed boundary. Traffic crossing a routed boundary always passes through a rou capable of routing in hardware. The Catalyst 5000 Family of switches may use an RSFC or RSM for functionality; however, any IP router supporting MLS is sufficient. Classification and marking based supports the following ACE options:

- IP source address(es)
- IP destination address(es)
- UDP, TCP, or both protocols
- TCP/UDP source port(s)
- TCP/UDP destination port(s)

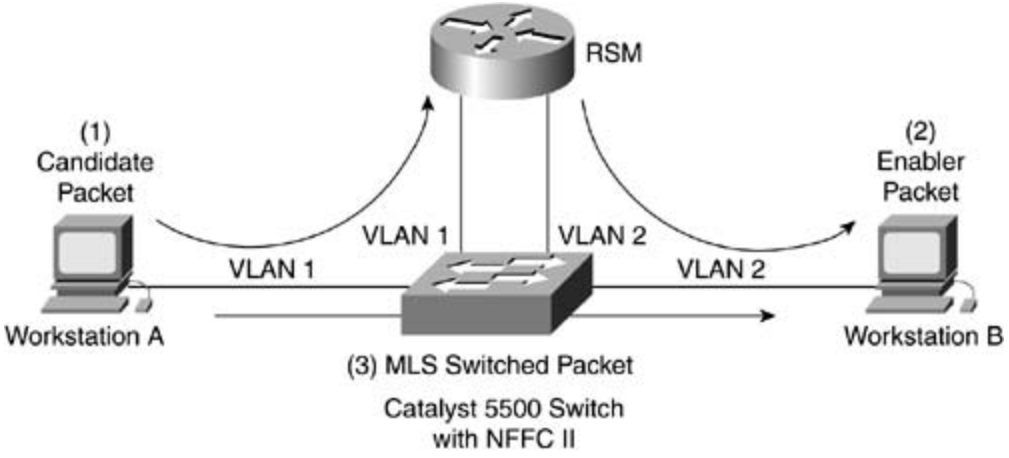
When ACE-based classification occurs, the switch marks the IP precedence bits in the IP header to CoS value. This behavior differs completely from newer platforms, such as the Catalyst 4000 IOS F switches. Later chapters discuss the differences in classification and marking behavior of the newer In addition, ACE-based marking rewrites CoS values written by the ingress port configuration or th and VLAN-based classification and marking configuration. CoS-to-IP precedence mapping or any o table is not configurable.

Furthermore, classification and marking based on ACE behaves differently depending on the MLS c
In brief, without MLS enabled, the switch carries out ACE-based classification on all traffic. With ML
the switch executes ACE-based classification only on MLS-switched traffic. The following sections pr
details on this subject.

MLS Fundamentals

As discussed in the "Catalyst 5000 Family of Switches QoS Architectural Overview" section of this c
enabling MLS allows the supervisor engine to perform Layer 2 rewrites of routed packets. Layer 2 r
include rewriting the source and destination MAC addresses and writing a recalculated *cyclic redun*
(CRC). Because the source and destination MAC address changes during Layer 3 rewrites, the switc
recalculate the CRC for these new MAC addresses. The switch learns Layer 2 rewrite information fr
router via an MLS protocol. Figure 4-3 illustrates the fundamentals behind MLS.

Figure 4-3. Logical Representation of Creating MLS Flow



InFigure 4-3, when Workstation A sends a packet to Workstation B, Workstation A sends the packe
default gateway. In Figure 4-3, the default gateway is the RSM. The switch (MLS-SE) recognizes th
an MLS candidate packet because the destination MAC address matches the MAC address of the ML
(MLS-RP). As a result, the switch creates a candidate entry for this flow. Next, the router accepts t
from Workstation A, rewrites the Layer 2 destination MAC address and CRC, and forwards the pack
Workstation B. The switch refers to the routed packet from the RSM as the enabler packet. The swi
seeing both the candidate and enabler packets, creates an MLS entry in hardware so that the switc
and forwards all future packets matching this flow. The MLS Switched Packet arrow in Figure 4-3 ir
flow. For more details and examples on the MLS architecture, consult the following technical docun
Cisco.com:

"Troubleshooting IP MultiLayer Switching" Document ID: 10554

When using the MLS feature on the Catalyst 5000 Family of switches in conjunction with QoS classi
marking based on ACEs, classification and marking only occurs on MLS-switched packets. This limi
presents the following important caveats.

- MLS-RP must see both the candidate and enabler packets to create a flow.

- Candidate and enabler packets are not subject to classification or marking based on ACEs.
- The switch removes MLS entries when the MAC Aging-time expires; therefore, flows must be s
- Packets with a destination network via a WAN port adapter on an RSM module are not subjec classification and marking based on ACEs.
- MLS ages entries based on an absolute timer; when flows age, the switch must relearn the en candidate and enabler packets.
- IP routing table changes cause all MLS entries to purge.

As a result of these limitations, a QoS implementation with ACE-based marking and MLS enabled c yields packets without the ACE-based classification and marking. For networks requiring strict appl ACE-based classification and marking, disable MLS. Disabling MLS may have side effects, such as a utilization on the MLS-RP. As a result, disabling MLS needs careful consideration and planning.

Configuring ACE-Based Classification and Marking

The switch determines which packets are destined for a router by the destination MAC address. As switch requires knowledge of the destination MAC of the router before any ACE-based classification marking occurs. Use the following commands to configure and verify the destination router MAC fo classification and marking, respectively:

set qos router-mac*MAC_addr vlan*

show qos router-mac [*MAC_addr* | *vlan*]

Configure multiple router MAC addresses for ACE-based classification and marking on multiple VLA routers.Example 4-9 illustrates a user configuring and verifying a Catalyst 5000 switch for the rout address on multiple VLANs.

Example 4-9. User Configuring and Verifying Router MAC Address for MLS

```
Console> (enable) set qos router-mac 00-30-f2-c8-8e-dc 4

Router MAC/Vlan is set for QoS.

Console> (enable) set qos router-mac 00-30-f2-c8-8e-dc 5
```

```
Router MAC/Vlan is set for QoS.


Console> (enable) show qos router-mac
Number  MAC address        Vlan #
---------------------------------
     1  00-30-f2-c8-8e-dc  4
     2  00-30-f2-c8-8e-dc  5
```

ACEs used for classification and marking utilize several options. Use the following configuration cor configure an ACE based solely on source and destination IP address and mask:

set qos ip-filter*cos src_IP_addr_spec dest_IP_addr_spec*

cos represents the CoS value that the switch writes to frames matching the ACE. *src_IP_addr_spec* the source IP address(es) and mask of the ACE, whereas *dest_IP_addr_spec* represents the destina address(es) and mask. The keyword any is optional for specifying all IP addresses, and the keywo represents an IP address for a single entry, (that is, 255.255.255.255 mask). Enter the host keyw entering the IP address. Example 4-10 illustrates two example ACE entries using the host and any respectively.

Example 4-10. ACE Example Using host and any Keywords

```
Console> (enable) set qos ip-filter 5 any host 10.10.10.10
qos ip-filter is set successfully.

Console> (enable) set qos ip-filter 5 192.168.1.0 255.255.255.0 any
qos ip-filter is set successfully.
```

Use the following command structure for specific TCP and UDP ports for an ACE:

```
set qos ip-filtercos {tcp | udp | any}src_IP_addr_spec src_port dest_IP_addr_spec
➥dest_port src_IP_addr_spec/dest_IP_addr
```

Theany keyword proceeding the CoS value is available to specify the ACE for both TCP and UDP pr
Example 4-11 illustrates several examples of using ACEs with TCP and UDP port number reference

Example 4-11. ACE Example Using TCP and UDP Ports

```
Console> (enable) set qos ip-filter 5 UDP any 30000 any 30000
Warning: This command will only apply to Unicast addresses.
qos ip-filter is set successfully.

Console> (enable) set qos ip-filter 4 TCP 192.168.100.0 255.255.255.0 16000 192.16
➥255.255.255.0 16000
qos ip-filter is set successfully.

Console> (enable) set qos ip-filter 6 any host 10.1.1.1 50000 host 10.1.1.2 50000
qos ip-filter is set successfully.
```

Use the following command to view the current configured ACEs on the switch:

show qos ip

Example 4-12 illustrates an example of using the show qos ip command.

Example 4-12. Example Output from the show qos ip Command

```
Console> (enable) show qos ip
There are 3 IP filter(s).
ACE# Src IP and Mask                  Dest IP and Mask
---- -------------------------------- --------------------------------
   1 any                               host 10.10.10.10
     Protocol Src Port Dst Port CoS
     -------- -------- -------- ---
     any      0        0        5




   2 192.168.1.0 255.255.255.0        any
     Protocol Src Port Dst Port CoS
     -------- -------- -------- ---
     any      0        0        5




   3 any                               any
     Protocol Src Port Dst Port CoS
```

```
    -------- -------- -------- ---
    udp      30000    30000    5


  4 192.168.100.0 255.255.255.0      192.168.101.0 255.255.255.0
    Protocol Src Port Dst Port CoS
    -------- -------- -------- ---
    tcp      16000    16000    4


  5 host 10.1.1.1                    host 10.1.1.2
    Protocol Src Port Dst Port CoS
    -------- -------- -------- ---
    any      50000    50000    6
```

The switch executes ACE entries in order, the same as access-control lists in Cisco IOS Software. A may be necessary to rearrange the order of the ACE entries. Use the following suffixes to the set q command to place ACE entries in the configuration in a specific order:

[**before***ACE#* | **modify***ACE#*]

Example 4-13 illustrates a user determining the existing ACE order, placing an ACE entry in a spec and verifying the configuration.

Example 4-13. Placing ACE Entries in Specific Order

```
Console> (enable) show qos ip
There are 3 IP filter(s).
ACE# Src IP and Mask                  Dest IP and Mask
---- -------------------------------- --------------------------------
   1 any                               host 10.10.10.10
     Protocol Src Port Dst Port CoS
     -------- -------- -------- ---
     any      0        0        5




   2 192.168.1.0 255.255.255.0        any
     Protocol Src Port Dst Port CoS
     -------- -------- -------- ---
     any      0        0        5




   3 any                               any
     Protocol Src Port Dst Port CoS
     -------- -------- -------- ---
     udp      30000    30000    5




   4 192.168.100.0 255.255.255.0      192.168.101.0 255.255.255.0
```

```
     Protocol Src Port Dst Port CoS
     -------- -------- -------- ---
     tcp      16000    16000    4


   5 host 10.1.1.1                   host 10.1.1.2
     Protocol Src Port Dst Port CoS
     -------- -------- -------- ---
     any      50000    50000    6


Console> (enable) set qos ip-filter 3 192.168.20.0 0.0.0.255 192.168.21.0 0.0.0.25

Console> (enable) show qos ip
There are 5 IP filter(s).
ACE# Src IP and Mask                   Dest IP and Mask
---- --------------------------------- ---------------------------------
   1 any                               host 10.10.10.10
     Protocol Src Port Dst Port CoS
     -------- -------- -------- ---
     any      0        0        5


   2 192.168.1.0 255.255.255.0         any
     Protocol Src Port Dst Port CoS
     -------- -------- -------- ---
```

```
  any      0        0        5


3 192.168.20.0 0.0.0.255          192.168.21.0 0.0.0.255
  Protocol Src Port Dst Port CoS
  -------- -------- -------- ---
  any      0        0        3


4 any                             any
  Protocol Src Port Dst Port CoS
  -------- -------- -------- ---
  udp      30000    30000    5


5 192.168.100.0 255.255.255.0     192.168.101.0 255.255.255.0
  Protocol Src Port Dst Port CoS
  -------- -------- -------- ---
  tcp      16000    16000    4


6 host 10.1.1.1                   host 10.1.1.2
  Protocol Src Port Dst Port CoS
```

```
-------- -------- -------- ---
any      50000    50000    6
```

Use the following commands to clear ACE entries:

clear qos ip-filter*ace_num*

clear qos ip-filter all

Extended Trust Option

The Catalyst 5000 Family of switches supports instructing an attached appliance of extended trust the*Cisco Discovery Protocol* (CDP). CDP is a Layer 2 protocol used to inform Cisco devices of each p parameters such as IP address, system name, Native VLAN, and egress port. Chapter 2, "End-to-E Quality of Service at Layer 3 and Layer 2," in the "Voice VLANs and Extended Trust" section discuss concept of extended trust. The following commands configure the extended trust options for a Cata switch:

set port qos*mod/ports***cos-ext***cos_value*

set port qos*mod/ports***trust-ext [trust-cos | untrusted]**

cos_value represents the requested reclassification CoS value sent to the attached appliance via CD trust-cos option requests the attached appliance to trust ingress CoS values on frames, whereas u signifies the appliance to not trust the CoS value in ingress frames and rewrite any ingress CoS val The most common appliance is the Cisco IP Phone.

Although the Catalyst 5000 Family of switches supports extended trust parameters in CDP messag attached appliances and auxiliary (voice) VLANs, it does not support the ability to provide power o

cabling infrastructure to power Cisco IP Phones or other appliances. For example, if you are using t
5000 Family of switches to apply extended trust parameters in CDP messages to a Cisco IP Phone,
power outlet for the phone rather than relying on inline power.

Congestion Avoidance

The Catalyst 5000 Family of switches achieves congestion avoidance through the use of queue management on certain line modules based on CoS values. No support for output scheduling or queuing based on DSCP or IP precedence values exists for the Catalyst 5000 Family of switches. Th two uplink modules and three line modules described in Table 4-2 represent the only line modules support output scheduling. All other line modules only utilize a single transmit queue, and the transmit queue tail drops all frames when exhausting buffer queue space.

The uplink modules and line modules in Table 4-2 utilize a single transmit queue with four configurable transmit queue drop thresholds, 1q4t, to achieve congestion avoidance. The switch drops packets with specific CoS values when the transmit buffer reaches specific thresholds. The Co values assigned to each threshold are configurable. This method of congestion avoidance delivers *Weighted Random Early Detection* (WRED) for a single queue. Chapter 1, "Quality of Service: An Overview," discussed WRED in more detail.

For example, Table 4-7 describes the default behavior of congestion management on the Catalyst 5000 Family of switches.

Table 4-7. Default Output Scheduling Behavior of Ports Supporting 1q4 Transmit Queue

Threshold No.	Threshold of Transmit Queue Buffer	CoS Values
1	30% Full	0 and 1
2	50% Full	2 and 3
3	80% Full	4 and 5
4	100% Full	6 and 7

Use the following command to configure the CoS values that map to a specific threshold number:

set qos map*port_type q# threshold#***cos***cos_list*

port_type is always 1q4t and *q#* is always 1 on the Catalyst 5000 Family of switches. The *threshold#* represents one of four thresholds used for CoS value mapping. After making configuration changes to the transmit queue, use the following commands to verify the configuratio

changes:

show qos info <**runtime** | **config**> *<mod/port>*

show qos info config*port_type tx*

Example 4-14 illustrates a user configuring and verifying the QoS threshold number to CoS mappir Only module 3 of this chassis supports QoS features; and therefore, the switch warns that QoS is n supported on modules 1, 2, 4, and 15.

Example 4-14. User Configuring and Verifying QoS Threshold to CoS Mapping

```
Console> (enable) set qos map 1q4t 1 1 cos 0-2
QoS is not supported on module 1.
QoS is not supported on module 2.
QoS is not supported on module 4.
QoS is not supported on module 15.
Qos tx priority queue and threshold mapped to cos successfully.

Console> (enable) set qos map 1q4t 1 2 cos 4
QoS is not supported on module 1.
QoS is not supported on module 2.
QoS is not supported on module 4.
QoS is not supported on module 15.
Qos tx priority queue and threshold mapped to cos successfully.
```

```
Console> (enable) show qos info config 1q4t tx

QoS setting in NVRAM for 1q4t transmit:

QoS is enabled

Queue and Threshold Mapping:

Queue Threshold CoS

----- --------- ---------------

1     1         0 1 2

1     2         3 4

1     3         5

1     4         6 7

Queue #  Thresholds - percentage

-------  ----------------------------------------

1        20% 40% 50% 100%
```

The threshold's values are also configurable on a global basis. Use the following command to configure the threshold percentage drop values:

set qos wred-threshold*port_type***tx queue***q# threshold_percentage_values*

As with all Catalyst 5000 QoS commands, *port_type* is always 1q4t and *q#* is always 1. *threshold_percentage_values* represents the four threshold drop values. Example 4-15 illustrates a user configuring the threshold drop values.

Example 4-15. User Configuring Drop Thresholds for Transmit Queues

```
Console> (enable) set qos wred-threshold 1q4t tx queue 1 20 30 40 95
```

```
WRED thresholds for queue 1 set to 20 and 30 and 40 and 95  on all WRED-capable 1q
  ports.
```

To view the number of packets dropped per threshold number, use the following command:

show qos statistics*mod_num/port_num*

Example 4-16 illustrates the use of the show qos statistics command.

Example 4-16. User Displaying the Number of Dropped Packet per Threshold Number

```
Console> (enable) show qos statistics 3/24
On Transmit: Port 3/24 has 1 Queue(s) 4 Threshold(s)
Q #  Threshold #:Packets dropped
---  ----------------------------------------------
1    1:59378 pkts, 2:14 pkts, 3:7 pkts, 4:8 pkts
```

For IP telephony networks, a CoS value of 5 generally represents VoIP traffic. As a result, the desir behavior is to never drop VoIP traffic unless the output buffer queue is full. As a result, use the following best practice configuration for switch ports connected to IP telephony devices, such as Cisco IP Phones:

Example 4-17. Best Practice Output Scheduling Configuration for IP Telephony Networks

```
!
```

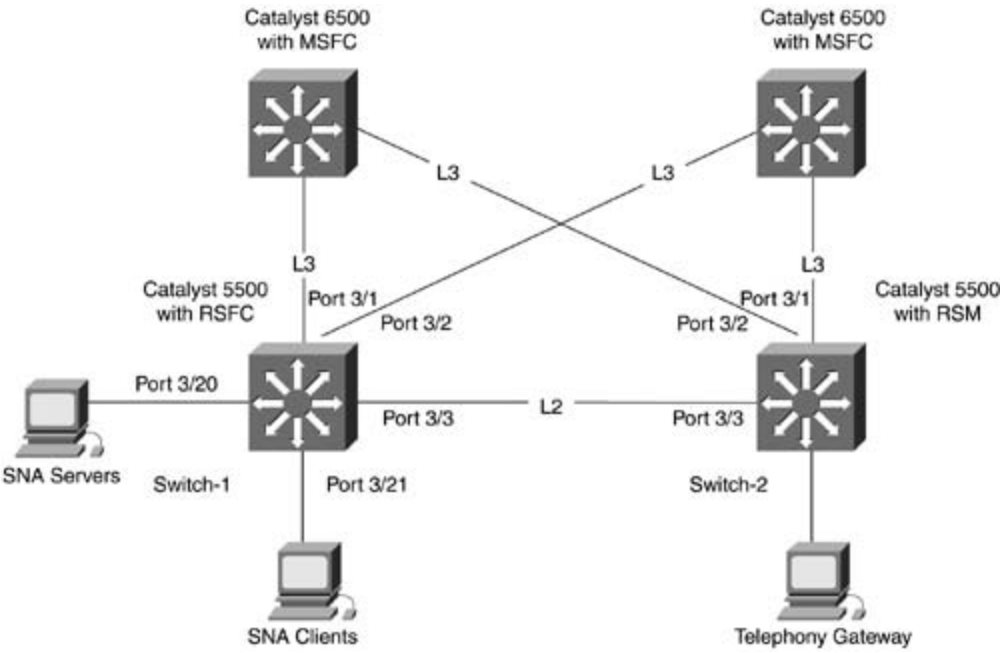
```
Console> (enable) show config
This command shows non-default configurations only.
Use 'show config all' to show both default and non-default configurations.
(text deleted)
!
#qos
set qos enable
set qos map 1q4t 1 1 cos 2
set qos map 1q4t 1 2 cos 4
set qos wred-threshold 1q4t tx queue 1 20 30 100 100
!
(text deleted)
end

Console> (enable) show qos info config 1q4t tx
QoS setting in NVRAM for 1q4t transmit:
QoS is enabled
Queue and Threshold Mapping:
Queue Threshold CoS
----- --------- ---------------
1     1         0 1 2
1     2         3 4
1     3         5
1     4         6 7
Queue #  Thresholds - percentage
-------  ----------------------------------------
1        20% 30% 100% 100%
```

The configuration in Example 4-17 assigns traffic with CoS values of 5 to threshold #3. In this manner, the switch drops traffic for CoS values 5, 6, and 7 only when the output buffer is full. In addition, the switch aggressively drops low-priority traffic for CoS values 0 to 4 when the queue is 20 percent to 30 percent full to avoid output buffer full conditions from ever occurring.

Case Study

Figure 4-4 illustrates a network that consists of two Catalyst 5000 switches implementing several QoS features to differentiate service in network.

Figure 4-4. Case Study Topology



A time-sensitive application using the *Systems Network Architecture* (SNA) protocols exists in the network depicted in Figure 4-4. In this topology, the application handles banking transactions. The client and servers send only untagged frames. To classify frames accordingly, the Switch-1 configuration consists of classification based on ingress port. This configuration is illustrated by the set port qos command in Example 4-17.

Furthermore, a telephony gateway exists in the network. The telephony gateway sends keepalive packets to other telephony devices across VLAN boundaries to the core of the network. The telephony gateway transmits all keepalives with UDP source and destination ports of 25000. The telephony gateway does not tag the frames, and as a result does not apply a CoS value to transmitted frames. To classify the frames appropriately, both Catalyst 5500 switches use ACL-based classification and marking to set the CoS value of frames matching the signature of the telephony keepalives. As a result, these switches only classify the keepalive frames and not regular traffic. Example 4-17 illustrates this ACL-based classification configuration for Switch-1.

For appropriate congestion management of frames, the switch-to-switch connections and the ports connecting the SNA servers and the telephony gateway all utilize WRED scheduling. The WRED scheduling is nondefault to apply heavier weighted scheduling to packets with a CoS value of 5. Example 4-18 shows this configuration for Switch-1.

Example 4-18. Case Study Configuration for Switch-1

```
Switch> (enable) show config
This command shows non-default configurations only.
Use 'show config all' to show both default and non-default configurations.
(text deleted)
begin
!
# ***** NON-DEFAULT CONFIGURATION *****
!
(text deleted)
!
#qos
set qos enable
set qos ip-filter 5 udp any 25000 any 25000
set qos map 1q4t 1 1 cos 2
set qos map 1q4t 1 2 cos 4
set qos wred-threshold 1q4t tx queue 1 20 30 100 100
!
# default port status is enable
!
!
#module 1 : 2-port 1000BaseX Supervisor IIIG
set vlan 5    1/1-2
set trunk 1/1  off negotiate 1-1005
set trunk 1/2  off negotiate 1-1005
!
#module 3 : 24-port 10/100BaseTX Ethernet
```

```
set vlan 1    3/1
set vlan 2    3/2
set vlan 4    3/20-24
set vlan 5    3/3-19
set port qos 3/20-21 cos 5
!
(text deleted)!
#module 15 : 1-port Route Switch Feature Card
!
#module 16 empty
!
(text deleted)
!
#qos router-mac
set qos router-mac 00-30-f2-c8-8e-dc 5
set qos router-mac 00-30-f2-c8-8e-dc 4
end
```

Summary

The Catalyst 5000 Family of switches supports QoS with only specific Supervisors and line modules. These switches support QoS features limited to classification of untagged frames, marking based on ACEs, and output scheduling. These features classify frames solely on CoS values. For most applications, DSCP classification is desirable. As a result, network designs should not introduce the Catalyst 5000 Family of switches into a QoS end-to-end architecture; instead, utilize the Catalyst 5000 Family of switches only in existing deployments. In summary, the QoS feature support on the Catalyst 5000 Family of switches is as follows:

- QoS support requires one of the following Supervisors: Supervisor III with NFFC II, Supervisor IIG, or Supervisor IIIG.
- QoS support requires traffic ingress and egress using the following line modules or uplink modules: WS-X5234-RJ45, WS-X5236-FX-MT, WS-X5239-RJ21, WS-U5537-FETX, or WS-U5538-FEFX-MMF.
- No support for input scheduling exists.
- Classification based on CoS only; no support for classification based on IP precedence or DSCP.
- Marking of CoS and ToS supported with limitations with regard to MLS.
- CoS value determines ToS value for ACE-based marking.
- Congestion avoidance achieved via WRED using a 1q4t mechanism.
- Global mapping of CoS values to thresholds is configurable.
- Thresholds are configurable on a global basis.

Part II: Advanced QoS Concepts

Chapter 5. Introduction to the Modular QoS Command-Line Interface

Not long ago, a unique *command-line interface* (CLI) existed for every QoS mechanism. This meant that if a user wanted to configure traffic shaping and congestion management, that user had to learn two totally different CLI structures. This made being proficient in the configuration of all available QoS mechanisms quite a challenge. Fortunately, Cisco IOS now has a better method for configuring the components of QoS. This new method is called the *Modular QoSCLI* (MQC).

This chapter covers the following aspects of MQC:

- MQC features and concepts
- Class maps
- Policy maps
- Servic policies

MQC Background, Terms, and Concepts

The MQC was originally introduced to support the configuration of *Class-Based Weighted Fair Queuing* (CBWFQ), but has now been expanded to include support for the configuration of nearly every QoS component. Although this list is not comprehensive, some of the QoS components that can be configured via the MQC include the following:

- Traffic classification
- Traffic marking
- Congestion management
- Congestion avoidance
- Traffic conditioning
- Header compression

The fact that so many different components of QoS can be configured via the MQC, and the fact that each of the components previously listed has several possible mechanisms, makes the MQC one of the most versatile parts of Cisco IOS. When discussing technology, versatility usually means complexity, which is one of the amazing things about the MQC. The MQC was specifically designed to *reduce* configuration complexity *and* provide versatility.

The simplicity of configuration is made possible through the use of a common configuration structure for all QoS components within the MQC. That is, the basic configuration steps for configuring all QoS mechanisms is the same, with only small variations in the configuration that are specific to the actual mechanism. You can configure all the mechanisms through a three-step process:

Step 1. Class map configuration

Step 2. Policy map configuration

Step 3. Service policy application

This chapter describes each step in the configuration process, as well as the configuration options, and also provides examples of MQC commands.

Step 1: The Class Map

The first step for configuring any QoS mechanism in the MQC is the configuration of a class-map. Simply stated, the *class map* defines which traffic you want the router to match. This is the fundamental step that allows the router to differentiate one traffic type from another. This is traffic classification, and without classification there can be no QoS.

To differentiate traffic, it is possible to match on one traffic characteristic or multiple characteristics. If you need to differentiate between traffic from 10.1.1.1 and traffic from 10.1.1.2, for example, the source IP address is the only characteristic that you need to configure. If you have multiple traffic streams from 10.1.1.1 and need to differentiate between those, however, as well as differentiate between multiple streams from 10.1.1.2, you probably need to classify traffic based on multiple criteria, such as TCP or UDP port.

A possible scenario in which this would come into play might be server 10.1.1.1 that serves production HTTP and FTP to the Accounting department, and server 10.1.1.2 that serves nonproduction HTTP and FTP to the IT group that develops applications for the Accounting department. Understanding that production traffic is the top priority, the development group needs their traffic to have a minimum bandwidth guarantee to enable that group to properly test a new HTTP application before delivering it to the Accounting department for production use. This means that there will be QoS requirements for *all* traffic from 10.1.1.1 and *some* traffic from 10.1.1.2. As such, just matching by IP address does not suffice. In this case, there is a requirement to match on multiple characteristics.

Configuring the Class Map

When matching on multiple characteristics, two options exist for creating the class map: match-any and match-all, as demonstrated in Example 5-1.

Example 5-1. Class Map Options

```
R1(config)#class-map ?

  WORD       class-map name

  match-all  Logical-AND all matching statements under this classmap

  match-any  Logical-OR all matching statements under this classmap
```

The match-any option is a logical OR operation, in which only one of the match conditions must be met for a packet to belong to a specific class. The match-all option is a logical OR operation, in which all match criteria must be met for a packet to belong to a specific class. You must choose one of these options before you configure the remaining class map parameters.

This section discusses the various configuration parameters for the class map, followed by a configuration example. The distinction between match-any and match-all is discussed as part

of the configuration example later in this chapter.

In the examples that follow, the match-all parameter is used.

```
R1(config)#class-map match-all ?
  WORD  class-map name
```

Next it is necessary to provide a name for the class map. For this example, the class map is named HTTP.

```
R1(config)#class-map match-all HTTP
R1(config-cmap)#
```

Notice that this router is now in config-cmap, which is class map configuration mode. The class map has now been created, but it does not have any information about which traffic it should pay attention to. There are, however, a few options within the class map other than which traffic to match, as the output in Example 5-2 demonstrates.

Example 5-2. Additional Class Map Options

```
R1(config-cmap)#?
QoS class-map configuration commands:
  description  Class-Map description
  exit         Exit from QoS class-map configuration mode
  match        classification criteria
```

```
  no            Negate or set default values of a command
  rename        Rename this class-map
```

It is possible, and highly recommended, to enter a description of the class map, using the description command. Use this in the same way that you would use the description command on an interface, and with the understanding that it has no actual impact on traffic. It is also possible to rename this class map, without losing the configuration of the class map. For example, if you decide that just calling the class HTTP is ambiguous, and decide that you want the class to be called ACCOUNTING-HTTP, you can change the name without destroying the configuration. This command was not always available, and it is a very convenient addition to the MQC.

```
R1(config-cmap)#rename ?
  WORD  new class-map name
```

The major function of the class map, however, is to match traffic. As such, the match argument has quite a few options. As with any software package, new features are constantly being added to Cisco IOS Software, so the list is constantly expanded. Example 5-3 shows a sample of the commands available.

Example 5-3. Options for the match Command

```
R1(config-cmap)#match ?
  access-group         Access group
  any                  Any packets
  class-map            Class map
  cos                  IEEE 802.1Q/ISL class of service/user priority values
  destination-address  Destination address
  fr-de                Match on Frame-relay DE bit
```

```
  input-interface      Select an input interface to match
  ip                   IP specific values
  mpls                 Multi Protocol Label Switching specific values
  not                  Negate this match result
  protocol             Protocol
  qos-group            Qos-group
  source-address       Source address
```

Within some of these match possibilities are additional match options. An example of that is the match ip option shown in Example 5-4.

Example 5-4. Class Map match Option

```
R1(config-cmap)#match ip ?
  dscp        Match IP DSCP (DiffServ CodePoints)
  precedence  Match IP precedence
  rtp         Match RTP port nos
```

You can view a list of the available options by using the interactive help menu.

Class Map Options: A Closer Look

The values on which the class map can be configured to match are explained in the previous output, but this list provides more detailed explanations of some of these options:

- access-group— Allows the configuration of an *access-control list* (ACL) and the matching of the criteria defined within that ACL.
- any— Allows for the matching of all packets. This option is typically used in a last-resort class, where the intention is to have all traffic not classified elsewhere put into a given class. This can also be used effectively with the not keyword, which is explained later in this list.
- class-map— This is the only way to combine the functions of the match-any and match-all class maps into a single match. Suppose, for instance, that you want traffic that is from either 10.1.1.1 or 10.1.1.2 *and* the destination IP address is 10.2.2.1 to match this class.

You could accomplish this by having a match-any class with an access-group statement to match the source IP address of 10.1.1.1 (defined by an ACL) and an access-group statement to match the source IP address of 10.1.1.2 (defined by another ACL). Then, a match-all class would be configured with one condition being that it matches the first class map and another condition being that it matches an ACL that defines the destination IP address of 10.2.2.1. Example 5-5 shows the needed configuration.

Example 5-5. Combining match-any and match-all Options

```
Router(config)#class-map match-any SOURCES
Router(config-cmap)#match access-group 103
Router(config-cmap)#match access-group 104
Router(config-cmap)#exit
Router(config)#class-map match-all DESTINATION
Router(config-cmap)#match class-map SOURCES
Router(config-cmap)#match access-group 105
Router(config-cmap)#exit
R1(config)#access-list 103 permit ip host 10.1.1.1 any
R1(config)#access-list 104 permit ip host 10.1.1.2 any
R1(config)#access-list 105 permit ip any host 10.2.2.1
```

- cos— Allows for the matching of the Layer 2 *class of service* (CoS) value in either the 802.1Q or *Inter-Switch Link* (ISL) header.
- destination-address— Don't be fooled, this is not for matching the destination IP address. This allows matching of the destination MAC address.
- fr-de— Allows for matching of packets that have the Frame Relay *discard eligible* (DE) bit set.
- input-interface— Allows for the matching of packets based on the interface through which they entered the router. This is useful in an environment in which multiple remote sites are connected via Frame Relay to a head-end router. In that environment, provided that point-to-point Frame Relay with subinterfaces is configured, all traffic from a given remote site can be matched based on the subinterface on which that traffic arrived.
- ip— Allows for matching based on the IP-specific values previously listed (IP precedence, *DiffServ codepoint* [DSCP], RTP port numbers).
- mpls— Currently, this allows only for the matching of packets based on the *multiprotocol*

label switching (MPLS) experimental bits. The command has multiple levels to provide flexibility for future matching of other MPLS-specific values.

- not— This is a very useful option, which can be used with the match-any option in a match-all class map to match all packets *except* those listed in the not criteria. For example, you could match all packets (using match-any) and then configure a match not statement for access group 101, which would match all packets *not* belonging to access group 101.

- protocol— Allows for the matching of certain predefined protocols. *Network-based application recognition* (NBAR) is a new classification engine that can recognize a large number of applications based on both static and dynamically assigned port numbers. On routers that support NBAR, the list of protocols is extensive, as shown in Example 5-6.

Example 5-6. match protocol Possibilities Where NBAR Is Supported

```
R1(config-cmap)#match protocol ?
  aarp              AppleTalk ARP
  apollo            Apollo Domain
  appletalk         AppleTalk
  arp               IP ARP
  bgp               Border Gateway Protocol
  bridge            Bridging
  bstun             Block Serial Tunnel
  cdp               Cisco Discovery Protocol
  citrix            Citrix Traffic
  clns              ISO CLNS
  clns_es           ISO CLNS End System
  clns_is           ISO CLNS Intermediate System
  cmns              ISO CMNS
  compressedtcp     Compressed TCP
  cuseeme           CU-SeeMe desktop video conference
  custom-01         Custom protocol custom-01
  custom-02         Custom protocol custom-02
```

```
custom-03          Custom protocol custom-03
custom-04          Custom protocol custom-04
custom-05          Custom protocol custom-05
custom-06          Custom protocol custom-06
custom-07          Custom protocol custom-07
custom-08          Custom protocol custom-08
custom-09          Custom protocol custom-09
custom-10          Custom protocol custom-10
decnet             DECnet
decnet_node        DECnet Node
decnet_router-l1   DECnet Router L1
decnet_router-l2   DECnet Router L2
dhcp               Dynamic Host Configuration
dlsw               Data Link Switching
dns                Domain Name Server lookup
egp                Exterior Gateway Protocol
eigrp              Enhanced Interior Gateway Routing Protocol
exchange           MS-RPC for Exchange
fasttrack          FastTrack Traffic - KaZaA, Morpheus, Grokster...
finger             Finger
ftp                File Transfer Protocol
gnutella           Gnutella Traffic - BearShare,LimeWire, Gnutella...
gopher             Gopher
gre                Generic Routing Encapsulation
http               World Wide Web traffic
icmp               Internet Control Message
imap               Internet Message Access Protocol
ip                 IP
```

ipinip	IP in IP (encapsulation)
ipsec	IP Security Protocol (ESP/AH)
ipx	Novell IPX
irc	Internet Relay Chat
kerberos	Kerberos
l2tp	L2F/L2TP tunnel
ldap	Lightweight Directory Access Protocol
llc2	llc2
napster	Napster Traffic
netbios	NetBIOS
netshow	Microsoft Netshow
nfs	Network File System
nntp	Network News Transfer Protocol
notes	Lotus Notes(R)
novadigm	Novadigm EDM
ntp	Network Time Protocol
pad	PAD links
pcanywhere	Symantec pcANYWHERE
pop3	Post Office Protocol
pppoe	PPP over Ethernet
pptp	Point-to-Point Tunneling Protocol
printer	print spooler/lpd
qllc	qllc protocol
rcmd	BSD r-commands (rsh, rlogin, rexec)
realaudio	Real Audio streaming protocol
rip	Routing Information Protocol
rsrb	Remote Source-Route Bridging
rsvp	Resource Reservation Protocol

```
secure-ftp          FTP over TLS/SSL
secure-http         Secured HTTP
secure-imap         Internet Message Access Protocol over TLS/SSL
secure-irc          Internet Relay Chat over TLS/SSL
secure-ldap         Lightweight Directory Access Protocol over TLS/SSL
secure-nntp         Network News Transfer Protocol over TLS/SSL
secure-pop3         Post Office Protocol over TLS/SSL
secure-telnet       Telnet over TLS/SSL
smtp                Simple Mail Transfer Protocol
snapshot            Snapshot routing support
snmp                Simple Network Management Protocol
socks               SOCKS
sqlnet              SQL*NET for Oracle
sqlserver           MS SQL Server
ssh                 Secured Shell
streamwork          Xing Technology StreamWorks player
stun                Serial Tunnel
sunrpc              Sun RPC
syslog              System Logging Utility
telnet              Telnet
tftp                Trivial File Transfer Protocol
vdolive             VDOLive streaming video
vines               Banyan VINES
vofr                voice over Frame Relay packets
xns                 Xerox Network Services
xwindows            X-Windows remote access
```

On routers that do not support NBAR, the list is quite a bit smaller, as shown in Example 5-

7.

Example 5-7. match protocol Possibilities Where NBAR Is Not Supported

```
R1(config-cmap)#match protocol ?
  aarp              AppleTalk ARP
  apollo            Apollo Domain
  appletalk         AppleTalk
  arp               IP ARP
  bridge            Bridging
  bstun             Block Serial Tunnel
  cdp               Cisco Discovery Protocol
  clns              ISO CLNS
  clns_es           ISO CLNS End System
  clns_is           ISO CLNS Intermediate System
  cmns              ISO CMNS
  compressedtcp     Compressed TCP
  decnet            DECnet
  decnet_node       DECnet Node
  decnet_router-l1  DECnet Router L1
  decnet_router-l2  DECnet Router L2
  dlsw              Data Link Switching
  ip                IP
  ipv6              IPV6
  ipx               Novell IPX
  llc2              llc2
  pad               PAD links
  qllc              qllc protocol
```

```
rsrb              Remote Source-Route Bridging

snapshot          Snapshot routing support

stun              Serial Tunnel

vines             Banyan VINES

vofr              voice over Frame Relay packets

xns               Xerox Network Services
```

- qos-group— Allows for the matching of a packet based on its qos-group marking. QoS groups are locally significant to a given router and are not carried with the packet once it leaves the router. Further, QoS groups are not mathematically significant. That is, a packet belonging to QoS Group 1 is no more, or less, important than a packet belonging to QoS Group 2. Assigning a packet to a QoS group is fairly simple as Example 5-8 demonstrates.

Example 5-8. Assigning a Packet to a QoS Group

```
route-map set-qos-group permit 10

match community 3

set ip qos-group 30
```

- source-address— Again, don't be fooled by the name of this option. This is for matching the source MAC address, not the source IP address.

Class Map Configuration Example

The most commonly used method is to match an ACL through the access-group option. Do not be fooled by the options for matching source and destination address. Those options refer to the MAC address; so if you want to match source or destination IP address, you will have to use an ACL. In the Accounting department example given earlier in the chapter, both the source address and a port number (either for HTTP or FTP) need to be matched. Because the class map configuration does not provide the capability to do that directly, ACLs are required for each. Example 5-9 shows traffic matching using ACLs.

Example 5-9. Using ACLs to Match Traffic

```
R1(config)#access-list 101 permit tcp any any eq www
```

```
R1(config)#access-list 102 permit tcp any any eq ftp

R1(config)#access-list 103 permit ip host 10.1.1.1 any

R1(config)#access-list 104 permit ip host 10.1.1.2 any
```

ACL 101 matches all HTTP traffic; ACL 102 matches all FTP traffic (on port 20 only), ACL 103 matches all traffic from 10.1.1.1, and ACL 104 matches all traffic from 10.1.1.2. The trick now is to combine the ACLs and class map configuration commands correctly, to achieve the desired result.Example 5-10 shows the class map configuration for matching HTTP traffic from 10.1.1.1.

Example 5-10. Class Map Configuration for Matching HTTP Traffic from a Specific Network

```
R1(config)#class-map match-all HTTP

R1(config-cmap)#match access-group 101

R1(config-cmap)#match access-group 103
```

This is where the difference between match-any and match-all comes into play. As highlighted earlier in the chapter, match-any is a logical OR operation, meaning if access group 101 *or* access group 103 are a match, the traffic belongs to this class. In this case, that would not be the desired behavior, because the intent is to match only traffic that matches *both* ACLs. That behavior is accomplished through the use of a class-map match-all, which is a logical AND operation. That means that access group 101 *and* access group 103 must be match for the traffic to belong to this class.

You can verify the configured class map with the show class-map command, as demonstrated inExample 5-11.

Example 5-11. Verifying the Class Map

```
R1#show class-map

 Class Map match-all HTTP (id 2)

   Match access-group  101

   Match access-group  103
```

```
 Class Map match-any class-default (id 0)

   Match any
```

The output from show class-map displays all the configured classes (in this case, there is only one), whether classes are a match-any or a match-all class, what the name of each class is, and which traffic belongs in those classes. Overall, this is a very useful command.

Also notice the class-default, which is automatically created whenever any other class is created. The purpose of class-default is to give all traffic that does not belong to any other class a place to go.

The ID numbers that are given in parentheses next to each class cannot be changed and have no meaning, other than to assist Cisco IOS Software in keeping the classes organized.

Going back to the Accounting department example, a total of four classes are needed: one to match HTTP traffic from 10.1.1.1, one to match FTP traffic from 10.1.1.1, one to match HTTP traffic from 10.1.1.2, and one to match FTP traffic from 10.1.1.2. Example 5-12 shows the complete configuration of the classes. Notice that the classes have been renamed for clarity.

Example 5-12. Displaying the Class Configuration

```
R1#show class-map
 Class Map match-any class-default (id 0)
   Match any

 Class Map match-all ACCOUNTING-HTTP (id 2)
   Match access-group  101
   Match access-group  103

 Class Map match-all DEVELOPMENT-FTP (id 5)
   Match access-group  102
   Match access-group  104

 Class Map match-all DEVELOPMENT-HTTP (id 4)
   Match access-group  101
```

```
    Match access-group  104


 Class Map match-all ACCOUNTING-FTP (id 3)

    Match access-group  102

    Match access-group  103
```

Now all four of the traffic source/type pairs that were presented earlier in the chapter have been classified, allowing for differentiated treatment to be given to each.

Step 2: The Policy Map

The function of the class map is only to identify traffic, based on the characteristics given within the class map; the actual treatment of that traffic is specified in a policy map. As discussed earlier in the chapter, many different QoS mechanisms can be configured via the MQC, so the policy map has quite a few options. Note that, on switching platforms, not all of these options are supported in hardware. Switching platforms, such as the Catalyst 6500, may support some options in software. For performance reasons, you should not configure policies that are not supported in hardware unless absolutely necessary, because there could be a severe performance penalty for doing so. Cisco constantly adds new hardware support for features and the hardware support is different for different hardware (such as, PFC1 versus PFC2). As such, you should always check the hardware support for these actions before configuring them. NBAR was discussed earlier in this chapter and is a good example of something that was not supported in hardware on the 6500 at the time of this writing.

Configuring the Policy Map

Like the class map, the policy map is configured from the global configuration mode in Cisco IOS Software and requires a name.

```
R1(config)#policy-map ?

  WORD  policy-map name


R1(config)#policy-map ACCOUNTING-POLICY

R1(config-pmap)#
```

The policy map ACCOUNTING-POLICY is now configured, and the router automatically moves into policy map configuration mode, as indicated by the config-pmap in the router's prompt. This configuration mode allows for the configuration of the specific policy map that was created and has several options, as demonstrated in Example 5-13.

Example 5-13. Policy Map Options

```
R1(config-pmap)#?

QoS policy-map configuration commands:

  class        policy criteria

  description  Policy-Map description

  exit         Exit from QoS policy-map configuration mode

  no           Negate or set default values of a command

  rename       Rename this policy-map
```

As with the class map configuration, it is recommended that a description be configured for all policy maps, but this is not required. Again there is a rename option, which is just a convenient way of renaming the policy map without losing the configuration. Before the rename option was available, the only way to change the name of a policy map was to delete the policy map and then re-add the same configuration under the new name. For anyone who works with QoS on a daily basis, this was a great addition. The main purpose of the policy map is, however, to create the policy or policies that will be applied to traffic of a given class. As such, the option for selecting a class is of the most interest. For more complicated QoS configurations, using meaningful names for classes and policies along with the description feature is of great operational value.

The only option listed under the class command is for you to enter the name of the class for which you want to configure a policy. A list of available classes is not given, but the supported classes include those that have been configured manually plus class-default.

Example 5-14 shows the options available under class-default.

Example 5-14. Policy Map Options Available Under class-default

```
R1(config)#policy-map ACCOUNTING-POLICY

R1(config-pmap)#class class-default

R1config-pmap-c)#?

QoS policy-map class configuration commands:

  bandwidth       Bandwidth

  exit            Exit from QoS class action configuration mode

  fair-queue      Enable Flow-based Fair Queuing in this Class

  no              Negate or set default values of a command

  police          Police
```

```
priority        Strict Scheduling Priority for this Class

queue-limit     Queue Max Threshold for Tail Drop

random-detect   Enable Random Early Detection as drop policy

service-policy  Configure QoS Service Policy

shape           Traffic Shaping

set             Set QoS values
```

Notice that a new prompt appears, after a class has been selected, to indicate that you have entered the class configuration submode within the policy map configuration. The options shown, other than the fair-queue option, are the same for all classes. The fair-queue option is available on some classes, but not all. A discussion of the details of this behavior is beyond the scope of this text but can be found at Cisco.com.

Policy Map Options: A Closer Look

Because the policy map is the portion of the configuration that dictates the actual treatment of traffic by the router, a more detailed explanation of the behavior of each option is provided here. exit and no are not covered because their behavior is the same as elsewhere in Cisco IOS Software.

- bandwidth— Allows for the configuration of CBWFQ. The specifics of CBWFQ operation are beyond the scope of this explanation, but this command provides a minimum bandwidth guarantee to this class of traffic.
- fair-queue— Not available in all classes. This command enables Flow-based Weighted Fair Queuing within this class.
- police— Allows for the configuration of a policer, also known as *rate limiting*. The police command, when used within a class, is called class-based policing.
- priority— Designates that this class is a *Low Latency Queuing* (LLQ) class, which should receive strict scheduling priority to minimize delay, jitter and packet loss. Also specifies the amount of bandwidth for this class.
- queue-limit— Designates the maximum number of packets that can be in this queue.
- random-detect— Enables Weighted Random Early Detection (WRED) for congestion avoidance. By default, IP precedence is used for weight determination, but additional options within this command allow for the WRED algorithm to look at the DSCP. This command also provides an option for enabling *explicit congestion notification* (ECN) on this class.
- service-policy— Allows for the configuration of hierarchical policies (policy within a policy), which may be used to achieve functionality not possible in a single policy. For example, a T1 can be shaped to 512 kbps via a top-level policy, and then that 512 kbps can be divided (using CBWFQ/LLQ) within a second-level policy. Top-level policies are

sometimes called *parent policies*, and second-level policies are sometimes called *child policies*.

- shape— Allows for the configuration of class-based shaping, which is generic traffic shaping performed on a per-class basis. In this case, only the traffic in this class would be shaped. This is in contract to interface-based shaping, in which all traffic on the entire interface is shaped.

- set— Allows for the marking of packets. Several fields can be marked through the use of the set command, including IP precedence, IP DSCP, MPLS experimental bits, Layer 2 CoS, the ATM *cell loss priority* (CLP) bit, and the QoS group.

Back to the Accounting department example again, because it is time to configure the actual policies for the classes that have been defined. In this example, the only parameter that is used is bandwidth. This is not necessarily the policy that you would use in your production environment and is intended only as a sample of the configuration parameters.

Policy Map Configuration Example

Assuming that the link speed is 1.544 Mbps, and the intent is to give 128 kbps to each of the production classes, 64 kbps to the DEVELOPMENT-HTTP class, and 32 kbps to the DEVELOPMENT-FTP class, the configuration would look like Example 5-15.

Example 5-15. Configuring a Policy Map with Class Maps

```
R1(config-pmap)#policy-map ACCOUNTING-POLICY
R1(config-pmap)#class ACCOUNTING-HTTP
R1(config-pmap-c)#bandwidth 128
R1(config-pmap-c)#exit
R1(config-pmap)#class ACCOUNTING-FTP
R1(config-pmap-c)#bandwidth 128
R1(config-pmap-c)#exit
R1(config-pmap)#class DEVELOPMENT-HTTP
R1(config-pmap-c)#bandwidth 64
R1(config-pmap-c)#class DEVELOPMENT-FTP
R1(config-pmap-c)#bandwidth 32
```

Notice that between the bandwidth statement for DEVELOPMENT-HTTP and the class name for

DEVELOPMENT-FTP that there is no exit command. It is acceptable to exit from each class after configuring it, but not necessary. When you type a new class name, the MQC automatically moves to the configuration mode for that class.

You can confirm the configuration that has been entered through the use of the show policy-map command, as shown in Example 5-16.

Example 5-16. Verifying the Policy Map

```
R1#show policy-map

  Policy Map ACCOUNTING-POLICY

    Class ACCOUNTING-HTTP

            Bandwidth 128 (kbps) Max Threshold 64 (packets)

    Class ACCOUNTING-FTP

            Bandwidth 128 (kbps) Max Threshold 64 (packets)

    Class DEVELOPMENT-HTTP

            Bandwidth 64 (kbps) Max Threshold 64 (packets)

    Class DEVELOPMENT-FTP

            Bandwidth 32 (kbps) Max Threshold 64 (packets)

    Class class-default
```

As indicated by the output, the queue limit for each of these classes is set to 64 packets (the default). This output also shows the policy map's name, the names of all the class maps within the policy, and the policy that has actually been configured for each class. Notice that this command does *not* show any information about the traffic that will belong to each class or whether each class is a match-any or a match-all.

As discussed previously, one of the greatest benefits to the MQC structure is the lack of a learning curve for configuring new options. Suppose, for instance, that you now need to add traffic shaping to the ACCOUNTING-FTP class, to prevent the class from using more than 256 kbps under any conditions. If you were not using the MQC, it would be necessary to learn an entirely new command structure. Because the MQC uses the same structure to configure many QoS components, however, the configuration requires only one additional line of configuration.

Example 5-17 is the same example configuration as Example 5-15, but with the addition of traffic shaping.

Example 5-17. Traffic Shaping Within a Class Map

```
R1(config-pmap)#policy-map ACCOUNTING-POLICY
R1(config-pmap)#class ACCOUNTING-HTTP
R1(config-pmap-c)#bandwidth 128
R1(config-pmap-c)#exit
R1(config-pmap)#class ACCOUNTING-FTP
R1(config-pmap-c)#bandwidth 128
R1(config-pmap-c)# shape average 256000
R1(config-pmap-c)#exit
R1(config-pmap)#class DEVELOPMENT-HTTP
R1(config-pmap-c)#bandwidth 64
R1(config-pmap-c)#class DEVELOPMENT-FTP
R1(config-pmap-c)#bandwidth 32
```

The only additional command is the command to add traffic shaping, which makes the learning curve much shorter for learning how to configure a new QoS mechanism.

Further, the verification of the new configuration doesn't require anything new. As shown in Example 5-18, to verify both the CBWFQ configuration and the newly added traffic shaping configuration you can use the same command that was previously used to verify the CBWFQ configuration.

Example 5-18. Verifying the Policy Map Configuration

```
R1#show policy-map
  Policy Map ACCOUNTING-POLICY
    Class ACCOUNTING-HTTP
            Bandwidth 128 (kbps) Max Threshold 64 (packets)
    Class ACCOUNTING-FTP
            Bandwidth 128 (kbps) Max Threshold 64 (packets)
      Traffic Shaping
```

```
     Average Rate Traffic Shaping

             CIR 256000 (bps) Max. Buffers Limit 1000 (Packets)

  Class DEVELOPMENT-HTTP

          Bandwidth 64 (kbps) Max Threshold 64 (packets)

  Class DEVELOPMENT-FTP

          Bandwidth 32 (kbps) Max Threshold 64 (packets)

  Class class-default
```

After configuring both the class map, which defines the traffic that the router should pay attention to, and the policy map, which defines the action(s) that the router should apply to each class of traffic, the only thing that is left is to define the interface(s) to which the policy map should be applied.

Step 3: Attaching the Service Policy

All the configuration options and steps outlined so far in this chapter mean nothing until the resulting policy map is applied to an interface. This is much the same as configuring an ACL; the configuration means nothing until the ACL is applied to an interface.

For the service policy, only two options affect functionality: input and output. There is also a history option, not covered here in detail because it is for information only and does enable you to configure any functionality. Example 5-19 shows the two functional options.

Example 5-19. Service Policy Options

```
R1(config-if)#service-policy ?

  history  Keep history of QoS metrics

  input    Assign policy-map to the input of an interface

  output   Assign policy-map to the output of an interface
```

Theinput option means that the policy map is applied to traffic that enters the router through this interface, and the output option means that the policy map is applied to traffic that leaves the router through this interface. The ability to apply a policy map input or output on a specific interface depends on which QoS mechanisms are used in the policy map. Some actions are only allowed in output policies.

If the policy map called ACCOUNTING-POLICY is going to be applied to traffic leaving the router through Serial1/0, the commands in Example 5-20 are used.

Example 5-20. Applying the Service Policy

```
R1#conf t

Enter configuration commands, one per line.  End with CNTL/Z.

R1(config)#interface serial 6/0

R1(config-if)#service-policy output ACCOUNTING-POLICY
```

If you need to apply another policy in the input direction, you could do so easily, as follows:

```
R1(config-if)#service-policy input ANOTHER-POLICY
```

If you forget that you already have a service policy applied in the input direction and attempt to apply another policy, however, the router issues a friendly reminder that you are not allowed to apply more than one policy to an interface:

```
R1(config-if)#service-policy input THIRD-POLICY

 Policy map ANOTHER-POLICY is already attached
```

You now have the option of detaching the policy that was previously applied, which would then enable you to apply the THIRD-POLICY to this interface, or you can just do nothing and the original policy will remain attached.

Because the policy named ANOTHER-POLICY was only used to illustrate the ability to attach multiple policies (one input and one output), it is not included in the following show command outputs.

There is no service policy–specific command, but there is a command to show the interface-specific policy map information, and you should use that command to view the details of the policy map after it has been applied to an interface. Example 5-21 demonstrates output from the show policy-map interface ? command.

Example 5-21. Options for Verifying Policy Map Configuration

```
R1#show policy-map interface ?

  Async             Async interface

  BVI               Bridge-Group Virtual Interface
```

```
  CTunnel             CTunnel interface
  Dialer              Dialer interface
  Ethernet            IEEE 802.3
  Group-Async         Async Group interface
  Lex                 Lex interface
  Loopback            Loopback interface
  MFR                 Multilink Frame Relay bundle interface
  Multilink           Multilink-group interface
  Null                Null interface
  Serial              Serial
  Tunnel              Tunnel interface
  Vif                 PGM Multicast Host interface
  Virtual-Template    Virtual Template interface
  Virtual-TokenRing   Virtual TokenRing
  input               Input policy
  output              Output policy
  |                   Output modifiers
```

The interface types speak for themselves, but the input and output options are interesting because they can be used to show you all the policies that are applied in either the input or output direction (depending on which command you choose). More commonly, however, you will probably use the more specific command in Example 5-22.

Example 5-22. Verifying Policy Map Configuration for a Specific Interface

```
R1#show policy-map interface serial 1/0
 Serial1/0

  Service-policy output: ACCOUNTING-POLICY
```

```
    Class-map: ACCOUNTING-HTTP (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: access-group 101
      Match: access-group 103
      Queueing
Output Queue: Conversation 265
        Bandwidth 128 (kbps) Max Threshold 64 (packets)
        (pkts matched/bytes matched) 0/0
        (depth/total drops/no-buffer drops) 0/0/0

    Class-map: ACCOUNTING-FTP (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: access-group 102
      Match: access-group 103
      Queueing
        Output Queue: Conversation 266
        Bandwidth 128 (kbps) Max Threshold 64 (packets)
        (pkts matched/bytes matched) 0/0
        (depth/total drops/no-buffer drops) 0/0/0
      Traffic Shaping
           Target/Average   Byte   Sustain   Excess    Interval  Increment
             Rate           Limit  bits/int  bits/int  (ms)      (bytes)
           256000/256000    1984   7936      7936      31        992

        Adapt  Queue     Packets   Bytes     Packets   Bytes     Shaping
```

```
        Active Depth                         Delayed   Delayed   Active
        -      0         0         0         0         0         no

    Class-map: DEVELOPMENT-HTTP (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: access-group 101
      Match: access-group 104
Queueing
        Output Queue: Conversation 267
        Bandwidth 64 (kbps) Max Threshold 64 (packets)
        (pkts matched/bytes matched) 0/0
        (depth/total drops/no-buffer drops) 0/0/0

    Class-map: DEVELOPMENT-FTP (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: access-group 102
      Match: access-group 104
      Queueing
        Output Queue: Conversation 268
        Bandwidth 32 (kbps) Max Threshold 64 (packets)
        (pkts matched/bytes matched) 0/0
        (depth/total drops/no-buffer drops) 0/0/0

    Class-map: class-default (match-any)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
```

```
    Match: any
```

As you can see, the output is far more detailed than the output from the previous command. In fact, this command probably provides one of the most detailed outputs of any command in Cisco IOS. Certainly, this is the most comprehensive output of any MQC command.

Summary

This chapter examined the three-step process for configuring all the QoS mechanisms supported by the MQC. These steps are as follows:

Step 1. Class map configuration

Step 2. Policy map configuration

Step 3. Service policy application

Class maps come in two types:

- match-any— A logical OR operation, which means that only one of the match conditions must be met for a packet to belong to this class
- match-all— A logical AND operation, which means that all match criteria must be matched for a packet to belong to this class

match-all is the default, if no option is specified. Class maps are capable of matching certain criteria, but the majority of matches come via the matching of an access group. As detailed earlier in the chapter, you can combine match-all and match-any class maps with access group matching to accomplish fairly complex matching criteria.

Policy maps define many different actions that you can apply to a given class. Some actions may only be applied in a single direction, not both. Not all of these actions can be applied to packets entering an interface; instead, they are only allowed as an output policy (for packets leaving an interface). Traffic shaping is a good example of a policy map action that cannot currently be applied to inbound packets. Further, not all actions that are specified are supported in hardware on Catalyst products, which could mean a severe performance impact for certain features. Always check your Cisco IOS version and specific hardware type to determine hardware support for software features.

The next chapter introduces the platform-specific configuration options and specific hardware available for the Catalyst 2950 and 3550 products. The following chapter includes examples, using the MQC explained in this chapter, that show you how to configure QoS features on these products.

Chapter 6. QoS Features Available on the Catalyst 2950 and 3550 Family of Switches

This chapter continues the discussion of QoS feature support on Catalyst switches with the Catalyst 2950 Family and 3550 Family of switches. From an architectural perspective, the Catalyst 2950 Family of switches supports only Layer 2 MAC address forwarding and does not support Layer 3 routing. However, the Catalyst 2950 Family of switches supports a multitude of Layer 3 features such as classification based on CoS or DSCP, security and QoS *access-control list* (ACL) support, policing, and *Internet Group Management Protocol* (IGMP) snooping. Because the Catalyst 2950 Family of switches does not support IP routing, the typical network installations find the Catalyst 2950 Family of switches acting as access layer switches. Moreover, the Catalyst 3550 Family of switches supports IP routing and is applicable as an access layer or distribution layer switch. The Catalyst 3550 Family of switches may act as a core switch in small networks aggregating Catalyst 3550 switches and other access layer switches, such as the Catalyst 2950, 2900XL, and 3500XL Family of switches.

Both the Catalyst 2950 and 3550 Family of switches supports a wide range of QoS features that rival higher-end switches, including the Catalyst 4000 IOS Family of switches and the Catalyst 6500 Family of switches. Because the Catalyst 2950 and Catalyst 3550 switches use the same Cisco IOS code base, both support a base set of QoS features while the Catalyst 3550 Family of switches supports a few additional QoS features. Table 6-1, in the "Software Requirements" section of this chapter, outlines the QoS features supported by both the Catalyst 2950 and 3550 Family of switches along with those features only supported on the Catalyst 3550 Family of switches.

Specifically, this chapter discusses QoS support on the Catalyst 2950 and 3550 Family of switches, and includes the following topics:

- Architectural Overview
- Input Scheduling
- Classification and Marking
- Policing
- Congestion Management and Avoidance
- Auto-QoS
- Case Study
- Summary

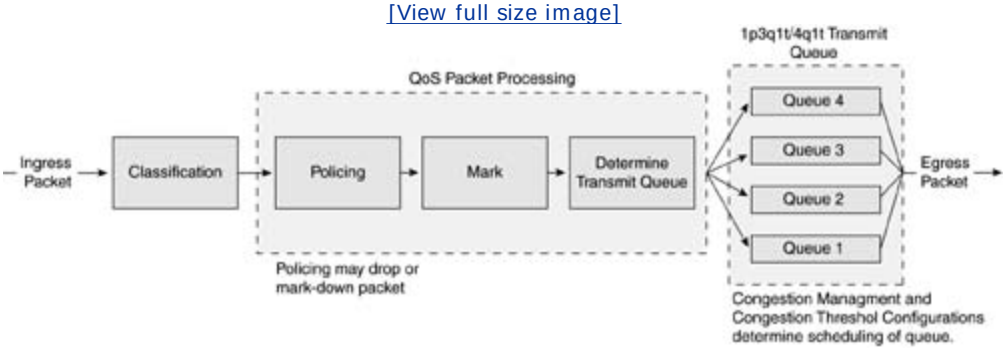
This chapter discusses both the Catalyst 2950 and Catalyst 3550 Family of switches concurrently because of the feature overlap of these switches. This chapter also indicates those features supported only on the Catalyst 3550 Family of switches on a per-feature basis.

Catalyst 2950 and Catalyst 3550 Family of Switches QoS Architectural Overview

The Catalyst 2950 and 3550 switches follow a standard model for QoS features. At the time of publication, all models of the Catalyst 2950 and 3550 Family of switches utilize the same QoS architecture. Specifically, the Catalyst 2950 and 3550 switches base QoS architecture on the following model.

Figure 6-1. Catalyst 2950 and 3550 QoS Architecture Model

[View full size image]



These switches do not modify the packet on ingress for classification and marking; instead, these switches assign an internal *differentiated services codepoint* (DSCP) value to the packet. These switches use the internal DSCP to make policing, queuing, and scheduling decisions. Furthermore, the internal DSCP determines the egress DSCP and *class of service* (CoS) values. The switches use the internal DSCP method of classification whether or not a packet contains an IP header. Policing may mark down the internal DSCP value for packets out-of-profile of a respective policer. As with the internal DSCP value, the switch carries the markdown DSCP value but does not modify the packet during rate policing.

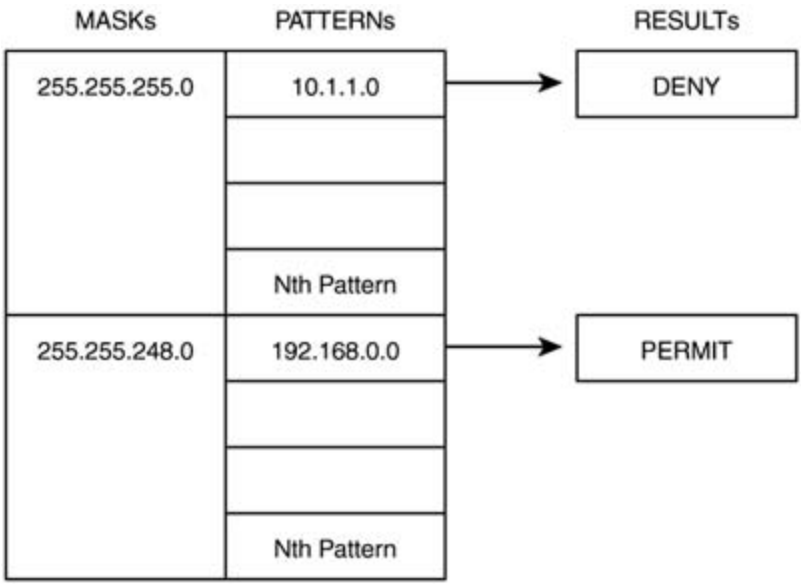
Both switches utilize *ternary content addressable memory* (TCAM) to carry out QoS features such as classification based on ACLs and policing. TCAM provides a high-rate packet-matching algorithm for determining result actions. Security ACLs and IP routing decisions also use TCAM for pattern matching. The use of TCAM is the method of choice for packet processing among current Catalyst switches.

In brief, switches populate TCAM with masks, values, and results. Values may represent IP addresses, protocol ports, DSCP values, or any other packet-matching field. Masks represent wildcard bits. Results represent packet actions such as Permit or Deny packets, as in the case of a security ACLs, or a pointer to more complex action such as marking the packet in the case of a classification ACL. The switch generates lookup keys for ingress packets in order to search TCAM tables for results. The switch compares lookup keys against masks and values. The first match provides a result. Catalyst switches utilize TCAM on packet ingress, during packet progressing, and on egress to gather results for actions such as Layer 3 forwarding and packet rewriting, QoS classification, policing, and security ACLs processing.

Figure 6-2 shows a logical example of a security ACL. The figure does not indicate exact

hardware implementation of TCAM, but rather a logical representation for understanding the use of TCAM. Actual hardware implementation of TCAM varies on each Catalyst Family of switch.

Figure 6-2. Logical Representation of TCAM



For additional technical information on the architectural use of TCAM, refer to the following technical document at Cisco.com:

> "Understanding ACL Merge Algorithms and ACL Hardware Resources on Cisco Catalyst 6500 Switches"

Exact implementations of TCAM vary significantly between Catalyst switching platforms. In addition, TCAM is a limited resource. The amount of ACLs that fit into TCAM are limited by the number of entries, range of ports specified, range of IP addresses, masks, and various other platform-specific constraints. These limitations are found in the configuration guides for each platform.

From a network design perspective, the QoS features supported on the Catalyst 2950 and 3550 Family of switches fit these switches into any end-to-end QoS design. Because the Catalyst 3550 Family of switches also supports IP routing features, the switch also fits well as a small distribution or core switch. Networks requiring more internal redundancy and additional port density should instead utilize the Catalyst 4500 or Catalyst 6500 Family of switches. Figure 6-3 shows a sample network design using Catalyst 2950 and Catalyst 3550 Family of switches in a small end-to-end design. Figure 6-4 illustrates the Catalyst 2950 and 3550 Family of switches in a campus network utilizing Catalyst 6500 switches in the core. The section, "Cisco IOS Software Feature Sets," discusses the EI and EMI acronyms found in Figures 6-3 and 6-4.

Figure 6-3. Sample Network Topology Using Catalyst 2950 and Catalyst 3550 Family of Switches

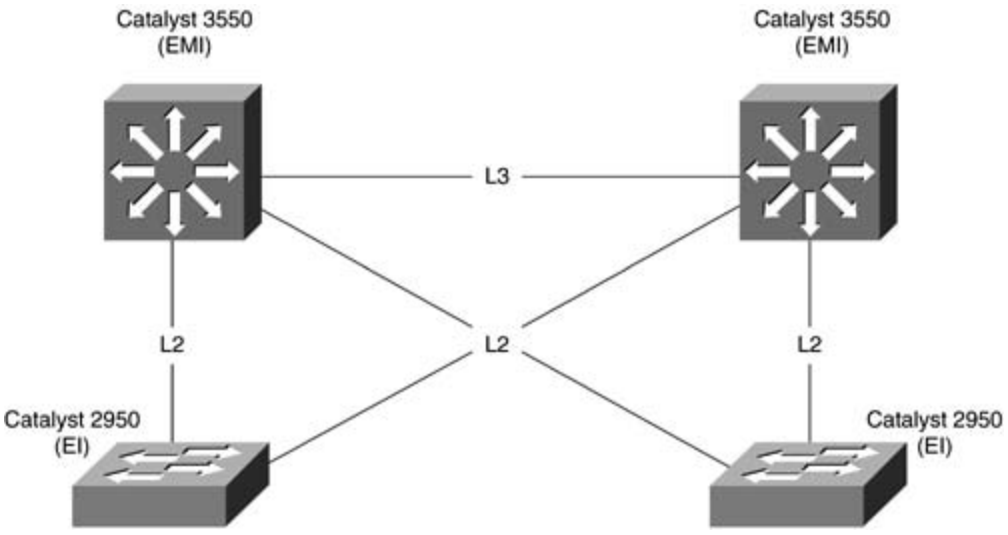
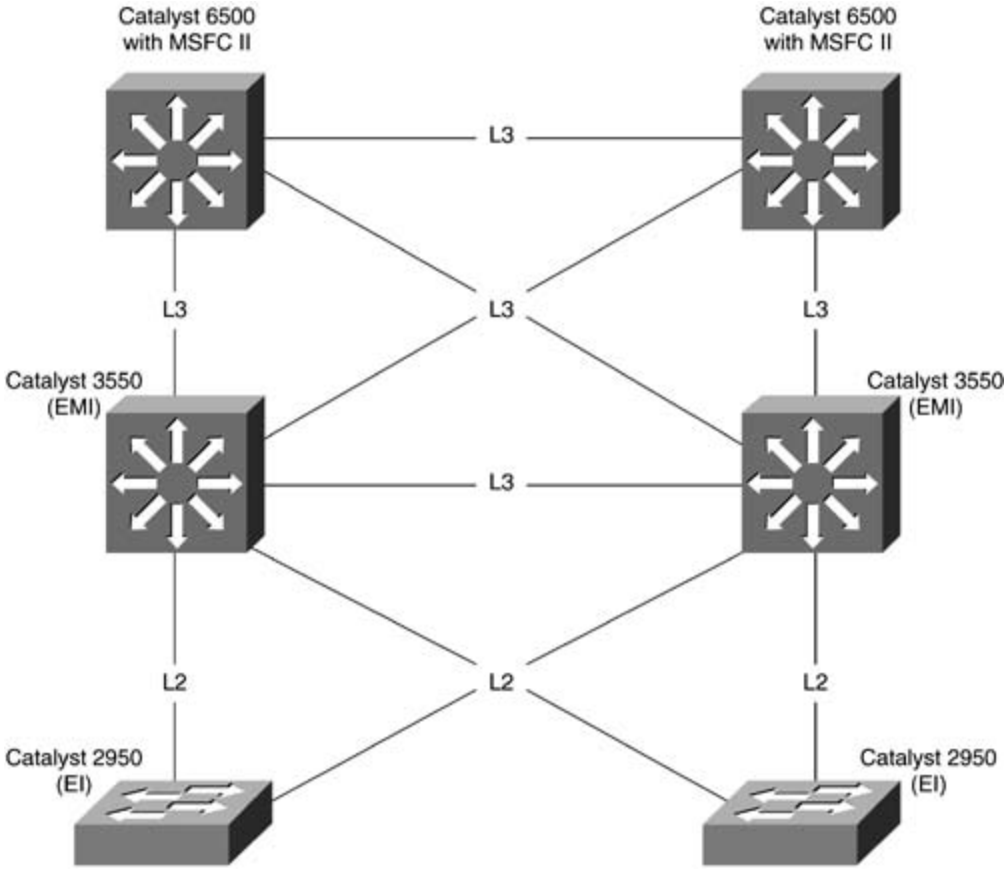


Figure 6-4. Sample Campus Network Topology Using Catalyst 2950 and Catalyst 3550 Family of Switches



Software Requirements

Both the Catalyst 2950 and Catalyst 3550 Family of switches utilize Cisco IOS Software. At the time of publication, these switches run the Cisco IOS 12.1 EA Software train.

These switches support many campus QoS features including policing, classification, marking, and scheduling of packets or frames. The Catalyst 2950 and 3550 switches share many common QoS features. Most of the common QoS features impart the same configuration and operational characteristics. One noteworthy difference in QoS support between the platforms is output scheduling. The Catalyst 2950 Family of switches supports only congestion management, whereas the Catalyst 3550 Family of switches supports both congestion management and congestion avoidance. The Catalyst 3550 Family of switches supports congestion management and avoidance using *weighted round-robin* (WRR) with configurable tail drop or *Weighted Random Early Detection* (WRED) thresholds. Both switches support strict-priority queuing. Furthermore, the Catalyst 3550 Family of switches supports several additional QoS features, including egress policing and several QoS mapping table configurations not supported on the Catalyst 2950 Family of switches. Table 6-1 outlines the supported QoS features of the Catalyst 2950 and 3550 Family of switches.

Table 6-1. Feature Parity Between the Catalyst 2950 and Catalyst 3550 Family of Switches in Cisco IOS 12.1(11)EA1

Feature	Platform Support
Classification	2950 (EI) 3550
Classification based on port trust state	2950 (EI) 3550
Classification based on ACLs	2950 (EI) 3550
Classification based on policy maps	2950 (EI) 3550
Classification based on trusting CoS	2950 (EI) 3550
Classification based on port CoS configuration	2950 (EI) 3550
Classification based on trusting DSCP	2950 (EI) 3550
Classification based on IP precedence	3550
Classification based on CDP from Cisco IP Phone	2950 3550
Marking	2950 (EI) 3550
Policing	2950 (EI) 3550
Ingress policing	2950 (EI) 3550
Egress policing	3550
Configuration of Mapping Tables	2950 (EI) 3550
CoS-to-DSCP	2950 (EI) 3550
IP precedence-to-DSCP	3550
DSCP-to-CoS	2950 (EI) 3550

Policed DSCP mapping table	3550
IP precedence-to-DSCP mapping table configuration	3550
DSCP-to-DSCP mapping	3550
DSCP-to-CoS	2950 (EI)
Output Queuing and Scheduling	2950 3550
Strict-priority scheduling	2950 3550
WRR scheduling	2950 3550
WRED congestion avoidance	3550
Tail-drop congestion avoidance	3550
Transmit queue size manipulation	3550
CDP = Cisco Discovery Protocol	

Cisco IOS Software Feature Sets

The Catalyst 2950 Family of switches runs two feature sets of Cisco IOS, a *standard image* (SI) and an *enhanced image* (EI). Several feature differences exist between the SI and EI versions. The SI version only supports 64 VLANs, for instance, whereas the EI version supports 256 VLANs with ACL support on physical interfaces. The SI only supports the output scheduling QoS features, whereas the EI adds support for classification, marking, and policing. Table 6-1 illustrates which QoS features are available in the SI and EI versions by noting EI for features that require the EI software version. Those switches that employ SI versions are upgradable to EI with specific restrictions. Refer to the product release notes for details.

The Catalyst 3550 Family of switches also runs two different feature sets of IOS, a *standard multilayer image* (SMI) and an *enhanced multilayer image* (EMI). With regard to IP routing, the SMI version only supports IP routing and a few routing features such as static routing and *Routing Information Protocol* (RIP). The EMI version supports other routing protocols and IP routing features including inter-VLAN routing, *Open Shortest Path First* (OSPF), *Interior Gateway Routing Protocol* (IGRP), *Extended Interior Gateway Routing Protocol* (EIGRP), *Border Gateway Protocol* (BGP), and the *Hot Standby Routing Protocol* (HSRP). Furthermore, the EMI version supports router ACLs and a higher number of multicast groups and MAC addresses per switch. Nevertheless, the QoS features are identical between the SMI and EMI versions. As a result, this chapter does not differentiate between the two versions. The Catalyst 3550 Family of switches is upgradable to the EMI version. Refer to the product release notes for details.

NOTE

In Cisco IOS 12.1(9)EA1, several software changes occurred that optimized the use of the TCAM space for QoS and security ACLs. As a result, both the Catalyst 2950 and 3550 Family of switches possibly support more ACL entries in Cisco IOS 12.1(9)EA than previous versions.

Global and Default Configuration

The Catalyst 2950 Family of switches does not require global configuration to enact on QoS configuration. The default trust behavior of the Catalyst 2950 is untrusted. The Catalyst 2950 Family of switches uses the default port CoS value of zero by default for the internal DSCP value. See the "Internal DSCP and Mapping Tables" section later in this chapter for more detailed information about internal DSCP.

The Catalyst 3550 Family of switches does support a global QoS configuration. By default, the QoS is disabled on the Catalyst 3550 Family of switches and, as a result, does not modify any CoS or DSCP value of a packet on ingress or egress. In addition, the switch does not carry out any congestion management or avoidance mechanisms. Use the following configuration command to globally enable QoS on a Catalyst 3550 switch:

```
[no]mls qos
```

Example 6-1 illustrates a user globally enabling QoS and verifying the configuration on a Catalyst 3550 switch.

Example 6-1. Enabling and Viewing QoS Global Configuration on a Catalyst 3550 Switch

```
Switch#config terminal

Switch(config)#mls qos

Switch(config)#end

Switch#show mls qos

QoS is enabled globally
```

The default trust behavior of the Catalyst 3550 Family of switches is untrusted when QoS is enabled. Furthermore, the switch carries out WRR scheduling with each queue receiving 25 percent of transmit bandwidth by default. The "Congestion Management" section of this chapter discusses WRR in more detail.

Input Scheduling

As with the other Catalyst platforms, neither the Catalyst 2950 nor the 3550 Family of switches support input scheduling, and they perform only *First-In, First-Out* (FIFO) Queuing of ingress packets. Not supporting input scheduling on any switch is not an issue when maintaining the supported packet-forwarding rate of the switch. Consult the Cisco product documentation for more information regarding the packet-forwarding rate of the Catalyst 2950 Family and Catalyst 3550 Family of switches.

Classification and Marking

Classification establishes an internal DSCP value, which the switches use to differentiate packets during packet processing, policing, and output scheduling and queuing. The Catalyst 2950 Family of switches supports the following classification options:

- Ingress port CoS configuration
- Trust CoS
- Trust DSCP
- Extended trust on voice and VLANs
- ACL-based classification
- Classified model based on internal DSCP
- CoS and DSCP mapping tables

The Catalyst 3550 Family of switches supports the following classification options:

- Ingress port CoS configuration
- Trust CoS
- Trust DSCP
- Trust IP precedence
- Extended trust on voice and VLANs
- Trust Cisco IP Phone
- ACL-based classification
- Classification model based on internal DSCP
- CoS and DSCP mapping tables

Both the trust CoS and trust DSCP options allow for DSCP and CoS passthrough, respectively. In addition, the trust DSCP option on the Catalyst 3550 Family of switches supports DSCP mutation.

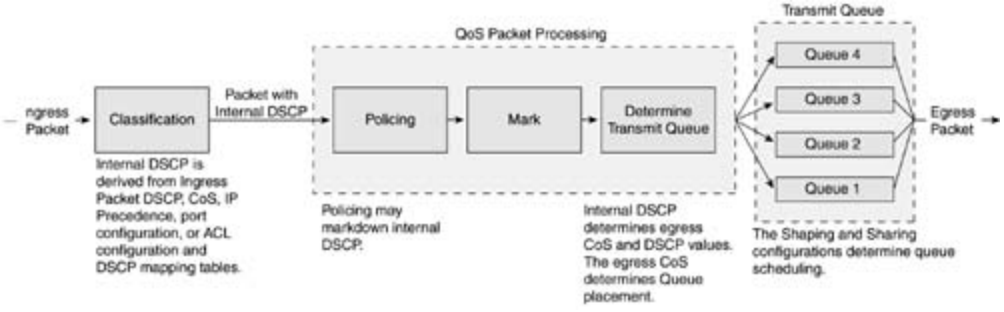
Internal DSCP and Mapping Tables

As mentioned in the architecture overview section, the Catalyst 2950 Family and 3550 Family of switches utilize an internal DSCP value to represent classification and marking of frames as the frames traverse the switch. Because the switches use internal DSCP values, the switch maps ingress DSCP, CoS, and IP precedence to the internal DSCP values when dictated by the classification configuration. When trusting DSCP, the switch maps ingress packets' DSCP directly to an internal DSCP value. The only exception is in configuring DSCP mutation. Figure 6-5

illustrates the logical depiction of internal DSCP as a packet traverses a Catalyst 2950 or 3550 switch.

Figure 6-5. Logical Depiction of Internal DSCP

[View full size image]



For CoS and IP precedence mapping to an internal DSCP, the switches use a CoS-to-DSCP and an IP precedence-to-DSCP mapping table, respectively. Tables 6-2 and 6-3 indicate the default mappings for the CoS-to-DSCP and IP precedence-to-DSCP mappings, respectively.

Table 6-2. Default CoS-to-DSCP Mapping Table

CoS	0	1	2	3	4	5	6	7
DSCP	0	8	16	24	32	40	48	56

Table 6-3. Default IP Precedence-to-DSCP Mapping Table

IP Precedence	0	1	2	3	4	5	6	7
DSCP	0	8	16	24	32	40	48	56

The IP precedence-to-DSCP mapping table is configurable on the Catalyst 3550 Family of switches, whereas both the Catalyst 2950 Family and the Catalyst 3550 Family of switches support custom configuration of the CoS-to-DSCP mapping tables. To configure the CoS-to-DSCP and IP precedence-to-DSCP mapping tables, use the following commands, respectively:

```
mls qos map cos-dscpDSCP-list

mls qos map IP-prec-dscpDSCP-list
```

DSCP-list represents the eight DSCP mapping values for each IP precedence value 0 to 7. For example, a *DSCP-list* of 0 10 16 25 36 46 48 50 for a CoS-DSCP mapping configuration results in the CoS value of 0 mapping to a DSCP value 0, CoS value of 1 mapping to a DSCP value of 10, and so forth. Examples 6-2 and 6-3 illustrate a user configuring and verifying configuration of a CoS-to-DSCP mapping table and a IP precedence-to-DSCP mapping table, respectively.

Example 6-2. User Configuring and Verifying CoS-to-DSCP Mapping Table

```
Switch#config terminal

Enter configuration commands, one per line. End with CNTL/Z.

Switch(config)#mls qos map cos-dscp 0 8 16 30 30 46 50 50

Switch(config)#end

Switch#show mls qos map cos-dscp

   Cos-dscp map:

        cos:   0  1  2  3  4  5  6  7

     --------------------------------

       dscp:   0  8 16 30 30 46 50 50
```

Example 6-3. User Configuring and Verifying IP Precedence-to-DSCP Mapping Table

```
3550#config terminal

Enter configuration commands, one per line. End with CNTL/Z.

3550(config)#mls qos map ip-prec-dscp 0 8 16 28 30 46 50 55

3550(config)#end
```

```
3550#show mls qos map ip-prec-dscp

   IpPrecedence-dscp map:

     ipprec:   0  1  2  3  4  5  6  7

     --------------------------------

       dscp:   0  8 16 28 30 46 50 55
```

Ingress Port CoS Configuration

The Catalyst 2950 Family and 3550 Family of switches allow for static configuration of ingress CoS values used for DSCP mapping. The switches use ingress port CoS configurations to classify untagged frames and override CoS values on tagged frames.

To configure a switch interface to represent a CoS value for an untagged frame, use the following interface commands:

```
mls qos trust cos

mls qos cos cos_value
```

cos_value represents the CoS value to assign to untagged frames. The switch requires both configuration commands for assigning CoS values to untagged frames.

Example 6-4 illustrates an example of an interface configured for trusting CoS for tagged frames and the default configuration of assigning a CoS value of 0 to untagged frames.

Example 6-4. Example Interface Configuration of Trusting CoS Values of Tagged Frames and Assigning Default CoS Value 0 to Untagged Frames

```
Switch#show running-config

Building configuration...
```

```
!
(text deleted)
interface FastEthernet0/1
 switchport access vlan 53
 switchport voice vlan 701
 no ip address
 mls qos trust cos
 spanning-tree portfast
(text deleted)
end
```

Example 6-5 illustrates an interface configured for trusting CoS for tagged frames and assigning a CoS value of 5 to untagged frames.

Example 6-5. Sample Interface Configuration of Trusting CoS Values of Tagged Frames and Assigning a Port CoS Value 5 to Untagged Frames

```
Switch#show running-config
Building configuration...
!
(text deleted)
interface FastEthernet0/1
 switchport access vlan 53
 switchport voice vlan 700
 no ip address
 mls qos cos 5
 mls qos trust cos
 spanning-tree portfast
(text deleted)
```

```
end
```

To configure an overriding CoS value on untagged and tagged frames on a switch interface, use the following commands:

```
mls qos trust cos

mls qos cos cos_value

mls qos cos override
```

The mls qos cos override signifies to override tags frames with the CoS value specified in the mls qos cos *cos_value* command. Example 6-6 shows an interface configuration using the override feature.

Example 6-6. Sample Interface Configuration of Trusting and Assigning a Port CoS Value 5 to Untagged and Tagged Frames

```
Switch#show running-config

Building configuration...

!

(text deleted)

interface FastEthernet0/1

 switchport access vlan 53

 switchport voice vlan 700

 no ip address

 mls qos cos 5

 mls qos trust cos

 mls qos cos override
```

```
 spanning-tree portfast

(text deleted)

end
```

Assigning a port CoS value to an ingress frame does not necessarily result in the switch transmitting the frame with that CoS value. The port CoS value only determines the internal DSCP value of the ingress frame, and the switch may mark the frame, mark down the frame during policing, or use a nondefault DSCP-to-CoS mapping table before egress transmission of the frame.

To verify any classification configuration of an interface, use the following command:

show mls qos interface*[interface-name]*

Example 6-7 illustrates sample output from this command on a Catalyst 3550 switch.

Example 6-7. Sample Output from the show mls qos interface Command

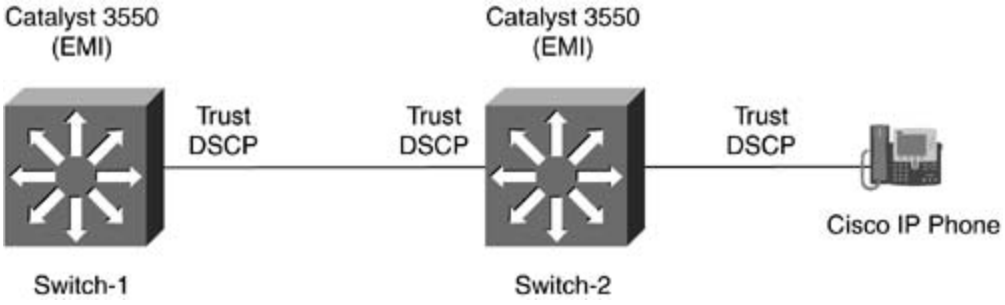
```
Switch#show mls qos interface GigabitEthernet 0/1

GigabitEthernet0/1

trust state: trust dscp

trust mode: trust dscp

COS override: dis

default COS: 0

DSCP Mutation Map: Default DSCP Mutation Map

trust device: none
```

Trust DSCP

The trust DSCP configuration maps an ingress packet's DSCP value directly to the internal DSCP value. Trust DSCP is the recommended trust configuration for directly attached switches that perform classification and marking on ingress. Because classification and marking occurs on ingress for interconnected switch ports, just trusting DSCP allows the current switch to maintain the original ingress classification and marking of the connected switch. Trusting DSCP is also a choice configuration for interfaces connected to Cisco IP Phones. Cisco IP Phones mark frames with DSCP and CoS values, and just trusting DSCP on frames from the Cisco IP Phone is a common practice.

Figure 6-6 illustrates an example of this behavior. Switch-2 trusts DSCP on the attached Cisco IP Phone. Switch-1 preserves the classification of Switch-2 by just trusting DSCP on the interface connecting the two switches. As a result, Switch-1 preserves the classification done by Switch-2.

Figure 6-6. Trust DSCP Example



To configure a switch interface to trust DSCP, use the following interface command:

```
mls qos trust dscp
```

Example 6-8 illustrates an interface configured for trusting DSCP on ingress packets.

Example 6-8. Sample Interface Configuration of Classifying Frames Based on DSCP Values

```
Switch#show running-config
Building configuration. . .
!
(text deleted)
interface FastEthernet0/1
 switchport access vlan 53
 switchport voice vlan 700
 no ip address
 mls qos trust dscp
 spanning-tree portfast
(text deleted)
end
```

Trust IP Precedence

The trust IP precedence configuration mirrors trusting CoS. Trusting IP precedence is a feature only available on the Catalyst 3550 Family of switches. Trusting IP precedence uses the IP precedence mapping table to map an ingress packet's IP precedence value to an internal DSCP value. Because some applications only mark IP precedence, trusting IP precedence rather than DSCP allows for ease of understanding and configuration. Table 6-3, in the "Internal DSCP and Mapping Tables" section, shows the default mapping table for mapping IP precedence to internal DSCP.Example 6-3, in the same section, illustrates a user modifying the IP precedence to internal DSCP mapping table.

To configure a switch interface to assign a trust DSCP, use the following interface command:

```
mls qos trust ip-precedence
```

Example 6-9 shows a sample configuration of a switch interface configured for trusting IP precedence.

Example 6-9. Sample Interface Configuration of Classifying Frames Based on IP Precedence Values

```
Switch#show running-config

Building configuration...

!

(text deleted)

interface FastEthernet0/1

 switchport access vlan 53

 switchport voice vlan 700

 no ip address

 mls qos trust ip-precedence

 spanning-tree portfast

(text deleted)

end
```

Voice VLANs and Extended Trust

Both the Catalyst 2950 Family and 3550 Family of switches support voice VLANs and extended trust options. The "Voice VLANs and Extended Trust" section of Chapter 2, "End-to-End QoS: Quality of Service at Layer 3 and Layer 2," discusses the concept and configurations of voice VLANs and extended trust.

Trust Cisco IP Phone Device

The trust CoS configuration trusts the CoS values of all ingress frames on tagged frames. The trust DSCP configuration trusts the DSCP values of all ingress frames. In the case of an attached Cisco IP Phone, the desired configuration is to trust CoS or trust DSCP depending on desired configuration.

In most configurations, workstations attach directly to Cisco IP Phones, which, in turn, are attached to switches as shown in Figure 6-7.

Figure 6-7. Cisco IP Phone Physical Network Diagram

Because the trust CoS or trust DSCP configuration does not validate the ingress frames, it is possible to send frames with specific DSCP values for malicious use on interfaces configured for trusting CoS or DSCP. As a result, the Catalyst 2950 Family and 3550 Family of switches support trusting CoS only when the switch connects to a Cisco IP Phone. In addition, the Catalyst 3550 Family of switches also supports trusting DSCP only when the switch connects to a Cisco IP Phone. These switches achieve this level of security by using the CDP. Because all Cisco IP Phones send CDP periodically and on linkup by default, the switch learns of connected Cisco IP Phones dynamically. When using this configuration option with trusting enabled, these switches only trust ingress frames when a Cisco IP Phone is attached. If the switch does not detect CDP packets from a Cisco IP Phone using this configuration, the switches use the port CoS configuration for determining CoS values associated with ingress frames. Because CDP is a proprietary protocol, only Cisco IP Phones support CDP.

To configure a switch interface to trust CoS only when a Cisco IP Phone is attached, use the following interface commands:

```
mls qos trust cos

mls qos trust device cisco-phone
```

To configure a switch interface to trust DSCP only when a Cisco IP Phone is attached, use the following interface commands:

```
mls qos trust dscp

mls qos trust device cisco-phone
```

Example 6-10 illustrates a sample configuration of an interface on a Catalyst 3550 configured for trusting DSCP when a Cisco IP Phone is connected to interface FastEthernet 0/1.

Example 6-10. Sample Interface Configuration of Classifying Frames Based on DSCP and Whether an IP Phone Is Connected to an Interface

```
Switch#show running-config

Current configuration : 157 bytes

!

(text deleted)

interface FastEthernet0/1

 switchport access vlan 53

 switchport voice vlan 700

 no ip address

 mls qos trust device cisco-phone

 mls qos trust dscp

 spanning-tree portfast

(text deleted)

end
```

Classifying Traffic by Using ACLs

The Catalyst 2950 Family and 3550 Family of switches support standard and extended IP ACLs and MAC ACLs for security and QoS purposes. For QoS purposes, these switches utilize ACLs in class maps for classifying packets. Using ACLs for classification allow for granularity when classifying packets. By using ACLs for classification, for example, the switch can classify packets that match only specific IP addresses or Layer 4 ports.

These switches use class maps to organize and group multiple ACLs for application to policy maps. Policy maps group class maps and class actions such as trusting, marking, and policing.

Chapter 5, "Introduction to the Modular QoS Command-Line Interface," expands on how to create and implement class maps and policy maps and includes examples. Consult Chapter 5 before reading the configuration examples and guidelines in the "Class Maps and Policy Maps" section later in this chapter.

To apply a policy map to an interface for ingress-supported classification, marking, or rate policing, use the following command:

service-policy input*policy-map-name*

policy-map-name refers to the name you configure for the policy map.

To apply a policy map to an interface for egress-supported classification, marking, or rate policing, use the following command:

service-policy output*policy-map-name*

policy-map-name refers to the name you configure for the policy map.

Example 6-11 illustrates a policy map, class map, and interface configuration for classifying traffic based on an ACL.

Example 6-11. Sample Configuration for Classifying Traffic Based on an ACL

```
Switch#show running-config

Building configuration...

!
```

```
mls qos
!
class-map match-all MATCH_ACL_100
  match access-group 100
!
!
policy-map Classify_ACL
  class MATCH_ACL_100
    trust dscp
!
!
(text deleted)
!
interface FastEthernet0/1
 switchport access vlan 2
 switchport voice vlan 700
 no ip address
 duplex full
 speed 100
 service-policy input Classify_ACL
 spanning-tree portfast
!
(text deleted)
!
access-list 100 permit ip 10.1.1.0 0.0.0.255 10.2.1.0 0.0.0.255
!
(text deleted)
end
```

Classification Passthrough Option

The classification passthrough option forces the switch to treat CoS and DSCP independently. The default behavior, and the lone behavior for all software versions prior to Cisco IOS versions 12.1.11(EA)1, of the switch is to modify the CoS or DSCP value depending on internal trust DSCP and any mapping tables. This is the behavior described in the previous sections of this chapter.

To recap earlier chapters, an interface configured for trusting DSCP modifies the CoS value on the frame on egress based on the internal DSCP and DSCP-to-CoS mapping table. Conversely, an interface configured for trusting CoS modifies the internal DSCP value on ingress according to the CoS-to-DSCP mapping table. The internal DSCP values determine the egress DSCP value.

To force the switch to treat CoS and DSCP independently, use the classification passthrough option. For trusting CoS configurations, the classification DSCP passthrough option uses the ingress CoS value of a frame for policing and scheduling without modifying the DSCP value of the frame on egress. Both the Catalyst 2950 Family and 3550 Family of switches support the trust CoS passthrough DSCP option.

The Catalyst 3550 Family of switches also supports the opposite behavior, trust DSCP passthrough CoS. Trust DSCP passthrough CoS enables the interface to classify, mark, police, and schedule packets without modifying the egress CoS value.

In practice, network administrators occasionally use these features to preserve CoS and DSCP values of frames when packets cross multivendor and multiplatform networks or ISP networks.

To configure an interface to trust CoS and passthrough DSCP, use the following interface command:

```
mls qos trust cos pass-through dscp
```

To configure an interface on a Catalyst 3550 Family of switches to trust DSCP and passthrough CoS, use the following interface command:

```
mls qos trust dscp pass-through cos
```

Example 6-12 illustrates a sample interface configuration of trusting CoS while passing through DSCP values.

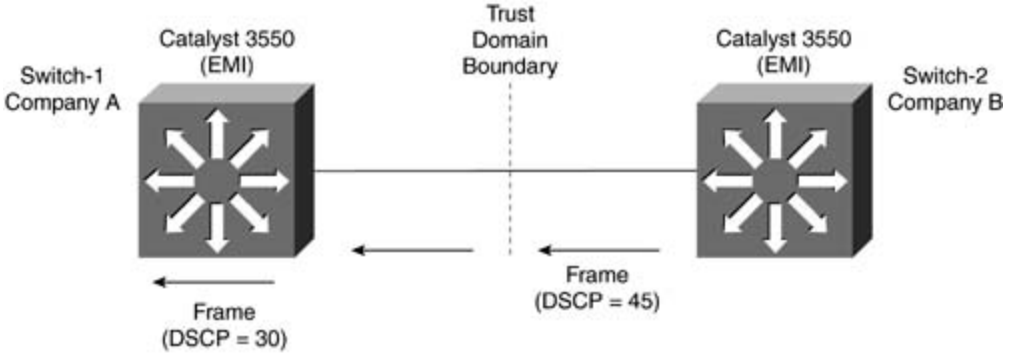
Example 6-12. Sample Interface Configuration of Trusting CoS and Passthrough DSCP

```
Switch#show running-config
Building configuration...
!
mls qos
!
(text deleted)
!
interface FastEthernet0/1
 switchport access vlan 2
 switchport voice vlan 700
 no ip address
 duplex full
 speed 100
 mls qos trust cos pass-through dscp
 spanning-tree portfast
!
(text deleted)
!
end
```

Ingress DSCP Mutation

Classification using ingress DSCP mutation enables the switch to map ingress DSCP values to alternate configured DSCP values. In essence, DSCP mutation is the logical equivalent to a DSCP-to-DSCP mapping. Ingress DSCP mutation is commonly used when connecting multiple autonomous QoS domains, as illustrated in Figure 6-8. This feature is only available on the Catalyst 3550 Family of switches.

Figure 6-8. DSCP Mutation



In this example, Switch 1 remaps packets with ingress DSCP values of 45 to 30 by way of the internal DSCP value. Example 6-13 provides a sample configuration of ingress DSCP mutation illustrating this behavior.

With regard to the DSCP mutation configuration, each Gigabit Ethernet interface supports a single DSCP mutation map. However, each group of 12 Fast Ethernet interfaces supports a single DSCP mutation map.

To configure the DSCP mutation map, use the following configuration command:

mls qos map dscp-mutation*dscp-mutation-name in-dscp***to***out-dscp*

dscp-mutation-name indicates the name assigned to the globally configured DSCP mutation map.*in-dscp* represents up to eight ingress DSCP vales for mapping to the single *out-dscp* value.

To configure and assign an interface for ingress DSCP mutation, use the following interface commands:

```
mls qos trust dscp

mls qos dscp-mutation dscp-mutation-name
```

dscp-mutation-name is the DSCP mutation global map name. The switch requires the trust DSCP configuration to enact on the DSCP mutation configuration.

Example 6-13 illustrates a sample configuration of ingress DSCP mutation for Figure 6-8.

Example 6-13. Sample Configuration for Ingress DSCP Mutation Applied to Figure 6-8

```
Switch#show running-config

Building configuration...

!

mls qos map dscp-mutation 40-45_reduce_30 40 41 42 43 44 45 to 30

mls qos

!

(text deleted)

!

interface FastEthernet0/1

 switchport access vlan 2

 switchport voice vlan 700

 no ip address

 duplex full

 speed 100

 mls qos trust dscp
```

```
 mls qos dscp-mutation 40-45_reduce_30
 spanning-tree portfast
!
(text deleted)
!
end
```

To verify the DSCP mutation map configurations, use the following command:

show mls qos maps dscp-mutation

Example 6-14 illustrates sample output for the preceding command for Example 6-13.

Example 6-14. Displaying QoS DSCP Mutation Mappings

```
Switch#show mls qos maps dscp-mutation
   Dscp-dscp mutation map:
   40-45_reduce_30:
     d1 :  d2 0  1  2  3  4  5  6  7  8  9
     ---------------------------------------
      0 :    00 01 02 03 04 05 06 07 08 09
      1 :    10 11 12 13 14 15 16 17 18 19
      2 :    20 21 22 23 24 25 26 27 28 29
      3 :    30 31 32 33 34 35 36 37 38 39
```

```
   4 :    30 30 30 30 30 30 46 47 48 49

   5 :    50 51 52 53 54 55 56 57 58 59

   6 :    60 61 62 63

Dscp-dscp mutation map:

Default DSCP Mutation Map:

  d1 :  d2 0  1  2  3  4  5  6  7  8  9

  ---------------------------------------

   0 :    00 01 02 03 04 05 06 07 08 09

   1 :    10 11 12 13 14 15 16 17 18 19

   2 :    20 21 22 23 24 25 26 27 28 29

   3 :    30 31 32 33 34 35 36 37 38 39

   4 :    40 41 42 43 44 45 46 47 48 49

   5 :    50 51 52 53 54 55 56 57 58 59

   6 :    60 61 62 63
```

Policing

Both The Catalyst 2950 Family and 3550 Family of switches support policing with trusting, marking DSCP markdown actions. Both switches use the modular *command-line interface* (CLI) as discussed Chapter 5 for configuring class maps and policy maps used for policing.

The Catalyst 2950 Family of switches supports only ingress policing on a per-port basis, whereas th Catalyst 3550 Family of switches supports per-port and aggregate policing in both ingress and egre configurations.Table 6-4 summarizes the policing options available to each switch. This section disc each of the following policing topics as it relates to the Catalyst 2950 and Catalyst 3550 Family of switches:

- Policing Resources and Guidelines
- Class Maps and Policy Maps
- Ingress and Egress Policing
- Individual and Aggregate Policing
- Port-Based, VLAN-Based, and Per-Port Per-VLAN-Based Policing
- Policing Actions

Table 6-4. QoS Policing Feature Available Per Platform

QoS Policing Feature	Catalyst 2950EI	Catalyst 3550
Port-based policing	Supported on all interfaces	Supported on all physic interfaces
Aggregate port-based policing	Not supported	Supported on all physic interfaces
Per-port VLAN-based policing	Not supported	Supported on ingress or
Policing rate parameters	Rate and burst only	Rate and burst only
No. of ingress policers per Gigabit Ethernet interfaces	60	128
No. of ingress policers per Fast Ethernet interfaces	6	8
No. of egress policers per Gigabit Ethernet interfaces	Not supported	8
No. of egress policers per Fast Ethernet interfaces	Not supported	8
Egress policers	Not supported	Supported

Policing Resources and Guidelines

Table 6-4 summarizes the QoS policing resource restrictions and guidelines for software version 12.1(11)EA1. As indicated in Table 6-4, the Catalyst 3550 Family of switches provides for additiona features over the Catalyst 2950 Family of switches. These features include per-port VLAN-based po and egress policing. Furthermore, the Catalyst 3550 Family of switches supports additional policing resources that provide for a larger number of policers.

Class Maps and Policy Maps

As with mainline Cisco IOS Software, class maps group ACLs and match statements for application policy maps. The class maps in effect define the classification criteria for policy maps that define ac Example 6-15 illustrates a sample configuration of a policy map.

Example 6-15. Sample Class Map and Policy Map Configuration

```
Switch#show running-config
Building configuration...
!
(text deleted)
mls qos
!
class-map match-any MATCH_LIST
  match access-group 100
  match ip precedence 5
  match ip dscp 35
!
!
policy-map RATE_MARK
  class MATCH_LIST
    police 1000000 8000 exceed-action drop
    set ip dscp 55
!
```

```
(text deleted)

interface GigabitEthernet0/1

 switchport trunk encapsulation dot1q

 switchport mode trunk

 no ip address

 service-policy input RATE_MARK

(text deleted)

access-list 100 permit ip 10.1.1.0 0.0.0.255 10.2.1.0 0.0.0.255

(text deleted)

!

end
```

The class map MATCH_LIST defines a class map with several classification matching rules. Becaus class map uses the match-any option, matching any of the three match statements results in polic executing the class actions. The other matching option, match-all, configures the switch to subject packet to all the match statements in order to enact on the policy map class actions.

Furthermore, the class map defines three matching rules. For the switch to execute the class actior policy map, a packet must match ACL 100, have an IP precedence value of 5, or have an IP DSCP v 35. Otherwise, the switch does not execute the class actions for the packet. Because the switch app policy map on ingress, the switch performs the matching operation on all ingress frames on GigabitEthernet0/1.

For packets that match the classification rules in the class map, the switch executes the class actio defined in the policy map. In Example 6-15, the switch rate limits this traffic by dropping frames al defined rate of 1.0 Mbps and sets the internal DSCP value to 55. Chapter 5 provides additional configuration information on class maps and policy maps. The "Traffic-Rate Policing" section later i chapter discusses the rate policer in Example 6-15.

Ingress and Egress Policing

Ingress policing logically refers to applying a set of class actions such as trusting, marking, or rate to specific packets as the switch receives packets inbound. Ingress policing logically occurs when a receives a packet, but actually occurs later in packet processing on Catalyst switches. Nevertheless logical concept of ingress policing is applying class actions to received packets. Egress policing app set of class actions on transmit; however, this feature is not found on all Catalyst switches. At the t publication, only the Catalyst 3550 Family and 4000 IOS Family of switches support egress policing the Catalyst 3550 Family of switches supports only traffic-rate policers for egress policing.

Applying a policy on ingress versus egress may result in significant behavioral differences in packet InFigure 6-9, for example, an ingress policer is rate limiting the traffic to 100 Mbps ingress on inte GigabitEthernet0/1. As shown in the diagram, ingress traffic from Switch-1 is at 1.0 Gbps. Based o

rate limiting, the switch restricts ingress traffic collectively to 100 Mbps. Assuming all ingress traffi Switch-2 is unicast, only 100 Mbps of total traffic is sent out interfaces GigabitEthernet0/2 and 0/3 Figure 6-10, the switch applies the same policer outbound, and the switch only limits the traffic transmitted out interfaces Gigabit Ethernet0/2 and 0/3 to 100 Mbps individually. Traffic from interf GigabitEthernet0/4 to other interfaces flows without any restriction on rate.

Figure 6-9. Ingress Policing

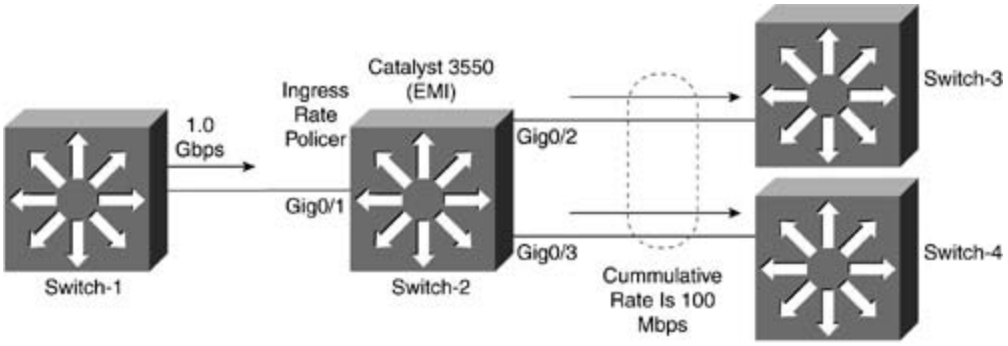
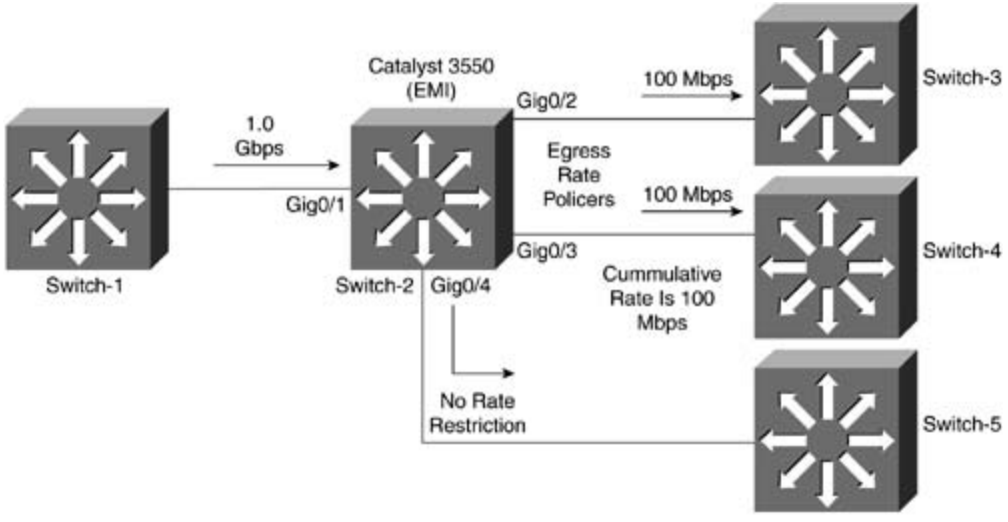


Figure 6-10. Egress Policing



Individual and Aggregate Policing

Individual policers apply bandwidth limits discretely to each interface for defined policy maps. The 2950 Family of switches supports only individual policers; however, the Catalyst 3550 Family of sw supports aggregate policers.

Class maps define traffic classes within a policy map. Use the following policy map class clause

configuration command to configure individual policers:

```
police rate burst [exceed-action {transmit | drop | policed-DSCP-transmit}]
```

Example 6-16 illustrates a sample configuration for individual policing. The policy map in Example drops packets exceeding the 100-Mbps rate defined in the policer for all packets ingress on interfac GigabitEthernet0/1.

Example 6-16. Sample Configuration of Individual Policer

```
Switch#show running-config
Building configuration...
!
(text deleted)
mls qos
!
class-map match-all MATCH_ALL_PCKTS
  match any
!
!
policy-map RATE_RESTRICT
  class MATCH_ALL_PCKTS
    police 100000000 16000 exceed-action drop
!
(text deleted)
!
```

```
interface GigabitEthernet0/1
 switchport trunk encapsulation dot1q
 switchport mode trunk
 no ip address
 service-policy input RATE_MARK
!
```

Aggregate policers apply rate-limiting constraints collectively among multiple class maps within th policy map. This behavior is unique compared to other switch platforms such as the Catalyst 4000 Family and Catalyst 6000 Family of switches. These switches use aggregate policers to apply rate-l constraints among multiple ports or VLANs.

Use the following global configuration command to configure an aggregate policer:

mls qos aggregate-police*aggregate-policer-name rate-bps burst-byte***exceed-action {d**

➥oliced-dscp-transmit}

aggregate_policer_name defines the name to represent the aggregate policer. Aggregate policers s dropping or marking down packets that exceed the define rate. Later sections in this chapter cover and burst parameters in more detail. To attach the aggregate policer, use the following policy map clause configuration command:

police aggregate*policer_name*

Aggregate policers cannot be used across different policy maps or interfaces. Define multiple aggre

policers to work around this limitation. Example 6-17 illustrates an aggregate policer.

Example 6-17. Sample Configuration of an Aggregate Policer

```
Switch#show running-config
Building configuration...
!
(text deleted)
mls qos aggregate-policer RATE_500MBPS 500000000 64000 exceed-action drop
mls qos
!
class-map match-all MATCH_ACL_100
  match access-group 100
class-map match-all MATCH_ACL_101
  match access-group 101
!
policy-map AGGR_TRAFFIC_LIMITING
  class MATCH_ACL_100
    set ip precedence 0
    police aggregate RATE_500MBPS
  class MATCH_ACL_101
    set ip precedence 5
    police aggregate RATE_500MBPS
!
(text deleted)
!
interface GigabitEthernet0/1
 switchport trunk encapsulation dot1q
 switchport mode trunk
```

```
 no ip address
 service-policy input AGGR_TRAFFIC_LIMITING
!
access-list 101 permit ip any any
access-list 101 permit ip 10.1.1.0 0.0.0.255 10.2.1.0 0.0.0.255
!
end
```

In Example 6-17, the switch subjects all traffic ingress from interface GigabitEthernet0/1 to the agg rate limiting policer of 500 Mbps. However, the switch rewrites the IP precedence value differently on class maps while maintaining the aggregate policer of 500 Mbps to traffic that matches either p

Use the following command to display the policy maps-to-policer configuration:

show mls qos aggregate-policer [*aggregate-policer-name*]

Example 6-18 illustrates the use of the show mls qos aggregate-policer command.

Example 6-18. Displaying Policy Map–to–Aggregate Policers Mapping

```
Switch#show mls qos aggregate-policer
aggregate-policer RATE_500MBPS 500000000 64000 exceed-action drop
Used by policy map AGGR_TRAFFIC_LIMITING
```

Port-Based, VLAN-Based, and Per-Port Per-VLAN-Based Policing

Port-based policing entails binding policy maps to individual ports. VLAN-based policing involves at
a policy map to a VLAN interface. The Catalyst 2950 Family of switches supports only port-based p
The Catalyst 3550 Family of switches supports port-based policing and a variant of VLAN-based po
referred to as per-port per-VLAN policing. The Catalyst 3550 Family of switches does not support a
policers to VLAN interfaces. Per-port per-VLAN policing consists of the typical class map clauses ne
second-class map with a VLAN-class match clause. The switch is only able to bind per-port per-VLA
trunk ports and VLAN access ports. Example 6-19 illustrates a sample configuration of per-port per
based policing.

Example 6-19. Sample Configuration of Per-Port Per-VLAN Policing

```
Switch#show running-config
Building configuration. . .
(text deleted)
!
mls qos
!
class-map match-any MATCH_LIST
  match access-group 100
  match ip precedence 5
  match ip dscp 35
class-map match-all MATCH_VLAN_LIST
  match vlan  2 100-105
  match class-map MATCH_LIST
!
!
policy-map RATE_LIMIT_VLAN_2_100-105
  class MATCH_VLAN_LIST
  set ip dscp 22


!
(text deleted)
```

```
!
interface GigabitEthernet0/1
 switchport trunk encapsulation dot1q
 switchport mode trunk
 no ip address
 service-policy input RATE_MARK
!
(text deleted)
!
access-list 100 permit ip 10.1.1.0 0.0.0.255 10.2.1.0 0.0.0.255
!
(text deleted)
!
end
```

The only difference between Example 6-19 and Example 6-17 is Example 6-19 applies the policy m to packets ingress from VLANs 2 and 100 through 105. This per-port per-VLAN configuration requi nested class map configuration to operate correctly. Furthermore, a per-port per-VLAN configuratio requires that the class map that nests the match VLAN classification rule with the regular class map use the match-all configuration option. Also, a per-port per-VLAN class map requires the match V class map clause before the match class map clause.

Policing Actions

The Catalyst 2950 Family and 3550 Family of switches support the following class actions:

- Trusting
- Marking
- Traffic-rate policing

Policing is similar to Cisco IOS rate limiting. Policing limits traffic flow the same as rate limiting. Ho policing uses the leaky token bucket algorithm, discussed in Chapter 2, to apply a burst parameter limiting. Both the Catalyst 2950 Family and 3550 Family of switches support dropping or marking traffic exceeding the rate.

Trusting Action

The use of trusting as a policing action is identical to trusting based on an ACL. The Catalyst 3550 f of switches supports trusting as a policing action. The supported trusting options include trusting C DSCP.

Policy maps organize the trusting actions using the following policy map class clause command:

```
trust [DSCP |  CoS]
```

Example 6-20 illustrates a Catalyst 3550 configured to trust DSCP for packets that match ACL 100.

Example 6-20. Sample Configuration of Trusting as a Policing Action

```
Switch#show running-config
Building configuration...
(text deleted)
!
mls qos
!
class-map match-all MATCH_ACL_100
  match access-group 100
!
!
policy-map Classify_ACL
  class MATCH_ACL_100
    trust dscp
!
```

```
!
(text deleted)
!
interface FastEthernet0/1
 switchport access vlan 2
 switchport voice vlan 700
 no ip address
 duplex full
 speed 100
 service-policy input Classify_ACL
 spanning-tree portfast
!
(text deleted)
!
access-list 100 permit ip 10.1.1.0 0.0.0.255 10.2.1.0 0.0.0.255
!
(text deleted)
end
```

For configurations using trusting in class map clauses, there is no need for a trusting configuration interface. You must carefully consider whether to configure an interface for trusting and whether to configure a policing action of trusting because the policing action takes precedence over the port configuration.

Marking Action

Both the Catalyst 2950 Family and 3550 Family of switches support marking as a policing action. B switches supports marking of DSCP using the following policy map class action command:

set ip dscp*new-dscp*

new-dscp indicates the DSCP value used to mark the frame in the policing action. The marking occ the internal DSCP value and the internal DSCP determines the DSCP value in the frame on egress. Catalyst 3550 Family of switches also supports marking of IP precedence. To configure marking of precedence in a policing action, use the following policy map class action command:

set ip precedence*new-precedence*

new-precedence indicates the IP precedence value used to mark the frame. The marking actually o the internal DSCP value by marking the 3 MSBs of the DSCP. Example 6-21 illustrates a Catalyst 3 switch configured to mark frames as a policing action.

Example 6-21. Sample Configuration of Marking as a Policing Action

```
Switch#show running-config
mls qos
!
class-map match-all MATCH_ACL_100
  match access-group 100
!
!
policy-map Mark_Frames
  class MATCH_ACL_100
    set ip dscp 45
```

```
!
(text deleted)
!
interface FastEthernet0/1
 switchport access vlan 2
 switchport voice vlan 700
 no ip address
 duplex full
 speed 100
 service-policy input Mark_Frames
 spanning-tree portfast
!
(text deleted)
!
access-list 100 permit ip 10.1.1.0 0.0.0.255 10.2.1.0 0.0.0.255
!
(text deleted)
end
```

Traffic-Rate Policing

The Catalyst 2950 Family and 3550 Family of switches use the leaky token bucket algorithm to det whether a packet is conforming or exceeding a specified policer rate. The leaky token bucket algori transparent to the policing behavior of the switch. An understanding of the leaky token bucket is or necessary when refining the burst size parameter of a policer. For more information about the leak bucket algorithm, see Chapter 2.

Use the following command to configure the Catalyst 2950 Family of switches traffic-rate and burst parameters of a class map policer:

```
police rate-bps burst-byte [exceed-action {drop | dscp dscp-value}]
```

Use the following commands to configure the rate and burst of a class map policer and an aggrega policer on a Catalyst 3550 switch, respectively:

```
police rate burst [exceed-action {drop | policed-DSCP-transmit}]]

mls qos aggregate-policer policer_name rate burst [exceed-action {transmit | drop |

➥ policed-DSCP-transmit}]
```

rate defines the actually policing rate. For the Catalyst 2950 Family of switches, the supported rate Mbps to 100 Mbps in 1-Mbps increments for Fast Ethernet interfaces and 8 Mbps to 1 Gbps for Giga capable interfaces. For the Catalyst 3550 Family of switches, the supported rates are 8 kbps to 2 G 1-bps increments for all interfaces. The switch may adjust the configured rate to a hardware suppo rate.

burst defines the burst size in bytes. The burst size needs to be at least the maximum packet size o frames touched by the policer for accurate policing. The following sections discuss determining the applicable burst size. For the Catalyst 2950 Family of switches, the supported burst sizes are 4096 16384, 32768, and 65536 bytes for Fast Ethernet ports and 4096, 8192, 16348, 32768, 65536, 13 and 262144 bytes for Gigabit Ethernet-capable interfaces. For the Catalyst 3550 Family of switches switches support configuring the burst size in the range of 8 MB to 2 GB. Example 6-22 illustrates a Catalyst 3550 configured for policing traffic at 100 Mbps with a burst size 16000 bytes.

Example 6-22. Sample Configuration of Rate Policing as a Policing Action

```
Switch#show running-config

mls qos

!

class-map match-all MATCH_ACL_100

  match access-group 100
```

```
!
!
policy-map RATE_RESTRICT
  class MATCH_ALL_PCKTS
    police 100000000 16000 exceed-action drop
!
(text deleted)
!
interface FastEthernet0/1
 switchport access vlan 2
 switchport voice vlan 700
 no ip address
 duplex full
 speed 100
 service-policy input Mark_Frames
 spanning-tree portfast
!
(text deleted)
!
access-list 100 permit ip 10.1.1.0 0.0.0.255 10.2.1.0 0.0.0.255
!
(text deleted)
end
```

The traffic-rate policer supports two actions for traffic exceeding the configured rate. These actions drop the packet or mark down the DSCP value of the frame. Marking down the frame actually occu internal DSCP value.

For the Catalyst 2950 Family of switches, use the follow command to configure the exceed-action o traffic-rate policer:

p**olice***rate-bps burst-byte* [**exceed-action** {**drop** | **dscp***dscp-value*}]

dscp-value indicates the DSCP value used to mark down the frame. Example 6-23 illustrates a sam configuration of marking down the DSCP value for traffic exceeding the rate specified in the policer

Example 6-23. Sample Configuration of Marking Down for the Exceeding of a Rate Policing

```
Switch#show running-config
Building configuration...
mls qos
!
class-map match-all MATCH_ACL_100
  match access-group 100
!
!
policy-map RATE_RESTRICT
  class MATCH_ALL_PCKTS
    police 100000000 16000 exceed-action dscp 35
!
(text deleted)
!
interface FastEthernet0/1
 switchport access vlan 2
 switchport voice vlan 700
```

```
 no ip address
 duplex full
 speed 100
 service-policy input Mark_Frames
 spanning-tree portfast
!
(text deleted)
!
access-list 100 permit ip 10.1.1.0 0.0.0.255 10.2.1.0 0.0.0.255
!
(text deleted)
end
```

For the Catalyst 3550 Family of switches, the switch uses a DSCP-policed transmit mapping table t
determine the DSCP value used for marking down the packet. By default, the policed DSCP mappin
maps packet DSCP values directly to the marked-down DSCP. As a result, the default behavior for
DSCP-policed transmit table is to not mark down the DSCP value associated with the packet. Use tl
following command to configure the policed DSCP mapping table:

mls qos map policed-dscp*dscp-list***to***mark-down-dscp*

dscp-list represents up to eight DSCP values. The switch marks down frames with these DSCPS to t
DSCP value specified by the *mark-down-dscp* value when exceeding the configured rate. To configu
exceed-action of the traffic-rate policer, use the following command:

```
police rate-bps burst-byte [exceed-action {drop | policed-dscp-transmit}]
```

Example 6-24 illustrates a user configuring and verifying the policed DSCP mapping table.

Example 6-24. User Configuring and Verifying the Policed DSCP Mapping

```
Switch#configure terminal
Switch(config)#mls qos map policed-dscp 30 31 32 33 34 35 to 20
Switch(config)#end
Switch#show mls qos map policed-dscp
   Policed-dscp map:
     d1 :  d2 0  1  2  3  4  5  6  7  8  9
     ----------------------------------------
      0 :    00 01 02 03 04 05 06 07 08 09
      1 :    10 11 12 13 14 15 16 17 18 19
      2 :    20 21 22 23 24 25 26 27 28 29
      3 :    20 20 20 20 20 20 36 37 38 39
      4 :    40 41 42 43 44 45 46 47 48 49
      5 :    50 51 52 53 54 55 56 57 58 59
      6 :    60 61 62 63
```

Burst Size

Because of the behavior of TCP/IP and UDP applications, packet drops due to policing may significa impact traffic throughput and may result in a packet-per-second throughput far below the configur policer rate. The burst parameter of policing attempts to handle this behavior by allowing periodic of traffic into the bucket.

Configuration of the burst size follows several other Catalyst platform recommendations. For TCP applications, use the following formula to calculate the burst size parameter used for policing:

<Burst> = 2 * <RTT> * <Rate>

RTT defines the approximate round-trip time for a TCP session. If RTT is unknown, use a RTT value
to 1 second depending on estimated latency. The burst calculation for a rate of 64 kbps and an RTT
ms is as follows

<Burst> = 2 * <.100 sec> * <64000 bits/sec>

<Burst> = 12800 bits = 1600 bytes

Nevertheless, from an application standpoint, rate policing always results in actual rates less than
configured rate regardless of the burst size. UDP applications react closer to the configured rate in
second; nevertheless, some UDP applications retransmit heavily upon packet loss resulting in perfo
far less than the configured rate. In brief, carefully consider burst rate and its effects on applicatio
applying policers.

Congestion Management and Avoidance

After the switch classifies, marks, and polices a packet against QoS rules, the switch queues the pa a transmit queue for output scheduling. Placing the packet into one of multiple queues allows for th to differentiate service by transmitting packets from the queue with specific order and priority. The 3550 Family of switches utilizes WRED and tail-drop queuing mechanisms for congestion avoidance Catalyst 2950 Family of switches does not support congestion avoidance. Both the Catalyst 2950 Fa 3550 Family of switches support four transmit queues. Although both switches support output sche utilizing WRR, the Catalyst 3550 Family of switches supports additional output scheduling and que configurations.

In brief, the Catalyst 2950 Family of switches supports the following output scheduling mechanism

- Strict-priority scheduling
- WRR scheduling

Strict-priority scheduling services packets placed into higher-priority queues before servicing packe lower-priority queues. Although this mechanism works well for high-priority traffic such as *Voice ov* (VoIP), this mechanism may starve transmission of traffic in lower-priority queues. As a result, the option of using WRR scheduling exists. WRR transmits traffic based on a weight value. In this mann queue receives an assigned weight of the total bandwidth. Therefore, during any period of time, th sends traffic out of every queue based on its weighted value. Subsequent sections discuss these me in more detail with configuration examples.

In brief, the Catalyst 3550 Family of switches supports the following queuing and output scheduling mechanisms:

- Expedite (strict-priority) queue
- WRR scheduling
- Configurable drop thresholds per output queue
- WRED congestion avoidance algorithm

The Catalyst 3550 Family of switches uses WRR scheduling for output scheduling in a manner simil Catalyst 2950 Family of switches. However, the Catalyst 3550 Family of switches allows for designa an expedite queue and configuration of congestion avoidance mechanisms using tail-drop threshold WRED. The designation of the expedite queue forces strict priority of transmission on the respectiv queue. The congestion avoidance algorithms attempt to subside congestion by dropping packets w priority before higher-priority packets in the same transmit queue. Later sections of this chapter di these features in more detail.

Congestion Management with the Catalyst 2950 and 3550 Family of Switc

As noted in the preceding section, both the Catalyst 2950 Family and Catalyst 3550 Family of switc perform output scheduling via strict-priority queuing and WRR, although each family of switches ex different default configurations. In summary, both the Catalyst 2950 Family and Catalyst 3550 Fam switches support the following congestion management configuration options:

- DSCP-to-CoS Mapping
- CoS-to-Transmit Queue Mapping
- Strict-Priority Queuing
- Configurable Weights for WRR

DSCP-to-CoS Mapping

The Catalyst 2950 Family and 3500 Family of switches use an internal DSCP value to differentiate s a packet traverses the switch. Because marking may occur on a packet during the marking and pol packet process, the CoS value of the packet needs to be updated on transmit. As a result, the switc employs the use of a DSCP-to-CoS mapping table for mapping the internal DSCP value to CoS valu packets before the switch schedules the packet into a transmit queue. Table 6-5 indicates the defau to-CoS mapping table.

Table 6-5. Default DSCP-to-CoS Mapping Table

DSCP Value	0–7	8–15	16–23	24–31	32–39	40–47	48–55	56
CoS Value	0	1	2	3	4	5	6	7

The DSCP-to-CoS mapping table is configurable on a global basis. Use the following global configu command to configure the DSCP-to-CoS mapping values:

mls qos map*dscp-cos dscp-list***to***cos*

dscp-list represents up to eight DSCP values separated by a space. CoS corresponds to the transmi value on the egress frame. Example 6-25 illustrates a user configuring and verifying the QoS DSCP mapping.

Example 6-25. Verifying and Configuring the DSCP-to-CoS Mapping

```
Switch#configure terminal
```

```
Switch(config)#mls qos map dscp-cos 37 38 39 to 5

Switch(config)#mls qos map dscp-cos 32 to 3

Switch(config)#end

Switch#show mls qos map dscp-cos

   Dscp-cos map:

     d1 :  d2 0  1  2  3  4  5  6  7  8  9

     ---------------------------------------

      0 :    00 00 00 00 00 00 00 00 01 01

      1 :    01 01 01 01 01 01 02 02 02 02

      2 :    02 02 02 02 03 03 03 03 03 03

      3 :    03 03 03 04 04 04 04 05 05 05

      4 :    05 05 05 05 05 05 05 05 06 06

      5 :    06 06 06 06 06 06 07 07 07 07

      6 :    07 07 07 07
```

CoS-to-Transmit Queue Mapping

With the Catalyst 2950 and 3550 Family of switches, output scheduling uses the strict-priority queu
WRR algorithm with four transmit queues. The switch queues traffic into the four transmit queues b
solely on the CoS value associated with the frame. By default, the switch places frames with higher
values into the higher-number queues. The internal DSCP value actually determines the CoS value
subsequently the transmit queue. Before the switches places packets into the transmit queues, the
determines the egress CoS value of the frame using the internal DSCP value and the DSCP-to-CoS
table.

In retracing the path of the packet, refer to the following outline and Figure 6-5 to understand the
internal DSCP in CoS-to-transmit queue mapping.

- Switch receives packet.
- Switch classifies packets with an internal DSCP value based on configuration.
- Switch polices, marks, or marks down the internal DSCP value.
- Switch determines egress CoS value based on internal DSCP value and DSCP-to-CoS mapping
- Switch queues the packet into one of four transmit queues based on egress CoS value, CoS-to
  queue mapping table, and any configured congestion avoidance configurations.

- Switch schedules packet out of the transmit queue based on strict-priority queuing or WRR.

Table 6-6 illustrates the default CoS-to-transmit queue mapping.

Table 6-6. Default CoS-to-Transmit Queue Mapping Table

CoS Value	Transmit Queue
0, 1	1
2, 3	2
4, 5	3
6, 7	4

Use the following command to display the CoS-to-transmit queue mapping:

show wrr-queue cos-map

The Catalyst 2950 Family of switches allows for custom configuration of the CoS-to-transmit queue on a global basis. The Catalyst 3550 Family of switches supports CoS-to-transmit queue mapping o interface basis. Use the following global and interface configuration command to configure the CoS transmit queue mapping:

wrr-queue cos-map*qid cos1..cosn*

q_id represents one of the four transmit queues. *cos1..cosn* represents configuration for up to eigh values to map to a transmit queue.

Example 6-26 illustrates a user custom configuring the CoS-to-transmit queue mapping table and

the DSCP-TxQueue mapping table configuration.

Example 6-26. User Configuring and Verifying the DSCP-to-Transmit Que Mapping Table

```
Switch#configure terminal

Switch(config)#wrr-queue cos-map 1 0 1 2

Switch(config)#wrr-queue cos-map 2 3 4

Switch(config)#wrr-queue cos-map 3 5

Switch(config)#end

Switch#show wrr-queue cos-map

CoS Value      :  0  1  2  3  4  5  6  7

Priority Queue :  1  1  1  2  2  3  4  4
```

Strict-Priority Queuing

By default, the Catalyst 2950 Family of switches schedules packets from transmit queues using the priority algorithm. Using this algorithm, the switch transmits all packets out of the higher-priority q before servicing lower-priority queues.

The Catalyst 3550 Family of switches utilizes the strict-priority queue configuration with WRR for s all the packets out of transmit queue four before servicing any other queue. The switch still utilizes scheduling packets out of the remaining queues. The behavior of this configuration is slightly differ the behavior on the Catalyst 2950 Family of switches. Priority Queuing on the Catalyst 2950 Family switches services higher-priority queues before lower-priority queues in order and does not suppor concept of a single expedite queue.

To configure the Catalyst 3550 Family of switches for the strict-priority scheduling of transmit queu use the following interface command:

priority-queue out

As a result of the strict-priority scheduling algorithm on both the Catalyst 2950 and 3550 Family of switches, the switches may never transmit traffic out of lower-priority queues when line-rate traffic higher-priority queues. Although this behavior is warranted for high-priority traffic such as VoIP, m applications of strict-priority queuing may starve lower-priority traffic. To remedy this situation, co configuring the switch for WRR bandwidth and WRED as discussed in the next section.

Configurable Weights for WRR

By default, the Catalyst 3550 Family of switches utilizes WRR for congestion management. The Cat 2950 Family of switches supports WRR as an optional configuration. Both the Catalyst 2950 Family 3550 Family of switches support adjusting the scheduling weight for each transmit queue. For the C 2950 Family of switches, WRR configuration is a global configuration. On the Catalyst 3550 Family switches, WRR configuration is per interface. The scheduling weight effectively determines the egre bandwidth per queue. Use the following egress bandwidth formula to determine WRR weight:

$(W/S) * B = n$

W represents the WRR weight of the queue, and *S* represents the sum of all weights of the active q is the available bandwidth of the outgoing interface(s), and *n* represents the egress bandwidth.

For example, each queue, 1 through 4, is assigned the following WRR weights on a Gigabit Etherne interface, respectively:

50 50 100 200

The*S* of the weights is 400. As a result of this configuration, each transmit queue receives the follo egress bandwidth:

Queue 1: 125 Mbps

Queue 2: 125 Mbps

Queue 3: 250 Mbps

Queue 4: 500 Mbps

To configure the scheduling weight of total bandwidth per queue, use the following global configura command for the Catalyst 2950 Family of switches and interface configuration command for the Ca 3550 Family of switches:

wrr-queue bandwidth*weight1...weight4*

weight1…weight4 represent four weighted values assigned to transmit queues 1, 2, 3, 4 respective the following command to display the configured WRR bandwidth per queue:

```
show wrr-queue bandwidth
```

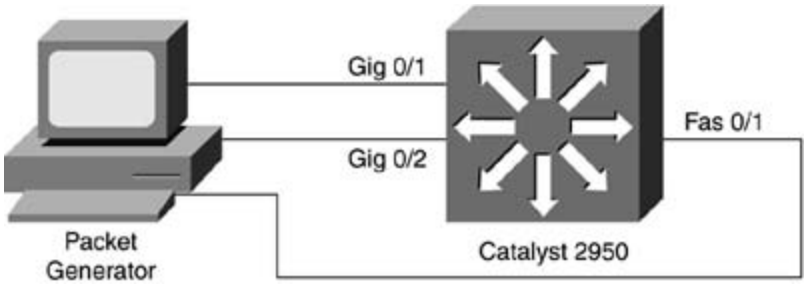
Example 6-27 illustrates a user configuring the WRR bandwidth per queue for heavier weights on h priority queues and verifying the WRR bandwidth configuration on a Catalyst 2950 switch. In this e queue 1 receives 10 percent of the bandwidth, queue 2 receives 20 percent of the bandwidth, queu receives 30 percent of the bandwidth, and queue 4 receives 40 percent of the bandwidth on egress

Example 6-27. User Configuring and Verifying a WRR Bandwidth Configu

```
Switch#configure terminal

Switch(config)#wrr-queue bandwidth 10 20 30 40

Switch(config)#end

Switch#show wrr-queue bandwidth

WRR Queue   :    1    2    3    4

Bandwidth   :   10   20   30   40
```

To demonstrate and measure the scheduling behavior of strict-priority queuing and WRR on transm queues, three packet-generator ports were connected to the switch as shown in Figure 6-11 using t configuration shown in Example 6-28. The first traffic-generator port connected to interface GigabitEthernet0/1 was sending 1 Gbps of traffic with a CoS value of 0, whereas the traffic-genera connected to interface GigabitEthernet0/2 was sending 100 Mbps of traffic with a CoS value of 5. A traffic generator connects to interface FastEthernet0/1 to measure the received rate of each packet Table 6-7 shows the result of a strict-priority queue versus several configurations using WRR with CoS-to-transmit queue mappings.

Figure 6-11. Network Topology for Demonstrating Strict-Priority Queu Versus WRR



Example 6-28. Interface Configuration for Demonstrating Strict-Priority Queuing

```
Switch#show running-config
Building configuration...
(text deleted)
!
interface FastEthernet0/1
 switchport access vlan 2
 switchport mode access
 mls qos trust cos
 spanning-tree portfast
!
(text deleted)
!
interface GigabitEthernet0/1
switchport access vlan 2
 switchport mode access
 mls qos trust cos
 spanning-tree portfast
!
!
interface GigabitEthernet0/2
```

```
switchport access vlan 2
 switchport mode access
 mls qos trust cos
 spanning-tree portfast
!
end
```

Table 6-7. Demonstrating Strict-Priority Queuing Versus WRR on Tran Queues on the Catalyst 2950

Trial Description	No. of Packets with CoS = 0 Received	No. of Packets with Co Received
Default configuration (strict-priority queuing)	0	148,805
WRR queue bandwidth 25 25 25 25	74,400	74,400
WRR queue bandwidth 10 20 30 40	35,705	113,100
WRR queue bandwidth 10 20 100 100	13,000	135,805

Congestion Avoidance with the 3550 Family of Switches

The Catalyst 3550 Family of switches supports the following congestion avoidance mechanisms on Ethernet-capable interfaces:

- Tail-drop thresholds
- WRED drop thresholds
- Transmit queue size manipulation

Furthermore, the Catalyst 3550 Family of switches supports only configuration of minimum reserve for congestion avoidance on Fast Ethernet interfaces.

Tail Drop

The Catalyst 3550 Family of switches utilizes two thresholds for congestion avoidance. By default, t maps all packets to a single threshold per queue. As a result, when a transmit queue becomes full, switch drops any further packets needing placement into the respective transmit queue. This behav

default when any queue becomes congested and is subsequently unable to hold additional packets.

The Catalyst 3550 Family of switches allows for two tail-drop thresholds, whereas the switch tail dr packets with a lower priority over packets with a higher priority. The switch CLI refers to these thre thresholds-1 and thresholds-2, respectively. The Catalyst 3550 Family of switches employs prioritie thresholds strictly on internal DSCP. The ingress interface determines the egress tail-drop threshol mapping and this configuration is only applicable to Gigabit Ethernet-capable interfaces. As a resul switch treats all ingress packets from Fast Ethernet interfaces with the threshold-2 configuration.

In practice, tail-drop thresholds are useful in mitigating congestion. Consider, for example, a large consisting of high-priority data and voice applications. The high-priority data application is built on and provides for file sharing between network users. The application adapts well to TCP/IP back-pr and packet loss. The network uses voice applications for IP telephony. Because both applications a the network administrators assign a DSCP value of 40 to the file-sharing application while using th DSCP value associated with Cisco IP Phones, 46. Packets from both applications occupy transmit qu The switch is using WRR to service queue 3 with half the available bandwidth on egress interfaces. implementing tail-drop thresholds, the network administrators noticed that the file-sharing applica filing the transmit queue 3 causing excessive drops for voice applications. To remedy this situation network administrators implemented tail-drop thresholds. The network administrators configured t to tail drop packets from the file-sharing application when the queue becomes 30 percent full while dropping voice application traffic when the queue is 100 percent full. This configuration yields amp transmit queue space for the voice applications, while the file-sharing program maintained high-pr service in transmit queue 3 but was unable to monopolize the buffer space.

To configure an interface for tail-drop congestion avoidance, use the following interface command.

wrr-queue threshold*queue-id threshold-percentage1 threshold-percentage2*

queue-id refers to the respective transmit queue, 1 through 4. *threshold-percentage1* and *threshol percentage2* refer to the tail-drop percentage thresholds respectively. Valid percentages are in the to 100.

To configure the ingress interface for threshold mapping, use the following interface configuration command:

wrr-queue dscp-map*threshold-id dscp1 ... dscp8*

threshold-id represents a number 1 or 2 for *threshold-precentage1* or *2*, respectively. *dscp1 … dscp* represents up to eight DSCP values for threshold mapping.

> **NOTE**
>
> As mentioned previously, the DSCP-to-threshold mapping resides on the ingress interface of traffic. As a result, for packets to adhere to specific thresholds on the egress interfaces, the sw requires the mapping configuration on the ingress interfaces. In addition, only Gigabit Etherne capable interfaces support DSCP-to-threshold mapping.

Example 6-29 illustrates sample interface configurations for DSCP mapping to tail-drop thresholds. GigabitEthernet0/1 is the ingress interface, whereas GigabitEthernet0/2 is the egress interface of t flow. Ingress packets with DSCP values 40 and 46 on GigabitEthernet0/1 map to threshold 2. The t percentages are 50 percent and 100 percent, respectively.

Example 6-29. Sample Interface Configuration for Congestion Avoidance Tail-Drop Thresholds

```
Current configuration : 198 bytes
!
interface GigabitEthernet0/1
 switchport trunk encapsulation dot1q
 switchport mode trunk
 no ip address
 wrr-queue dscp-map 2 40 46
!
interface GigabitEthernet0/2
 switchport trunk encapsulation dot1q
 switchport mode trunk
 no ip address
 mls qos trust dscp
 wrr-queue threshold 1 50 100
```

!

WRED Drop

With the tail-drop congestion avoidance mechanism, the Catalyst 3550 Family of switches drops pa
when queues reach a certain percentage threshold or become full. The use of tail drop may result i
undesirable behavior of applications that specifically use TCP/IP. When a queue becomes full or rea
certain percentage for packets matching a specific threshold, the tail drop instantaneously drops pa
until the respective queue is no longer full. When these packet drops occur, TCP/IP applications rec
bandwidth accordingly. When the queue is no longer full and able to accept more packets, howeve
applications begin to increase throughput, which results in another full transmit queue condition. T
the transmit queue more effectively and prevent TCP/IP from increasing and decreasing throughpu
relatively the same time, the condition also known as *global synchronization of TCP*, the Catalyst 3
Family of switches supports WRED. Chapter 2, in the "Congestion Avoidance" section, discusses glo
synchronization of TCP in more detail.

The Catalyst 3550 Family of switches uses the WRED algorithm to randomly drop packets in a tran
queue before thresholds and queue full conditions. In this manner, multiple TCP/IP applications ra
reduce bandwidth instead of simultaneously reducing the transmit rate. Tail-drop conditions may s
when using the WRED algorithm under periods of considerable congestion, low-percentage thresho
or with applications that do not decrease throughput accordingly. A practical example is using WRE
handle congestion on an interface to a core switch that carries multiple file transfers and voice app
traffic. To prevent congestion of higher-priority traffic and prevent all the file transfers from throttl
bandwidth at the same time, WRED provides the best solution over tail dropping.

Configuration and implementation of WRED mirrors tail drop. The Catalyst 3550 Family of switches
two configurable thresholds that signify percentages to randomly discard packets. In this manner,
switch is configurable to allow for random drop of packets with a lower priority over packets with a
priority. The switch CLI refers to these thresholds as thresholds-1 and thresholds-2, respectively. T
Catalyst 3550 Family of switches employs assignment to thresholds strictly based on internal DSCP
ingress interface determines the egress tail-drop threshold mapping and this configuration is only a
to Gigabit Ethernet-capable interfaces. As a result, the switch treats all ingress packets from Fast E
interfaces with the threshold-2 configuration.

To configure the egress interface for WRED, use the following command:

wrr-queue random-detect max-threshold*queue-id threshold-percentage1 threshold-perc*

queue-id refers to the respective transmit queue, 1 through 4. *threshold-percentage1* and *threshol*
percentage2 refer to the tail-drop percentage thresholds, respectively. Valid percentages are in the

to 100.

To configure the ingress interface for threshold mapping, use the following interface configuration command:

wrr-queue dscp-map*threshold-id dscp1 ... dscp8*

threshold-id represents a number 1 or 2 for *threshold-precentage1* or *2*, respectively. *dscp1 … dscp* represents up to eight DSCP values for threshold mapping.

Example 6-30 shows a sample interface configuration for DSCP mapping to WRED thresholds. GigabitEthernet0/1 is the ingress interface, whereas GigabitEthernet0/2 is the egress interface of t flow. Ingress packets with DSCP values 40 and 46 on GigabitEthernet0/1 map to threshold 2. The t percentages are 50 percent and 100 percent, respectively.

Example 6-30. Sample Interface Configuration for Congestion Avoidance WRED Thresholds

```
Current configuration : 198 bytes
!
interface GigabitEthernet0/1
 switchport trunk encapsulation dot1q
 switchport mode trunk
 no ip address
 wrr-queue dscp-map 2 40 46

!
interface GigabitEthernet0/2
 switchport trunk encapsulation dot1q
 switchport mode trunk
```

```
 no ip address
 mls qos trust dscp
 wrr-queue random-detect max-threshold 1 20 90
 wrr-queue random-detect max-threshold 2 25 100
 wrr-queue random-detect max-threshold 3 50 100
!
```

Use the following command to gather statistics about queue drops per threshold:

show mls qos interface [*interface_name*]**statistics**

Example 6-31 illustrates sample output from the show mls qos interface statistics command for statistics.

Example 6-31. Sample Output from the show mls qos interface statistics Command for WRED Statistics

```
Switch#show mls qos int gig0/1 statistics
GigabitEthernet0/1
Ingress
  dscp: incoming   no_change  classified policed    dropped (in bytes)
Others: 0          0          0          0          0
Egress
  dscp: incoming   no_change  classified policed    dropped (in bytes)
Others: 547610028     n/a       n/a      0          0
WRED drop counts:
```

```
  qid  thresh1    thresh2   FreeQ
   1 : 385873     7637384   511
   2 : 0          0         1024
   3 : 0          0         1024
   4 : 0          0         1024
```

Transmit Queue Size

The Catalyst 3550 Family of switches allows for transmit queue size manipulation on a per-queue b
Gigabit Ethernet-capable interfaces. By default, each queue receives 25 percent of the transmit que
the time of publication, the Catalyst 3550 Family of switches supports 4096 packet queues per Giga
Ethernet interface; subsequently, each queue receives 1024 packet queues per Gigabit Ethernet int
Queue size manipulation is useful when wanting to increase or decrease queue sizes based on prot
application. For example, queuing large amounts of voice and live video is not necessary. By the ti
voice or video arrives at the destination, the application will most likely discard the packets. Hearir
delayed audio or seeing delayed video for live sources such as IP telephony is undesirable. Neverth
data transfers are less sensitive to delay. As a result, storing a fair amount of data packets in a que
not affect the performance of the data-centric applications.

A practical example is to minimize the queue size for delay-sensitive applications such as voice and
while utilizing larger queue sizes for data transfers. To configure the percentage of transit queues s
each queue, use the following interface command:

wrr-queue queue-limit*weight1 weight2 weight3 weight4*

Use the following formula to determine the queue size for a Gigabit Ethernet interface:

$(W/S) * Q = n$

W represents the WRR weight of the queue, and *S* represents the sum of all weights of the active q
is the total queue size of the outgoing interface. Gigabit Ethernet interfaces on the Catalyst 3550 F
switches utilize 4096 packet buffers. *n* represents the queue size per queue.

For example, each queue, 1 through 4, is assigned the following queue size weights on a Gigabit E
interface, respectively:

10 20 30 40

The *S* of the weights is 100. As a result of this configuration, each interface configures for the follow queue sizes:

Queue 1: 410

Queue 2: 820

Queue 3: 1228

Queue 4: 1638

Because of hardware implementation, the Catalyst 3550 Family of switches rounds the queue size t closest configurable queue size. Use the show mls qos interface*interfaces*statistics command wh port is in the shutdown state to view the exact queue size provided by the hardware.

Minimum Buffer Size for Fast Ethernet Interfaces

The Catalyst 3550 Family of switches supports configuration of minimum buffer size for Fast Etherr interfaces. The buffer size manipulation allows for adjusting the buffer queue size for different type traffic. Generally, time-sensitive data such as VoIP traffic requires smaller queue sizes to avoid jitt whereas large data transfer of images and files benefits from larger queue size.

The Catalyst 3550 Family of switches uses globally configured, minimum buffer levels for applicatic Ethernet interfaces. These switches support up to 8 distinct buffer levels, and each buffer level den packet buffer size in a range of 10 to 170 packets. Use the following global command to configure t minimum buffer levels:

[**no**]**mls qos min-reserve***min-reserve-level min-reserve-buffersize*

min-reserve-level represents of the eight global configurable buffer levels, and *min-reserve-buffers* represents the buffer size in number of packets.

To attach the *mim-reserve-level* to an interface queue, use the following command:

[**no**]**wrr-queue min-reserve***queue-id min-reserve-level*

queue-id represents the interface transmit queue; *min-reserve-level* refers to the global minimum l
level to attach to the transmit queue.

Use the following command to verify the minimum buffer level configuration:

show mls qos interface [*interface_name*]**buffers**

Example 6-32 illustrates a user displaying a sample interface configuration using minimum buffer l
verifying the configuration. In this example, the switch uses small packet buffers for higher-priority
and larger packet buffers for low-priority traffic:

Example 6-32. User Verifying Configuration of Minimum Buffer Levels

```
Switch#show running-config
Building configuration...
!
(text deleted)
!
mls qos min-reserve 1 170
mls qos min-reserve 7 20
mls qos min-reserve 8 10
mls qos
!
!
interface FastEthernet0/4
 no ip address
```

```
 mls qos trust dscp
 wrr-queue bandwidth 10 20 30 40
 wrr-queue min-reserve 2 5
 wrr-queue min-reserve 3 7
 wrr-queue min-reserve 4 8
!
(text deleted)
!
end
Switch#show mls qos interface buffers
FastEthernet0/1
Minimum reserve buffer size:
 170 100 100 100 100 100 20 10
Minimum reserve buffer level select:
 1 2 3 4
```

Auto-QoS

Auto-QoS eases the deployment of QoS on Catalyst switches by applying recommended QoS configuration for typical networks, especially those deploying VoIP. For the Catalyst 2950 Family and Catalyst 3550 Family of switches, the Auto-QoS feature, as supported in 12.1.12c(EA1), aids in configuration of QoS for classification, egress scheduling, and congestion management of VoIP traffic for Cisco IP telephony. Upcoming software versions will offer additional Auto-QoS features.

Auto-QoS Classification

The Catalyst 2950 Family and 3550 Family of switches support two methods of classification when Auto-QoS is enabled. The first method classifies traffic strictly based on ingress CoS. Moreover, this configuration option just adds the trust CoS command to the interface for classification purposes. This method is useful for classifying traffic for interfaces connecting other switches.

The second method uses the trust Cisco IP Phone classification method as discussed in the "Trust Cisco IP Phone Device" section of this chapter. In brief, this method trusts ingress CoS only when a Cisco IP Phone is connected to an interface. This configuration option just adds the mls qos trust device cisco-phone and mls qos trust cos commands to the interface for classification.

To configure an interface for Auto-QoS of classification based on trust, use the following interface command:

```
auto qos voip trust
```

To configure an interface for Auto-QoS of classification based on trust and whether a Cisco IP phone is discovered on the interface, use the following interface command:

```
auto qos voip cisco-phone
```

Both of these classification options for Auto-QoS trust CoS for ingress frames. With these configuration, the switch classifies all untagged frames with a CoS value of zero regardless of the DSCP value. As a result, Auto-QoS is not the method of choice for applying classification based on DSCP or IP precedence.

Auto-QoS Congestion Management

Both the configuration options discussed in the preceding section also configure the Catalyst 2950 Family and Catalyst 3550 Family of switches for congestion management.

Foremost, the Auto-QoS feature modifies the CoS-to-DSCP mapping resulting in the nondefault internal DSCP values for classified frames. On the Catalyst 2950 Family of switches, the CoS-to-DSCP mapping is a global configuration. On the Catalyst 3550 Family of switches, the CoS-to-DSCP mapping is an interface configuration. The nondefault CoS-to-DSCP mapping is a follows for CoS values 0 through 7, respectively: 0 8 16 26 32 46 48 56. In addition, Auto-QoS configures both switches for mapping CoS 5 to queue 4.

In addition, the Auto-QoS feature modifies egress scheduling such that the switch only uses three egress transmission queues. The Auto-QoS feature achieves this configuration by assigning CoS value 5 to queue 4; CoS values 3, 6, and 7 to queue 3; and CoS values 0, 1, and 2 to queue 1. The Auto-QoS feature assigns the CoS value of 5 to queue 4, the expedite queue, because Cisco IP Phones mark voice traffic with CoS value 5. The Auto-QoS assigns CoS value 3 along with the CoS value 6 and 7, which are used by routing protocols and spanning-tree *bridge protocol data units* (BPDUs), because Cisco IP Phones mark control traffic with a CoS value of 3.

Auto-QoS also assigns different weights for WRR to each queue. For the Catalyst 2950, queuing is based on strict priority for queue 4. Therefore, queue 4 does not receive a weighted value. Similarly, for the Catalyst 3550 Family of switches, Auto-QoS configures queue 4 as the expedite queue. On both switches, queue 3 and queue 1 receive weights of 80 percent and 20 percent, respectively.

Finally, on the Catalyst 3550, Auto-QoS adjusts the buffer size of the Gigabit Ethernet and Fast Ethernet interface queues. The higher-priority queues receive a smaller queue size, whereas the lower-priority queues receive a larger queue size. This algorithm is used because high-priority traffic is affected by jitter and queuing packets increases jitter. Furthermore, data traffic occupying lower-priority queues is less affected by jitter; therefore, storing more packets prevents excessive drops while maintaining low-priority service.

In summary, the Auto-QoS provides a macro for configuring classification and congestion management based on trusting CoS for all ingress frames or only on interfaces connected to Cisco IP Phones.

Auto-QoS on the Catalyst 2950 Family and Catalyst 3550 Family of switches adds the following configuration commands as illustrated in Tables 6-8 to 6-12 in software version 12.1.12c(EA1):

Table 6-8. Catalyst 2950 Auto-QoS-Added Global Configuration Commands

```
wrr-queue bandwidth 20 1 80 0
wrr-queue bandwidth 20 1 80 0
wrr-queue cos-map 1 0 1 2 4
wrr-queue cos-map 3 3 6 7
wrr-queue cos-map 4 5
mls qos map cos-dscp 0 8 16 26 32 46 48 56
```

Table 6-9. Catalyst 2950 Auto-QoS-Added Interface Configuration Commands

```
mls qos trust cos
mls qos trust device cisco-phone
```

The `mls qos trust device cisco-phone` command is only added by Auto-QoS when the interface is configured for the `cisco-phone` option.

Table 6-10. Catalyst 3550 Auto-QoS-Added Global Configuration Commands

```
mls qos map cos-dscp 0 8 16 26 32 46 48 56
mls qos min-reserve 5 170
mls qos min-reserve 6 10
mls qos min-reserve 7 65
mls qos min-reserve 8 26
mls qos
```

Table 6-11. Catalyst 3550 Auto-QoS-Added Gigabit Interface Configuration Commands

```
mls qos trust cos
mls qos trust device cisco-phone
wrr-queue bandwidth 20 1 80 0
wrr-queue queue-limit 80 1 20 1
wrr-queue cos-map 1 0 1 2 4
wrr-queue cos-map 3 3 6 7
wrr-queue cos-map 4 5
priority-queue out
```

Table 6-12. Catalyst 3550 Auto-QoS-Added Fast Ethernet Interface Configuration Commands

```
mls qos trust cos
mls qos trust device cisco-phone
wrr-queue bandwidth 20 1 80 0
wrr-queue min-reserve 1 5
wrr-queue min-reserve 2 6
wrr-queue min-reserve 3 7
wrr-queue min-reserve 4 8
wrr-queue cos-map 1 0 1 2 4
wrr-queue cos-map 3 3 6 7
wrr-queue cos-map 4 5
priority-queue out
```

Auto-QoS supports a debug output that displays all commands applied globally and to each interface upon configuration of Auto-QoS on an interface. Use the following debug command to display this output:

```
debug autoqos
```

In addition, use the following command to display all configuration commands previously added

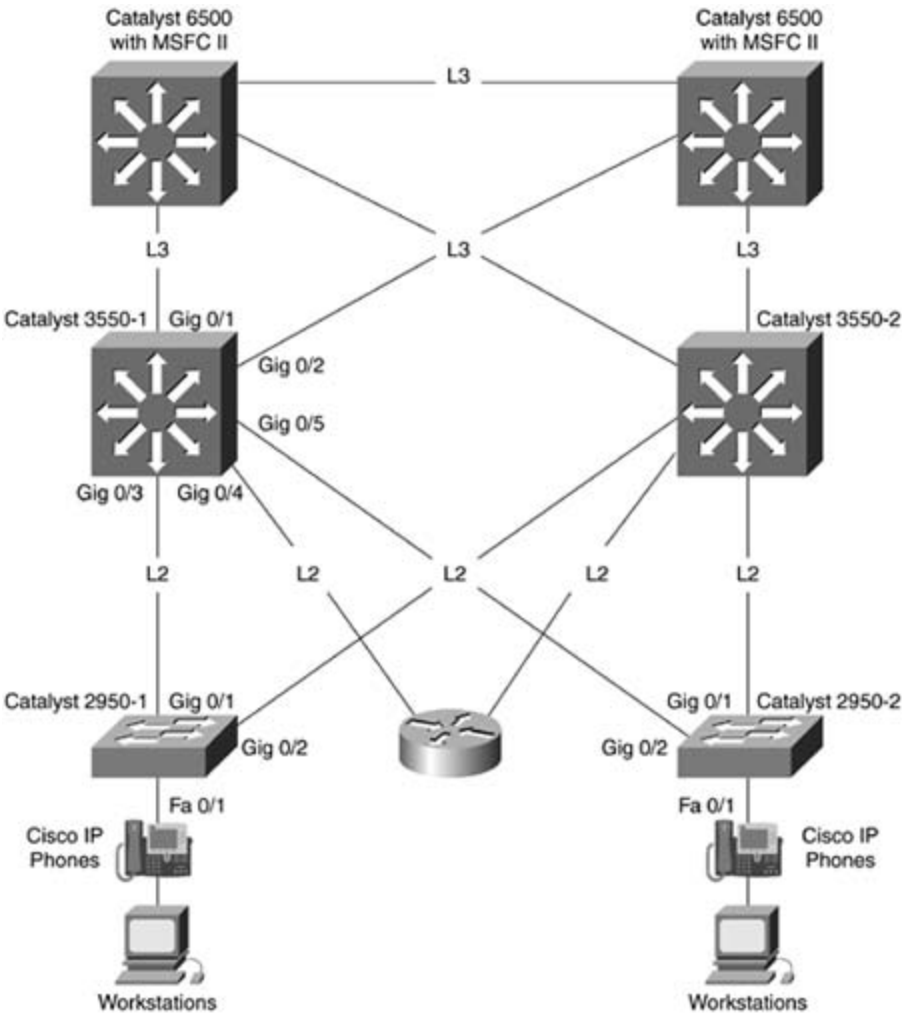
by Auto-QoS:

```
show auto qos
```

Case Study

Figure 6-12 shows several QoS features in a campus network topology. To illustrate several QoS features in a single case study, this case study exaggerates the practicality of the use of QoS in this topology.

Figure 6-12. Case Study Topology



Example 6-33 illustrates the configuration for the Catalyst 2950s in Figure 6-12. A review of the configuration follows the example.

Example 6-33. Catalyst 2950 Case Study Configuration

```
Switch#show running-config
Building configuration...
!
(text deleted)
!
wrr-queue bandwidth 10 20 30 40
wrr-queue cos-map 1 0 1
wrr-queue cos-map 2 2 3
wrr-queue cos-map 3 4
wrr-queue cos-map 4 5 6 7
!
!
class-map match-all MATCH_FILE_SHARING
  match access-group 100
!
!
policy-map RATE_LIMIT_FILE_SHARING
  class MATCH_FILE_SHARING
    police 8000000 8192 exceed-action drop
!
(text deleted)
!
interface FastEthernet0/1
 switchport access vlan 2
 switchport voice vlan 701
 no ip address
 service-policy input RATE_LIMIT_FILE_SHARING
 mls qos trust device cisco-phone
```

```
 mls qos trust dscp
 spanning-tree portfast
!
(text deleted)
!
interface GigabitEthernet0/1
 switchport mode trunk
 no ip address
 mls qos trust dscp
!
(text deleted)
!
access-list 100 permit tcp any any eq 29999
!
(text deleted)
!
end
```

The Catalyst 2950s in Figure 6-12 act as access layer switches and configure identically. These switches aggregate access ports for Cisco IP Phones and workstations. In this topology, the workstations connect directly to the Cisco IP Phones. Because the Cisco IP Phones assign a DSCP value of 46 to all egress voice frames, an ideal configuration is to trust DSCP of ingress frames on the switch interface when a Cisco IP Phone is attached. As a result, the sample configuration uses the `trust dscp` command based on an attached Cisco IP Phone. The commands `mls qos trust device cisco-phone` and `mls qos trust dscp` achieve this configuration.

Furthermore, to isolate the VoIP traffic into a separate VLAN, the configuration applies voice VLANs; the example uses VLAN 701 for voice traffic and VLAN 2 for all workstation traffic. The `spanning-tree portfast` configuration command tells the switch to immediately forward frames on linkup instead of waiting on spanning-tree convergence.

The design uses a policer to limit the amount of file-sharing traffic received on each port. The policer classifies the file-sharing traffic on TCP port 29999 and limits the traffic to 8 Mbps ingress on each interface.

For output scheduling, the design uses WRR for all interfaces with each queue 1 through 4 getting 10 percent, 20 percent, 30 percent, and 40 percent of the egress bandwidth,

respectively. In addition, all frames with a CoS value of 5 are designated to the highest queue 4. Because all VoIP traffic uses CoS 5 by default, CoS 5 frames were chosen for queue 4.

Example 6-34 illustrates the configuration for the Catalyst 3550s in Figure 6-12. A review of the configuration follows the example.

Example 6-34. Catalyst 3550 Case Study Configuration

```
Switch#show running-config
Building configuration...
!
(text deleted)
!
mls qos map dscp-mutation 40_down_32 40 to 32
mls qos map policed-dscp  1 2 3 4 5 6 7 8 to 0
mls qos map policed-dscp  9 10 11 12 13 14 15 16 to 0
mls qos map policed-dscp  17 18 19 20 21 22 23 24 to 0
mls qos map policed-dscp  25 26 27 28 29 30 31 32 to 0
mls qos map policed-dscp  33 34 35 36 37 38 39 to 0
mls qos map policed-dscp  40 41 42 43 44 45 46 47 to 1
mls qos map policed-dscp  48 49 50 51 52 53 54 55 to 1
mls qos map policed-dscp  56 57 58 59 60 to 1
mls qos map policed-dscp  61 62 63 to 2
!
!
class-map match-all RESTRICT_INTERNET_ONLY
  match access-group 151
!
!
policy-map RESTRICT_INTERNET
  class RESTRICT_INTERNET_ONLY
```

```
    police 10000000 16000 exceed-action policed-dscp-transmit
!
interface GigabitEthernet0/1
 ip address 10.1.1.0 255.255.255.248
 mls qos trust dscp
 service-policy input RESTRICT_INTERNET
 wrr-queue random-detect max-threshold 1 20 100
 wrr-queue random-detect max-threshold 2 40 100
 wrr-queue cos-map 4 5
 priority-queue out
!
!
interface GigabitEthernet0/2
 ip address 10.1.2.0 255.255.255.248
 mls qos trust dscp
 service-policy input RESTRICT_INTERNET
 wrr-queue random-detect max-threshold 1 20 100
 wrr-queue random-detect max-threshold 2 40 100
 wrr-queue cos-map 4 5
 priority-queue out
!
!
interface GigabitEthernet0/3
 ip address 10.10.1.2 255.255.255.0
 mls qos trust dscp
 service-policy input RESTRICT_INTERNET
 wrr-queue random-detect max-threshold 1 20 100
 wrr-queue random-detect max-threshold 2 40 100
```

```
 wrr-queue cos-map 4 5
 priority-queue out
 standby 101 ip 10.10.1.1
!
!
interface GigabitEthernet0/4
 switchport trunk encapsulation dot1q
 switchport mode trunk
 ip address 10.10.2.2 255.255.255.0
 mls qos trust dscp
 service-policy input RESTRICT_INTERNET
 wrr-queue random-detect max-threshold 1 20 100
 wrr-queue random-detect max-threshold 2 40 100
 wrr-queue cos-map 4 5
 priority-queue out
 standby 102 ip 10.10.2.1
!
interface GigabitEthernet0/5
 switch access vlan 5
 switchport mode access
 ip address 10.11.1.1 255.255.255.252
 mls qos trust dscp
 mls qos dscp-mutation 40_down_32
 service-policy input RESTRICT_INTERNET
 wrr-queue random-detect max-threshold 1 20 100
 wrr-queue random-detect max-threshold 2 40 100
 wrr-queue cos-map 4 5
 priority-queue out
```

```
!
 (text deleted)
!
access-list 151 permit ip any 10.0.0.0 0.0.0.255
access-list 151 permit ip 10.0.0.0 0.0.0.255 any
access-list 151 permit ip any 172.16.0.0 0.0.255.255
access-list 151 permit ip 172.16.0.0 0.0.255.255 any
!
(text deleted)
!
end
```

The Catalyst 3550s in Figure 6-12 act as distribution layer switches, and both Catalyst 3550s configure identically except for IP addresses. These switches aggregate access layer switches for distribution into the core. As a result, the design utilizes the Catalyst 3550s as IP routers; hence, the IP addresses on the Gigabit Ethernet interfaces.

Because these Catalyst 3550s act as distribution switches on interfaces GigabitEthernet0/1 through 0/4, there is no need to reclassify traffic. As a result, these switches accept the classification and marking done by the access layer Catalyst 2950 switches and the core Catalyst 6500 switches.

The switch mutates DSCP from the router connected to interface GigabitEthernet0/5. The DSCP mutations mark down frames with DSCP values of 40 to 32. In effect, this configuration is marking down IP precedence values of 5 to 4. The goal of this configuration is to reduce the priority of traffic originating from networks connected off the router.

With regard to policing, the distribution switches are responsible for limiting traffic destined strictly for the Internet. The campus network uses the private network addresses, 10.0.0.0/8 and 172.16.0.0/16, for all intranet traffic. As a result, the Catalyst 3550 switches rate limits the Internet traffic to 100 Mbps per link to each core switch. The design uses the Catalyst 3550 rather than the Catalyst 2950 for limiting intranet traffic due to the larger number of ACLs and ACL options supported on the Catalyst 3550 Family of switches.

For congestion management, the design uses WRR for each interface with bandwidth configurations of 100 Mbps, 200 Mbps, and 300 Mbps for each transmit queue 1 through 3. The switch transmits traffic occupying queue 4 before the switch services any other queue because it is configured as a priority queue. As a result, the WRR bandwidth associated with this queue has no meaning. For queues 1 through 3, using WRR with the configured bandwidth parameters allows for higher-priority traffic to consume more bandwidth per interface. Furthermore, each interface employs the use of congestion avoidance through WRED. Only the low-priority queues drop traffic at specific thresholds.

This configuration assists in avoiding congestion on a per-queue basis. Finally, the wrr-queue

cos-map 4 5 configuration informs the switch to map CoS 5 traffic to transmit queue 4. Because CoS 5 traffic usually represents VoIP traffic, the ideal situation is to transmit the traffic with strict priority.

Summary

The Catalyst 2950 Family of switches provides for a wide range of QoS features suited specifically for access layer implementation. You can summarize QoS feature support on the Catalyst 2950 Family of switches as follows:

- Classification, marking, and policing require the EI software.
- Support exists for classification based on port CoS configuration, trusting configuration, and ACLs.
- Per-port ingress policing is supported.
- DSCP-to-CoS mapping table is configurable on a global basis.
- Output scheduling uses strict-priority queuing by default.
- WRR scheduling is configurable as an alternative to strict-priority queuing.

The Catalyst 3550 Family of switches provides for a wide range of QoS features suited specifically for access layer and distribution layer implementation. You can summarize QoS feature support on the Catalyst 3550 Family of switches as follows:

- No QoS feature differences exist between SMI and EMI software versions.
- Support exists for classification based on port CoS configuration, trusting configuration, and ACLs.
- Ingress and egress policing is supported on individual interfaces.
- A variant of VLAN-based policing exists as per-port VLAN-based policing.
- The DSCP-to-CoS mapping table is configured on a per-interface basis.
- Congestion management uses WRR with a strict-priority queuing option.
- Congestion avoidance utilizes the tail-drop and WRED techniques.

Chapter 7. QoS Features Available on the Catalyst 4000 IOS Family of Switches and the Catalyst G-L3 Family of Switches

This chapter continues the tour of QoS feature support on Catalyst switches. This chapter covers two distinct product lines, the Catalyst 4000 IOS Family of switches and the Catalyst G-L3 Family of switches. The Catalyst 4000 IOS Family of switches encompasses any Catalyst 4000 chassis with a Supervisor III or IV Engine. The Catalyst G-L3 switches includes the Catalyst 2948G-L3, 4908G-L3, and the Catalyst WS-X4323-L3 (the Layer 3 services module for a Catalyst 4000 switch). Table 7-1 describes the various Catalyst 4000 platforms. Table 7-1 is identical to Table 3-7 in Chapter 3, "Overview of QoS Support on Catalyst Platforms and Exploring QoS on the Catalyst 2900XL, 3500XL, and Catalyst 4000 CatOS Family of Switches," and is included here for quick reference.

Table 7-1. Catalyst 4000 CatOS Versus Cisco IOS Software Platform Matrix

Catalyst 4000 Model	Reference Family	Description
Catalyst 2948G	Catalyst 4000 CatOS	48-port 10/100BASE-TX switch ports + 2 1000BASE-X GBIC[*] switch ports
Catalyst 2980G	Catalyst 4000 CatOS	80-port 10/100BASE-TX switch ports + 2 1000BASE-X GBIC switch ports
Catalyst 2980G-A	Catalyst 4000 CatOS	80-port 10/100BASE-TX switch ports + 2 1000BASE-X GBIC switch ports
Catalyst 2948G-L3	Catalyst G-L3 Switch	48-port 10/100BASE-TX + 2 1000BASE-X GBIC Layer 3 switche ports
Catalyst 4003 + WS-X4012 Supervisor I Engine	Catalyst 4000 CatOS	3-slot modular chassis + 2 1000BASE-X GBIC ports on Layer 2 supervisor
Catalyst 4006 + WS-X4013 Supervisor II Engine	Catalyst 4000 CatOS	6-slot modular chassis + 2 1000BASE-X GBIC ports on Layer 2 supervisor
Catalyst 4006 + WS-X4014 Supervisor III Engine	Catalyst 4000 IOS	6-slot modular chassis + 2 1000BASE-X GBIC ports on Layer 2/3 supervisor
Catalyst 4006 + WS-X4515 Supervisor IV Engine	Catalyst 4000 IOS	6-slot modular chassis + 2 1000BASE-X GBIC ports on Layer 2/3 supervisor

Catalyst WS-X4232-L3 Layer 3 Services Module	Catalyst G-L3 Switch	Layer 3 router module for Catalyst 4003 and 4006 chassis with Supervisor I or II Engine
Catalyst 4503 + WS-X4013 Supervisor III Engine	Catalyst 4000 CatOS	3-slot modular chassis + 2 1000BASE-X GBIC ports on Layer 2 supervisor
Catalyst 4503 + WS-X4014 Supervisor III Engine	Catalyst 4000 IOS	3-slot modular chassis + 2 1000BASE-X GBIC ports on Layer 2/3 supervisor
Catalyst 4503 + WS-X4515 Supervisor IV Engine	Catalyst 4000 IOS	3-slot modular chassis + 2 1000BASE-X GBIC ports on Layer 2/3 supervisor
Catalyst 4506 + WS-X4014 Supervisor III Engine	Catalyst 4000 IOS	6-slot modular chassis + 2 1000BASE-X GBIC ports on Layer 2/3 supervisor
Catalyst 4506 + WS-X4515 Supervisor IV Engine	Catalyst 4000 IOS	6-slot modular chassis + 2 1000BASE-X GBIC ports on Layer 2/3 supervisor
Catalyst 4507R + WS-X4515 Supervisor IV Engine	Catalyst 4000 IOS	7-slot modular chassis + 2 1000BASE-X GBIC ports on Layer 2/3 supervisor
Catalyst 4840G	Catalyst Layer 3 server load-balancing switch	40-port 10/100BASE-TX + 1000BASE-X GBIC Layer 3 server load-balancing switch
Catalyst 4908G-L3	Catalyst G-L3 Switch	12-port 1000BASE-X GBIC Layer 3 switch
Catalyst 4912G	Catalyst 4000 CatOS	12-port 1000BASE-X GBIC Layer 3 switch

[*] GBIC = Gigabit interface converter

This chapter provides information about architecture, supported QoS features, configuration examples, and case studies for both the Catalyst 4000 IOS Family of switches and the Catalyst G-L3 Family of switches.

QoS Support on the Catalyst 4000 IOS Family of Switches

QoS feature support on the Catalyst 4000 IOS Family of switches surpasses those features support Catalyst 4000 CatOS Family of switches. Tables 3-1 through 3-5 in Chapter 3 summarize the gene feature support on the Catalyst 4000 IOS Family of switches.

Specifically, the first section of this chapter covers the following topics and QoS features supported Catalyst 4000 IOS Family of switches:

- Architecture Overview
- Software Requirements
- Global Configuration
- Input Scheduling
- Internal DSCP
- Classification and Marking
- ACL-based Classification
- Policing
- Congestion Management
- Auto-QoS
- Case Study

Catalyst 4000 IOS Family of Switches QoS Architectural Overview

The Catalyst 4000 IOS Family of switches bases packet-processing performance solely on the use c *content addressable memory* (TCAM). Packet processing includes application of QoS rules utilizing These switches also use TCAM for Layer 2 lookups, Layer 3 lookups, Layer 3 rewrite information, a *control list* (ACL) processing. The use of TCAM in this manner allows the Catalyst 4000 IOS Family switches to achieve line-rate performance at 64 Gbps.

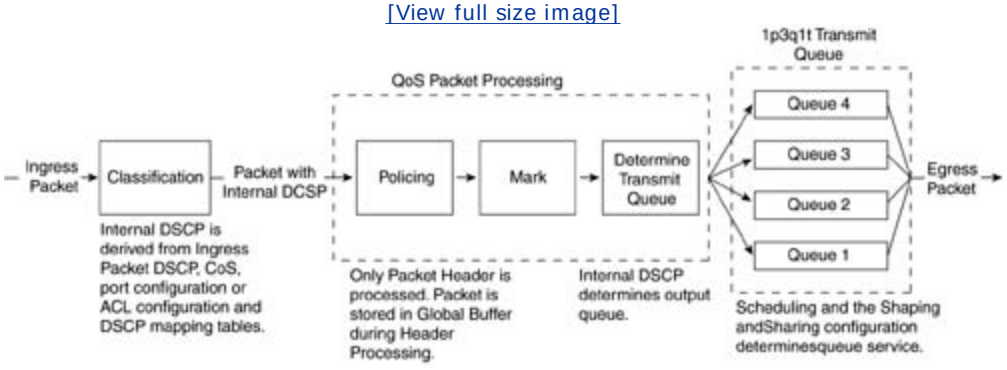
The use of TCAM and the unique architecture of the Catalyst 4000 IOS Family of switches allows fo application of QoS rules while maintaining line-rate performance for nonblocking ports. Table 3-9 i 3 discusses which front-panel ports are nonblocking per line card. The switches' architecture proces packet against all QoS rules regardless of whether any configured QoS rules exist. A default QoS ru exists for application to every packet processed. As a result, when the switch subjects packets to Q and processing, no change occurs in system performance. This behavior holds for ACL processing a

The industry term *hardware switching* refers to the discussed use of the TCAM component for packe processing. If the switch is unable to hardware switch a packet, the CPU must process the packet i The term *software switched* describes the packet processing orchestrated by an onboard CPU. An o CPU employs Cisco IOS Software for instruction on how to process and forward packets. Packets ha the CPU experience lower packet-processing performance compared to hardware-switched packets of the limited speed of the onboard CPU. In relation to QoS on the Catalyst 4000 IOS Family of swi packet processing needs to occur in hardware, because of the limited software-switching performa

CPU. A trade-off of utilizing TCAM for the line-rate performance is that hardware switching does no all the Cisco IOS QoS features available in the latest Cisco IOS Software versions. Hardware switch supports a subset of Cisco IOS QoS features for packet processing. The content of this chapter focu on QoS features that support hardware switching. In addition, TCAM is a limited resource varying p platform. The resource limitations affect the size and number of ACLs entries and policy maps allov that do not fit into TCAM are not executed on the Catalyst 4000 IOS Family of switches. Chapter 6, Features Available on the Catalyst 2950 and 3550 Family of Switches," presents the first platform t TCAM for QoS features. This chapter includes more detailed information on the TCAM architecture. depicts the logical model of QoS packet flow in a Catalyst 4000 IOS switch. Later sections refer to t

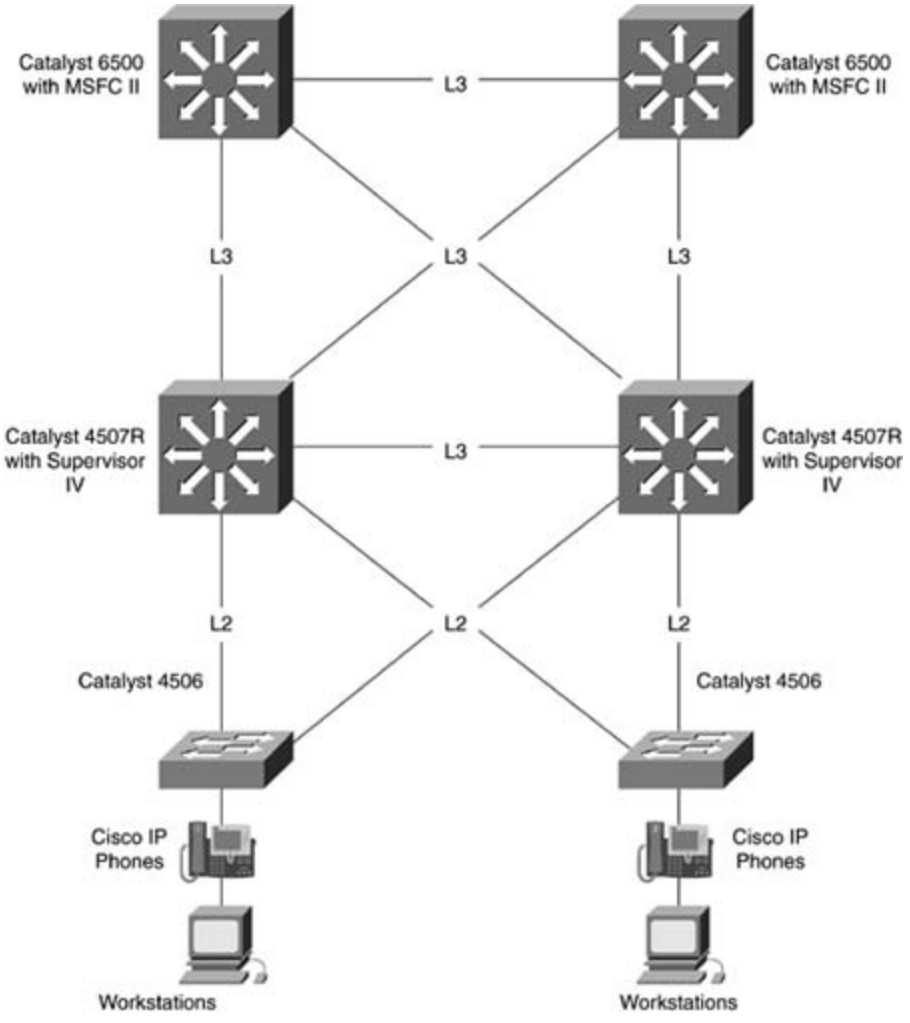
Figure 7-1. Logical QoS Architecture for the Catalyst 4000 IOS Famil Switches

[View full size image]



Because of the QoS features supported on the Catalyst 4000 IOS Family of switches, these switche into an end-to-end QoS design as core, distribution, or access layer switches. Networks that requir than 64 Gbps or 48 Mpps in the core need to use the Catalyst 6500 platform instead. Figure 7-2 ill sample network design using a Catalyst 4000 IOS switch in the distribution layer.

Figure 7-2. Sample Network Topology Using Catalyst 4000 IOS Switc



Software Requirements

All versions of the Catalyst 4000 IOS Software support QoS features. Several QoS attributes differ the existing software versions. The initial releases of the Cisco IOS Catalyst 4000 software are Cisc versions 12.1(8a)EW and 12.1(8a)EW1. A notable software change applicable to QoS occurs in Cis Software version 12.1(11b)EW1. In the original software versions, the switch counts packet drops from transmit queue overflow as output errors. In 12.1(11b)EW1 and later software versions, the s counts output queue buffer drops as output drops rather than output errors. Example 7-1 compare switch counts output queue drops in 12.1(8a)EW and 12.1(8a)EW1 versus 12.1(11b)EW1 and later versions.

Example 7-1. Transmit Buffer Overflow Counter Differences Between Sof Versions 12.1(8a)EW and 12.1(8a)EW1 and Later Versions

```
! 12.1(8a)EW:

Switch#show interface FastEthernet 6/1
```

```
FastEthernet6/1 is up, line protocol is up
  Hardware is Fast Ethernet Port, address is 0007.508b.84e0 (bia 0007.508b.84e0)
  MTU 1500 bytes, BW 100000 Kbit, DLY 100 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  Full-duplex, 100Mb/s
  input flow-control is off, output flow-control is off
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input never, output never, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/2000/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue: 0/40 (size/max)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     373781138 packets input, 2447156352 bytes, 0 no buffer
     Received 11 broadcasts, 0 runts, 0 giants, 0 throttles
     4 input errors, 1 CRC, 0 frame, 0 overrun, 0 ignored
     0 input packets with dribble condition detected
     926574 packets output, 76097292 bytes, 0 underruns
     3120970731 output errors, 0 collisions, 0 interface resets
     0 babbles, 0 late collision, 0 deferred
     1 lost carrier, 0 no carrier
     0 output buffer failures, 0 output buffers swapped out
! 12.1(11b)EW1 and Later:
Switch#show interface FastEthernet 6/1
FastEthernet6/1 is up, line protocol is up
```

```
  Hardware is Fast Ethernet Port, address is 0007.508b.84e0 (bia 0007.508b.84e0)
  MTU 1500 bytes, BW 100000 Kbit, DLY 100 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  Full-duplex, 100Mb/s
  input flow-control is off, output flow-control is off
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input never, output never, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/2000/0/0 (size/max/drops/flushes); Total output drops: 250355824
  Queueing strategy: fifo
  Output queue: 0/40 (size/max)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     373781138 packets input, 2447156352 bytes, 0 no buffer
     Received 11 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
     0 input packets with dribble condition detected
     926574 packets output, 76097292 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 babbles, 0 late collision, 0 deferred
     1 lost carrier, 0 no carrier
     0 output buffer failures, 0 output buffers swapped out
```

Moreover, Cisco IOS Software version 12.1(11b)EW adds support for voice VLANs and inline power
Cisco IOS Software version 12.1(12c)EW adds the ability to disable the IP header *differentiated ser*
(DiffServ)*codepoint* (DSCP) rewrite functionality. The switch rewrites the IP header DSCP by defaul
Disabling the rewrite feature prevents all untrusted ports from rewriting DSCP to zero. In addition,
ignores any port DSCP configuration or marking, because the policing with the IP header DSCP rew

functionality is disabled.

Use the following command to enable and disable the QoS IP header DSCP rewrite functionality:

```
[no]qos rewrite ip dscp
```

Global Configuration

The Catalyst 4000 IOS Family of switches requires QoS to be enabled globally for the switch to exe QoS configurations. To enable QoS globally on the Catalyst 4000 IOS Family of switches, enter the command in the global configuration mode:

```
[no]qos
```

Example 7-2 illustrates a user globally enabling QoS and verifying the configuration on a Catalyst 4 switch.

Example 7-2. Enabling and Viewing QoS Feature Support on a Catalyst 4(Switch

```
Switch#config terminal

Switch(config)#qos

Switch(config)#end

Switch#show qos

QoS is enabled globally
```

```
IP header DSCP rewrite is enabled
```

The global QoS configuration applies to all interfaces by default. However, the Catalyst 4000 IOS F switches supports disabling of QoS on a per-interface basis. The same command, no qos disables per-interface basis. Example 7-3 shows a user disabling QoS feature support on interface FastEthe

Example 7-3. Disabling QoS Feature Configuration on a Per-Interface Bas

```
Switch#config terminal

Switch(config)#interface FastEthernet 1/1

Switch(config)#no qos

Switch(config)#end
```

Input Scheduling

As with the Catalyst 4000 CatOS switches, the Catalyst 4000 IOS Family of switches does not supp scheduling and performs only FIFO queuing of ingress packets. For line-module ports that are nonb FIFO queuing does not pose a significant issue. Nonblocking line-module ports can deliver traffic to switching fabric at line rate. Oversubscribed ports share bandwidth and data transmit contention ir two to eight ports, depending on line module. Chapter 3 in the "Input Scheduling" section discusse scheduling as it relates to the Catalyst 4000 platform in general. Chapter 3 also includes Table 3-9 4000 Line Modules Architecture," which discusses whether line modules are nonblocking and the oversubscription architecture.

Internal DSCP

The Catalyst 4000 IOS Family of switches represents all frames with an internal DSCP value. This i DSCP value characterizes packet priority as the frame traverses the switch. Internal DSCP values r 0 to 63, which is the same as the DiffServ specification for DSCP discussed in RFC 2474 and 2475. processing involves policing, marking, and congestion management of packets using internal DSCP differentiating services.

By default, Cisco IP Phones mark RTP packets with a DSCP value of 46. On trusted ports, the switc the internal DSCP value from the ingress DSCP. Placing these Cisco IP Phones' *Voice over IP* (VoIP) into a strict priority queue illustrates an example of differentiating service using internal DSCP.

The switch must determine internal DSCP values for all ingress packets. Many parameters exist for determining internal DSCP values. These parameters include ingress packet's DSCP or *class of serv* value, trusting configuration, static port DSCP or CoS configuration, and ACL-based configurations.

The following section discusses the available parameters for determining internal DSCP and their r

to the QoS mapping tables.

Classification and Marking

Classification determines the internal DSCP value, which distinguishes packets as they traverse the
The switch uses internal DSCP values to make policing, marking, and output scheduling decisions.
Figure 7-1 for a logical diagram of QoS packet flow.

Classification is the primary focus of this section. For IP and non-IP traffic, several options exist for
classifying frames to a specific DSCP value. The following sections discuss these options.

Trusting DSCP

This section explains trusting DSCP on the Catalyst 4000 IOS Family of switches. For a detailed exp
of the Cisco Catalyst trust concept, consult the Chapter 2, "End-to-End QoS: Quality of Service at L
Layer 2," section "Cisco Catalyst QoS Trust Concept."

The concept of trusting DSCP denotes that the switch uses an ingress frame's IP DSCP value to der
internal DSCP value. For example, frames received on a trusted interface with a DSCP value of 50
denote the internal DSCP value of 50.

Use the following interface command to configure an interface to trust DSCP on ingress packets:

```
qos trust dscp
```

Example 7-4 illustrates a sample configuration of a Catalyst 4000 IOS switch interface configured t
DSCP values on ingress packets.

Example 7-4. Sample Configuration of an Interface Configured for Trustir

```
Switch#show running-config

Building configuration...

(text deleted)

!

qos
```

```
!
(text deleted)
!
interface FastEthernet6/1
 switchport access vlan 2
 qos trust dscp
 no snmp trap link-status
 spanning-tree portfast
!
(text deleted)
!
end
```

For interfaces configured for trusting DSCP, the switch does not rewrite an ingress packet's CoS va
however, the egress CoS value of the final transmitted frame depends on the DSCP-to-CoS mappin
This chapter covers DSCP-to-CoS mapping tables for the Catalyst 4000 IOS Family of switches in t
titled "Mapping Internal DSCP to CoS."

> NOTE
>
> Because the DSCP value includes the IP precedence bits, all configurations use DSCP values ra
> than IP precedence values. A trust IP precedence configuration option does not exist on the Ca
> 4000 IOS Family of switches.

Trusting CoS

For the Catalyst 4000 IOS switch, the notion for trusting CoS signifies that the switch derives the i
DSCP value from the CoS value of an ingress frame. The CoS-to-DSCP mapping determines exactly
CoS value maps to an internal DSCP value. For example, when trusting CoS, an ingress CoS value
to an internal DSCP value of 40 with the default CoS-to-DSCP mapping table.

In addition, the switch does not rewrite the DSCP value of the packet on ingress when configured f
CoS. The packet's DSCP is not rewritten by the switch on ingress since the internal DSCP value ser
classification parameter during packet-processing. Consequently, the egress DSCP value of the fina
transmitted frame depends solely on the internal DSCP. A multitude of factors determine the interr
including classification, marking, policing, and mapping tables. For example, a switch configured fo

CoS that receives frames on an interface with a DSCP value of 26 and a CoS value of 5, uses only t value of 5 to derive the internal DSCP. Assuming default CoS to DSCP mapping, the switches calcu internal DSCP as a value of 40. Without any marking occurring and assuming default DSCP to CoS the DSCP value and CoS value of the egress frame will be 40 and 5, respectively.

Use the following interface command to configure an interface to trust CoS on ingress packets:

```
qos trust cos
```

Example 7-5 illustrates a Catalyst 4000 IOS interface configured to trust CoS values for internal DS mapping on ingress packets.

Example 7-5. Sample Configuration of an Interface Configured for Trustir

```
Switch#show running-config
Building configuration...
(text deleted)
!
qos
!
(text deleted)
!
interface FastEthernet6/1
 switchport access vlan 2
 qos trust cos
 no snmp trap link-status
 spanning-tree portfast
!
```

```
(text deleted)

!

end
```

Table 7-2 indicates the default CoS-to-DSCP mapping that occurs on ingress frames on an interfac configured for trusting CoS.

Table 7-2. Default CoS-to-Internal DSCP Mapping Table

CoS Value	0	1	2	3	4	5	6	
DSCPValue	0	8	16	24	32	40	48	

The CoS-to-DSCP mapping table is configurable such that any ingress packet's CoS value may map internal DSCP value. Use the following command to configure the CoS-to-DSCP mapping table:

qos map cos*cos-list to dscp*

cos-list represents the CoS values to map to the configured DSCP value. The command accepts the entries separated by a space.

For displaying the current configured CoS-to-DSCP mapping, use the following command:

show qos maps cos dscp

Example 7-6 illustrates an example of a user displaying, configuring, and verifying the CoS-to-DSC mapping table.

Example 7-6. Displaying, Configuring, and Verifying the CoS-to-DSCP Ma Table

```
Switch#show qos maps cos dscp

CoS-DSCP Mapping Table

   CoS:   0  1  2  3  4  5  6  7

--------------------------------

  DSCP:   0  8 16 24 32 40 48 56

Switch#configure terminal

Switch(config)#qos map cos 1 2 3 to dscp 35

Switch(config)#qos map cos 4 5 6 to dscp 45

Switch(config)#end

Switch#show qos maps cos dscp

CoS-DSCP Mapping Table

   CoS:   0  1  2  3  4  5  6  7

--------------------------------

  DSCP:   0 35 35 35 45 45 45 56
```

To restore the CoS-to-DSCP mapping table back to the default configuration, use the following glol configuration command:

```
no qos map cos to dscp
```

Untrusted Interfaces

The term *untrusted port* refers to an interface that does not utilize the DSCP or CoS value of an ing
for determining the internal DSCP value. Unlike the default behavior of the Catalyst 4000 CatOS sv
where all ports are trusted, the default behavior for all interfaces of a Catalyst 4000 IOS switch is t
DSCP or CoS (that is, untrusted). As a result, the switch classifies the frame with an internal DSCP
zero by default. The terms *portDSCPconfiguration* and *port CoS configuration* identify untrusted po
well. Unless the switch changes the DSCP value of the frames by policing or marking, the egress va
zero.

Use the following commands to configure a switch to override an untrusted interface with specific D
CoS values:

qos dscp*dscp_value*

qos cos*cos_value*

dscp_value and *cos_value* symbolize the DSCP or CoS value used in determining the internal DSCP

Example 7-7 illustrates a sample configuration of an interface configured for reclassifying the DSCP
40.

Example 7-7. Sample configuration of an Interface Configured for Reclas
DSCP

```
Switch#show running-config
Building configuration...
(text deleted)
!
interface FastEthernet6/1
 switchport access vlan 2
 qos dscp 40
 no snmp trap link-status
```

```
 spanning-tree portfast
!
(text deleted)
!
end
```

Non-IP Frames

IP frames contain bits for IP precedence and DSCP; non-IP frames do not include an IP header or b
precedence or DSCP. The Catalyst 4000 IOS switch prioritizes packets based only on DSCP or CoS.
result, the switch classifies non-IP frames exclusively based on ingress CoS values or port CoS con
Trusting CoS is a valid configuration for non-IP frames. Port CoS configuration treats a port as unt
rewrites CoS values to a specified value. By default, the switch reclassifies the CoS value to zero. B
internal DSCP value does not require an IP header, the switch subsequently maps the configured o
CoS value to an internal DSCP value using the configurable CoS-to-DSCP mapping table. The prece
classification sections—"Trusting CoS" and "Untrusted Interfaces"—include examples for displaying
configuring, and verifying trusting CoS configurations and untrusted configurations.

Displaying Port Trust Configuration

To verify an interface trust configuration, use the following command:

```
show qos interface {{FastEthernetinterface-number} | {GigabitEthernetinterface-num
➥{VLANvlan_id} | {Port-channelnumber}}
```

Example 7-8 displays a sample output of the show qos interface*interface_name* command. This e
illustrates an interface configured for trusting DSCP. The output displays both the global and interf
configuration. In addition, the command displays the trust port state as either DSCP, CoS, or untru
the respective reclassification values for the untrusted configuration.

Example 7-8. Sample Output of the show qos interface Command

```
Switch#show qos interface GigabitEthernet 1/1

QoS is enabled globally

Port QoS is enabled

Port Trust State: 'DSCP'

Default DSCP: 0 Default CoS: 0

Appliance trust: none

Tx-Queue    Bandwidth    ShapeRate    Priority    QueueSize

              (bps)        (bps)                 (packets)

  1         250000000    disabled     N/A         1920

  2         250000000    disabled     N/A         1920

  3         250000000    disabled     normal      1920

  4         250000000    disabled     N/A         1920
```

The appliance trust field indicates the 802.1p trust configuration to communicate via *Cisco Discove* (CDP) to a neighbor appliance. *Appliance trust* is the IOS term for extended trust, as discussed in C The "Output Scheduling" section of this chapter discusses the transmit queues, bandwidth, and sha output.

ACL-Based Classification

Because the ultimate goal of classification is to determine the marking and scheduling of the frame methods of classification exist. The Catalyst 4000 IOS Family of switches allows for classification of packets based on standard, extended, and named IP ACLs in addition to the port trust configuratio addition, the switches support classification occurring strictly on IP precedence values or DSCP valu without the use of an ACL. ACL-based classification often inherits properties of the untrusted config because the switch does not use the packet's DSCP or CoS values for determining packet classificat

Chapter 5, "Introduction to the Modular QoS Command-Line Interface," elaborates on how to creat implement class maps and policy maps and includes examples that show you how to do so. Consul 5 before reading the following "Class Map" and "Policy Map" sections. The following sections discus maps and policy maps, with a focus on options supported by hardware switching.

Class Maps

As with mainline Cisco IOS Software, class maps label ACLs for application on policy maps. A limite of available Cisco IOS class map match commands is available for hardware switching on the Cata IOS Family of switches. The following class map criteria are available for hardware switching:

- Standard, extended, or named IP or MAC ACLs
- IP precedence values
- DSCP values
- All packets MAC-address based ACL is also supported

To create class map clauses for ACL-based classification, use the following class map configuration command:

```
match access-group {acl_index | nameacl_name}
```

acl_index represents the ACL number and *acl_name* refers to a named ACL. Up to eight class clause configurable per class map. Example 7-9 illustrates a user configuring a class map for matching pa against either of two ACLs.

Example 7-9. User Configuring a Class Map Matching One of Two Configu ACLs

```
Switch#configure terminal

Enter configuration commands, one per line.  End with CNTL/Z.

Switch(config)#access-list 110 permit udp any host 10.1.1.2 eq 12000

Switch(config)#access-list 111 permit tcp any host 192.168.100.1 eq 50000

Switch(config)#class-map match-any TEST

Switch(config-cmap)#match access-group 110

Switch(config-cmap)#match access-group 111

Switch(config-cmap)#end
```

To configure IP precedence and DSCP value matching criteria, use the following class map commar respectively:

```
match ip precedence ipp_value1 [ipp_value2 [ipp_valueN]]

match ip dscp dscp_value1 [dscp_value2 [dscp_valueN]]
```

To match against IP precedence or DSCP, you must use the qos trust dscp configuration on the in
Otherwise, the switch uses the default or port DSCP configuration value.

The match-any class map configuration command configures the class map to match packets on ar
Moreover, the match-all class map configuration command option configures the class map to ma
packets against all ACLs.

Example 7-10 shows a class map configured to match packets against several DSCP values.

Example 7-10. Class Map Configured to Match Against DSCP Values

```
Switch#configure terminal

Enter configuration commands, one per line.  End with CNTL/Z.

Switch(config)#class-map match-all MATCH_DSCP_VALUES_4-7

Switch(config)#match ip dscp 50 51 52 53 54 55

Switch(config-cmap)#end
```

For additional information about creating and applying class maps, see Chapter 5.

Policy Maps

Class maps define classification criteria; policy maps organize the class map classification criteria v
policing and marking actions. Policing and marking configuration derive the class actions. Do not c
policing with policy maps. Whereas policy maps may include an action based on rates, they also in
support for other QoS actions such as marking.

On the Catalyst 4000 IOS Family of switches, policy maps tie up to eight class map actions togethe
map action may consist of a traffic-limiting policer. Each interface on the Catalyst 4000 IOS Family
switches supports a single policy for ingress traffic and a single policy for egress traffic.

Example 7-11 illustrates a sample policy map configuration for marking packets based on UDP por
action shown, set ip precedence 5, rewrites all ingress packets received on interface FastEtherne
matching the ACL 101 criteria.

Example 7-11. Sample Policy Map and Class Map Configuration

```
Switch#show running-config
Building configuration...
(text deleted)
!
qos
!
(text deleted)
!
class-map match-all UDP_PORT_50000_59000
  description MATCH PACKETS ON UDP PORTS 50000 TO 59000
  match access-group 101
!
policy-map TEST
 description MARK FRAMES ON UDP PORT 50000 TO 59000 WITH IP PRECEDENCE 5
  class UDP_PORT_50000_59000
    set ip precedence 5


!
interface FastEthernet2/1
 service-policy input TEST
 no snmp trap link-status
 spanning-tree portfast
(text deleted)
!
```

```
(text deleted)
!
access-list 101 permit udp any any range 50000 59000
!
(text deleted)
!
end
```

To view ingress packet matching against the class map clauses of a policy map, use the following c

show policy-map interface [{**FastEthernet***interface-number*} | {**GigabitEthernet**
interface-number} | {**Port-channel***number*} | {**VLAN***vlan_id*}] [**input** | **output**]

Example 7-12 shows the sample output of the show policy-map interface command for the polic sample configuration in Example 7-11.

Example 7-12. Viewing Class Map Matches of a Policy Map

```
Switch#show policy-map interface FastEthernet6/1
 FastEthernet6/1
  service-policy input: TEST
    class-map: UDP_PORT_50000_59000 (match-all)
      9628 packets
      match: access-group 101
```

```
    set:
      ip precedence 5
  class-map: class-default (match-any)
    3948 packets
    match: any
      3948 packets
```

Policing

The Catalyst 4000 IOS Family of switches supports individual and aggregate policing in both ingres egress configurations. In brief, aggregate policing limits a shared rate and burst parameter among associated ports or VLANs. With individual policing, each port or VLAN uses its own exclusive rate a parameters. This section discusses the following policing topics:

- Policing Resources
- Port-based and VLAN-based Policing
- Individual and Aggregate Policing
- Policing Actions
- Traffic-Rate Policing
- Leaky Token Bucket Algorithm
- Burst Size Parameter
- Guaranteed Rate of Policer
- Policing Accuracy
- DSCP Policed Action
- Marking Action
- Trusting Action

Policing Resources

The architecture of the Catalyst 4000 IOS Family of switches limits the number of input policers an policers to 1024. The architecture processes all packets for policing regardless of configuration. As the Cisco Catalyst 4000 IOS reserves two input and two output policers for null processing. Subseq 1022 policers for input policing and 1022 policers for output policing are available.

Port-Based and VLAN-Based Policing

Port-based policing entails binding policy maps to individual ports. VLAN-based policing involves at policy map to a VLAN interface.

For a configuration in which a VLAN-based policy exists but a port-based policy does not exist, the subjects ingress packets to the VLAN-based policer. This behavior occurs regardless of the port's po configuration for VLAN-based or port-based policing. To work around this behavior, create and atta policy map to the interface. A null policy map consists of a defined police map with a class map cla match all packets and no configured actions. The use of this null policer to work around VLAN-base behavior applies to both ingress and egress policers.

To configure an interface to use VLAN-based policies and to fall back on port-based policies if no VL policies exist, use the following interface command:

```
qos vlan-based
```

Example 7-13 shows and interface configured for VLAN-based QoS.

Example 7-13. An Interface Configured for VLAN-Based Policing

```
Switch#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Switch(config)#interface fastethernet 2/1
Switch(config-if)#qos vlan-based
Switch(config-if)#end
```

Individual and Aggregate Policing

Individual policers apply bandwidth limits discretely to each interface for defined policy maps. Clas define traffic classes within a policy map. Use the following policy map class clause configuration co configure individual policers:

```
police rate burst [[conform-action {transmit | drop}]  [exceed-action {transmit | d
➥oliced-dscp-transmit}]]
```

Example 7-14 shows a sample configuration of an individual policer. In this example, the policer li
ingress traffic from both FastEthernet 6/1 and 6/2 separately to the 1.54 Mbps specified in the indi
policer.

Subsequent sections cover configuring the policer rate, burst, and actions parameters shown in Exa
14 and 7-15.

Example 7-14. Sample Configuration of an Individual Policer

```
Current configuration : 327064 bytes
!
(text deleted)
qos
(text deleted)
!
(text deleted)
!
class-map match-any UDP_PORT_1200
  description MATCH PACKETS ON UDP PORT 1200
  match access-group 102
!
policy-map TEST_INDIVIDUAL
  class UDP_PORT_1200
    police 1.54 mbps 16000 byte conform-action transmit exceed-action drop
```

```
!
(text deleted)
!
interface FastEthernet6/1
 switchport access vlan 2
 service-policy input TEST_INDIVIDUAL
 no snmp trap link-status
!
interface FastEthernet6/2
 switchport access vlan 2
 service-policy input TEST_INDIVIDUAL
 no snmp trap link-status
!
access-list 102 permit udp any any eq 1200
!
(text deleted)
!
end
```

Aggregate policers cumulatively restrict bandwidth among all ports and VLANs per class map claus policy map. Use the following global configuration command to configure aggregate policers:

qos aggregate-policer*policer_name rate burst* [[**conform-action** {**transmit** | **drop**}]

  [**exceed-action** {**transmit** | **drop** | **policed-dscp-transmit**}]]

policer_name defines the name to represent the aggregate policer. To attach the aggregate policer following policy-map class clause configuration command:

police aggregate*policer_name*

Example 7-15 provides a sample configuration of an aggregate policer. In this example, the policer ingress traffic from both FastEthernet 6/1 and 6/2 cumulatively to the specified 1.54-Mbps rate spe the aggregate policer.

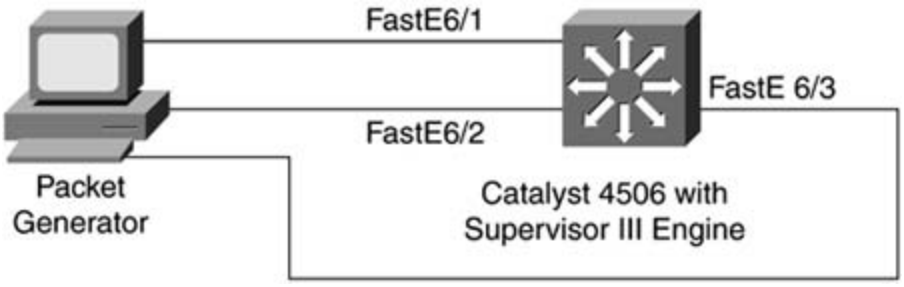
Example 7-15. Sample Configuration of an Aggregate Policer

```
Current configuration : 327064 bytes
!
(text deleted)
!
qos aggregate-policer 1_MBPS_RATE_EXCEED_DROP 1540000 bps 8000 byte conform-action
➥transmit exceed-action drop
qos
(text deleted)
!
interface FastEthernet6/1
 switchport access vlan 2
 service-policy input TEST_AGGREGATE
 no snmp trap link-status
!
interface FastEthernet6/2
```

```
 switchport access vlan 2
 service-policy input TEST_AGGREGATE
 no snmp trap link-status
!
(text deleted)
!
class-map match-any UDP_PORT_1200
  description MATCH PACKETS ON UDP PORT 1200
  match access-group 102
!
policy-map TEST_AGGREGATE
  class UDP_PORT_1200
    police aggregate 1_MBPS_RATE_EXCEED_DROP
!
(text deleted)
!
access-list 102 permit udp any any eq 1200
!
(text deleted)
!
end
```

To demonstrate and measure this behavior, two packet generator ports were connected to the swit ports FastEthernet 6/1 and 6/2 as shown in Figure 7-3. The packet generator ports sent traffic at 1 unicast to a destination connected off interface FastEthernet 6/3. A packet generator port connecte FastEthernet 6/3 measured the traffic rate for this flow.

Figure 7-3. Topology for Demonstrating Individual and Aggregate Pol



Three trials were conducted. For each trial, the packet count and traffic rate were measured. The tı included measuring traffic against a configuration without a policy, using the individual policer sho Example 7-14, and using the aggregate policer shown in Example 7-15.Table 7-3 displays the resu

Table 7-3. Demonstration of Individual Versus Aggregate Policer

Trial	No Policers	Individual Policer Shown in Example 7-15	Aggregate Policer Sh Example 7-16
Rate Received on FastEthernet 6/3	100 Mbps	3.08 Mbps	1.5 Mbps

The trial results clearly indicate that the individual policers apply traffic rates respective to each po whereas aggregate policers apply traffic rates as the sum of all applicable ports.

Policing Actions

The Catalyst 4000 IOS Family of switches supports the following policing actions:

- Traffic-rate policing
- Marking IP DSCP
- Marking IP precedence
- Trusting DSCP
- Trusting CoS

The Catalyst 4000 IOS Family of switches supports transmitting or dropping the packet for both th conforming and exceeding actions of a traffic-rate policer. In addition, the exceeding action is confi for marking down the DSCP on the respective packet. With regard to the Catalyst 4000 IOS Family switches, a policed-dscp-transmit action refers to the marking down of a packet.

Traffic-Rate Policing

The packet-processing step of the QoS model from Figure 7-1 is responsible for determining wheth

processed packet is conforming to or exceeding a specified rate of a policer. The Catalyst 4000 IOS switches uses the leaky token bucket algorithm to determine whether a packet is conforming to or a specified policer rate.

The following commands configure the *rate* and *burst* of a class map policer and an aggregate poli respectively:

```
police rate burst [[conform-action {transmit | drop}] [exceed-action {transmit | dr
  policed-dscp-transmit}]]
qos aggregate-policer policer_name rate burst [[conform-action {transmit | drop}]
  [exceed-action {transmit | drop | policed-dscp-transmit}]]
```

rate defines the actual policing rate. The supported rates for the Catalyst 4000 IOS Family of switc from 32 kbps to 32 Gbps in 1-bps increments. When entering the value for rate, the prefixes kilo, r giga are optional using the k, m, and g command, respectively.

burst defines the burst size in bytes. The burst size needs to be at least the maximum packet size c touched by the policer for accurate policing. The burst size parameter also supports the following o prefixes: kilo, mega, and giga. The following sections discuss determining the applicable burst size

Leaky Token Bucket Algorithm

The switch applies the rate and burst parameters defined in the policer to all packets processed. Th token bucket algorithm takes snapshots of packet flow to determine whether a packet is conformin exceeding the specified rate. The configured policer rate defines the number of tokens removed at interval. The leaky token bucket algorithm does not actually store, process, or buffer any packet. R switch architecture relies on the leaky token bucket algorithm to distinguish a packet in a flow as c or exceeding the configured policer rate. The switch architecture uses the algorithm for both ingres egress policing.

The number of tokens entering the bucket correlates to the ingress packet size of a frame. A specif of tokens leak from the bucket every 16 nanoseconds for the Catalyst 4000 IOS Family of switches. number of tokens leaking from the bucket relates to the configured policer rate. The burst size defi maximum number of bytes in the bucket at any interval.

When the token bucket cannot accommodate any new tokens, the algorithm determines that the pa corresponding to these tokens are exceeding the policer rate and the packet processing engine exe exceed action of the policer.

In brief, the token bucket algorithm provides a logical, visual portrait of how policing limits traffic. portrait defined becomes useful when considering the burst size and the effect of the token bucket

on TCP and UDP protocols.

> NOTE
>
> Buffering does not occur on the packets in the bucket. The Leaky Token Bucket Algorithm only snapshots traffic to determine whether a packet is conforming or exceeding the configured pol rate.

Burst Size Parameter

Because of the flow-control nature of TCP/IP and UDP application behavior, packet drops significan traffic behavior and may result in a packet-per-second performance far below the configured police burst parameter of policing attempts to handle the torrent nature of TCP/IP and web traffic by allo period surges of traffic into the bucket.

Configuration of the burst size follows several other Catalyst platform recommendations. For TCP applications, use the following formula for calculating the burst size parameter used for policing:

<burst> = 2 * <RTT> * <rate>

RTT defines the approximate *round-trip time* for a TCP session. If RTT is unknown, use a RTT value millisecond or 1 second depending on estimated latency. Example 7-16 illustrates the burst calcula rate of 64 kbps and an unknown RTT.

Example 7-16. Sample Burst Calculation

```
<burst> = 2 * <1 sec> * <64000 bits/sec>

<burst> = 128000 bits = 16000 bytes
```

Nevertheless, from an application perspective, TCP/IP traffic-rate policing always results in actual r than the configured rate regardless of the burst size. UDP applications police closer to the configur pure bits per second; however, some UDP applications retransmit heavily upon packet loss. As a re rate for applications that use UDP may also fall well below the configured rate. In summary, syster administrators must carefully plan and consider application behavior and resiliency to rate policing applying policers.

Guaranteed Rate of Policer

During any time interval, the leaky token bucket algorithm implementation on the Catalyst 4000 IO of switches guarantees the following conforming policing rate:

Conforming rate <= (<configured_rate bits/sec> * <1 byte/8 bits> * <period>) + <burst_si

packet

Consider Example 7-14, for example, where the aggregate policer defines a rate of 1.54 Mbps and a size of 8000 bytes. Using the preceding formula, the guaranteed conforming traffic rate for a 1-sec interval calculates as follows:

Conforming rate <= (1540000 bits/sec * 1 byte/8 bits * 1 sec) + 8000 bytes + 1 packet = bytes

In bits per second, the conforming rate is 2.006 Mbps, assuming an average 100-byte packet size.

Because of the nature of applications that use the TCP/IP and UDP/IP protocols, use careful plannir configuring the burst size.

Policing Accuracy

Moreover, the architecture of the Catalyst 4000 IOS Family of switches bestows policing at a finite rates distributed between 32 kbps and 32 Gbps in 1-bps increments. Because of the hardware arch specified rates adjust up or down to the nearest hardware-capable rate. The adjusted policy rate is within 1.5 percent of the configured rate. Subsequently, two distinct policy rates must differ by at l percent.

DSCP-Policed Action

The Catalyst 4000 IOS Family of switches uses the DSCP-policed concept to mark down packets. Fo policed traffic, the switch marks the frame with a DSCP value derived from the QoS DSCP-policed r table. The default mapping table maps 1:1 with the internal DSCP values; therefore, the default m action results in no change to the DSCP value. The QoS DSCP-policed mapping table requires a nor configuration to actually mark down packets that exceed the rate specified in the policing action.

Use the following command to configure the QoS DSCP-policed mapping tables:

```
qos map dscp policed dscp-list to dscp mark-down-dscp
```

dscp-list represents up to eight DSCP values that configure to represent the DSCP *mark-down-DSC* Example 7-17 displays the default QoS DSCP-policed mapping table, configures the DSCP-policed r table for marking down DSCP values 50 to 59 to 0, and verifies the configuration.

Example 7-17. Displaying, Configuring, and Verifying the QoS DSCP-Polic Mapping Table

```
Switch#show qos map dscp policed

Policed DSCP Mapping Table (DSCP = d1d2)

d1 : d2  0  1  2  3  4  5  6  7  8  9

------------------------------------

 0 :    00 01 02 03 04 05 06 07 08 09

 1 :    10 11 12 13 14 15 16 17 18 19

 2 :    20 21 22 23 24 25 26 27 28 29

 3 :    30 31 32 33 34 35 36 37 38 39

 4 :    40 41 42 43 44 45 46 47 48 49

 5 :    50 51 52 53 54 55 56 57 58 59

 6 :    60 61 62 63

Switch(config)#qos map dscp policed 50 51 52 53 54 55 56 57 to dscp 0

Switch(config)#qos map dscp policed 58 59 to dscp 0

Switch#show qos map dscp policed

Policed DSCP Mapping Table (DSCP = d1d2)

d1 : d2  0  1  2  3  4  5  6  7  8  9

------------------------------------

 0 :    00 01 02 03 04 05 06 07 08 09

 1 :    10 11 12 13 14 15 16 17 18 19

 2 :    20 21 22 23 24 25 26 27 28 29

 3 :    30 31 32 33 34 35 36 37 38 39

 4 :    40 41 42 43 44 45 46 47 48 49

 5 :    00 00 00 00 00 00 00 00 00 00

 6 :    60 61 62 63
```

To configure the traffic-rate policer for marking down the DSCP value for the exceed action, use th dscp-transmit keyword for the exceed-action parameter for the following individual policy map aggregate global policer configuration commands:

```
police rate burst [[conform-action {transmit | drop}] [exceed-action {transmit | dr
  policed-dscp-transmit}]]
qos aggregate-policer policer_name rate burst [[conform-action {transmit | drop}]
  [exceed-action {transmit | drop | policed-dscp-transmit}]]
```

Example 7-18. Sample Configuration for policed-dscp-transmit Exceed Ac an Individual Policer

```
Current configuration : 327064 bytes
!
(text deleted)
!
qos map dscp policed 50 51 52 53 54 55 56 57 to dscp 7
qos map dscp policed 58 59 to dscp 7
qos
(text deleted)
class-map match-all UDP_PORT_10000
  description MATCH PACKETS ON DESTINATION UDP PORT 10000
  match access-group 105
!
policy-map MARK_BASED_ON_RATE
  class UDP_PORT_10000
    police 32000 bps 16000 byte conform-action transmit exceed-action policed-dscp
!
```

```
interface FastEthernet6/1
 switchport mode access
 service-policy input MARK_BASED_ON_RATE
 spanning-tree portfast
!
(text deleted)
!
access-list 105 permit udp any any eq 10000
!
end
```

Marking Action

Policy maps allow for marking of packets using ACL-based classification. Class maps frame ACLs fo maps. Review the "ACL-Based Classification" section earlier in this chapter for discussion of class m ACL options.

Policy maps organize the marking action using the following policy map class clause command:

```
set ip [dscp | precedence] [value]
```

value represents the actual value to mark on the packet for DSCP or IP precedence. Example 7-19 marking based on ACL-based classification.

Example 7-19. Sample Configuration of Marking Based on ACL Classificat

```
Current configuration : 327064 bytes
!
```

```
(text deleted)
!
qos
!
(text deleted)
!
class-map match-all UDP_PORT_10000
  description MATCH PACKETS ON DESTINATION UDP PORT 10000
  match access-group 105
!
!
policy-map ACL_MARK
  class UDP_PORT_10000
   set ip dscp 40
!
!
interface FastEthernet6/1
 switchport mode access
 service-policy input ACL_MARK
 spanning-tree portfast
!
(text deleted)
access-list 105 permit udp any any eq 10000
(text deleted)
!
end
```

Trusting Action

Trusting DSCP or CoS using a policing action is another way to refer to trusting DSCP or CoS based ACL. Packets that match the configured class clause have the internal DSCP determined based on i packets' DSCP or CoS value. The switch does not alter the internal DSCP of frames that do match t clause ACLs. For configurations using trusting in class map clauses, there is no need for a trusting configuration on the interface.

Configuring an interface for trusting and configuring a policing action of trusting needs careful cons because a trusting configuration on an interface classifies ingress frames before a policy.

Policy maps organize the trusting actions using the following policy map class clause command:

```
trust [dscp | cos]
```

Example 7-20 illustrates a policy map configured to trust DSCP for a specific class map.

Example 7-20. Sample Configuration for a Policy Map Configured to Trust

```
Current configuration : 327064 bytes
!
(text deleted)
!
qos
!
(text deleted)
!
interface FastEthernet3/1
 switchport access vlan 2
 switchport voice vlan 700
 service-policy input TRUST_UDP_GT_10000
```

```
 no snmp trap link-status
 tx-queue 3
   priority high
 spanning-tree portfast
!
(text deleted)
!
class-map match-all UDP_PORT_GT_10000
  match access-group 150
!
!
policy-map TRUST_UDP_GT_10000
  class UDP_PORT_GT_10000
    trust dscp
!
(text deleted)
access-list 150 permit udp any any gt 10000
(text deleted)
end
```

Congestion Management

After the switch classifies and processes packets against QoS policies, the switch places the packet transmit queues for output scheduling. Refer to Figure 7-1 for a logical diagram of this behavior.

The Catalyst 4000 IOS Family of switches supports congestion management through the use these and scheduling mechanisms that occur after packet processing. Classification distinguishes packets multiple egress queues, whereas scheduling differentiates service by transmitting packets out of th queues in a specific order.

Queuing and scheduling use the following QoS-specific features for building the congestion manage model:

- Port Transmit Queues
- Mapping Internal DSCP to Transmit Queues
- Strict-Priority Queuing
- Sharing
- Shaping
- Mapping Internal DSCP to CoS

Port Transmit Queues

The Catalyst 4000 IOS Family of switches uses a shared memory architecture. The shared memory architecture handles all output queuing and scheduling on the supervisor engine versus other Catal platforms that depend on line-module architecture for output queuing and scheduling. As a result, utilize four transmit egress queues labeled 1 to 4. Gigabit Ethernet interfaces employ a queue size packets, whereas Fast Ethernet and nonblocking Gigabit Ethernet interfaces utilize a queue size of packets.Table 3-9 in Chapter 3 discusses which line-module interfaces are nonblocking. Software v available at the time of publication do not allow for configuration of the queue size.

Mapping Internal DSCP to Transmit Queues

After packet processing occurs, the switch places the packet into a transmit queue for scheduling. E the switch places packets with higher DSCP values into higher-numbered transmit queues. Neverth switch services all transmit queues round-robin by default. The strict-priority queue, shaping, and configurations allow for differentiating service based on the transmit queue. Table 7-4 lists the defa internal DSCP to transmit queue mapping.

Table 7-4. Default Internal DSCP-to-Transmit Queue Mapping Tabl

DSCPValues	Transmit Queue
0–15	0
16–31	1
32–47	2
48–63	3

Use the following command to display the internal DSCP–to–transmit queue mapping:

show qos maps dscp tx-queue

Use the following global configuration command to configure the internal DSCP to the transmit que

[no]qos map dscp*dscp_values***to tx-queue***queue-id*

dscp_values represents configuration for up to eight DSCP values to map to a transmit queue. *queu* represents one of the four transmit queues. For configuring mapping of more than eight DSCP valu multiple commands.

Example 7-21 displays the default DSCP-TxQueue mapping table, configures the DSCP-TxQueue m table for assigning DSCP values 40 to 49 to queue 1, and verifies the DSCP-TxQueue mapping tabl configuration.

Example 7-21. Displaying, Configuring, and Verifying the DSCP-to-Transı Queue Mapping Table

```
Switch#show qos maps dscp tx-queue

DSCP-TxQueue Mapping Table (DSCP = d1d2)

d1 : d2  0  1  2  3  4  5  6  7  8  9

-------------------------------------

 0 :    01 01 01 01 01 01 01 01 01 01

 1 :    01 01 01 01 01 01 02 02 02 02

 2 :    02 02 02 02 02 02 02 02 02 02

 3 :    02 02 03 03 03 03 03 03 03 03

 4 :    03 03 03 03 03 03 03 03 04 04

 5 :    04 04 04 04 04 04 04 04 04 04
```

```
 6 :    04 04 04 04

Switch#configure terminal

Enter configuration commands, one per line.  End with CNTL/Z.

Switch(config)#qos map dscp 40 41 42 43 44 45 46 47 to tx-queue 1

Switch(config)#qos map dscp 48 49  to tx-queue 1

Switch(config)#end

Switch#show qos maps dscp tx-queue

DSCP-TxQueue Mapping Table (DSCP = d1d2)

d1 : d2  0  1  2  3  4  5  6  7  8  9

---------------------------------------

 0 :    01 01 01 01 01 01 01 01 01 01

 1 :    01 01 01 01 01 01 02 02 02 02

 2 :    02 02 02 02 02 02 02 02 02 02

 3 :    02 02 03 03 03 03 03 03 03 03

 4 :    01 01 01 01 01 01 01 01 01 01

 5 :    04 04 04 04 04 04 04 04 04 04

 6 :    04 04 04 04
```

Strict-Priority Queuing

The Catalyst 4000 IOS Family of switches offers strict-priority, Low-Latency Queuing by designatin queue 3 as a high-priority transmit queue. By assigning transmit queue 3 as a high-priority transm the switch transmits packets out of transmit queue 3 before any other queue until the queue reach share rate. The next section, "Sharing," discusses share rate.

The recommended use of the high-priority queue is for time-sensitive packet flows, such as VoIP fl tickers, and, in some cases, video conferencing. Because of the aggressive scheduling nature of the priority, strict-priority queues may starve lower-priority queues when the high-priority queue cons contains packets to transmit.

Use the following interface transmit queue command to configure transmit queue 3 as a high-prior

```
priority high
```

Example 7-22 shows a sample configuration of an interface configured with a strict-priority queue.

Example 7-22. Sample Configuration of Interface Configured for Strict-Pr Queuing

```
Current configuration : 327064 bytes
!
(text deleted)
qos
!
(text deleted)
!
interface FastEthernet3/1
 switchport access vlan 2
 switchport voice vlan 700
 no snmp trap link-status
 tx-queue 3
   priority high
 spanning-tree portfast
!
(text deleted)
!
end
```

Sharing

The Catalyst 4000 IOS Family of switches supports QoS bandwidth sharing per transmit queue on nonblocking Gigabit Ethernet interfaces. Table 3-9 in Chapter 3 discusses which Gigabit Ethernet li modules and ports are nonblocking.

Because different DSCP and CoS values map to different queues, sharing differentiates services by guaranteeing queue bandwidth. Bandwidth sharing forces minimum bandwidth per transmit queue practical example of using sharing is with high-bandwidth applications such a *Network File System* Generally, applications using NFS require high bandwidth with minimal packet loss. Configuring sh 500 Mbps for transmit queue 2 and mapping NFS packets to transmit queue 2 forces the switch to 500 Mbps of egress traffic to the adjacent switch for NFS packets. In this configuration, the system administrator must adjust other transmit queues' bandwidth rates to compensate for the 500 Mbps for transmit queue 2 given that the total bandwidth of the egress interface is limited to 1 Gbps. Exa provides a sample configuration for the described application of sharing.

Example 7-23. Sample Configuration of Shaping

```
Building configuration...

Current configuration : 225 bytes

!

qos

!

(text deleted)

!

interface GigabitEthernet1/1

 switchport trunk encapsulation dot1q

 switchport mode trunk

 no snmp trap link-status

 tx-queue 1

   bandwidth 125 mbps

 tx-queue 2

   bandwidth 500 mbps

 tx-queue 3

   bandwidth 125 mbps
```

```
!

(text deleted)

!

end
```

The switch maintains sharing rates by treating all queues as high-priority queues during periods w transmit queue's egress traffic rate is below the configured share values. Initially, the switch round packets until a queue reaches its share. At this instance, the switch services round-robin all other c under the defined share. The switch still handles a configured strict-priority queue as defined in the Priority Queuing" section. The switch services the strict-priority queue before all other queues until its configured share value. When all queues reach their share rate, the switch services the queues robin. The default bandwidth parameter applied to each transmit queue is 250 Mbps. Misconfigurin bandwidth parameters may result in transmit queue starvation, where the switch does not service with a low-bandwidth parameter sufficiently.

To configure guaranteed minimum bandwidth per output queue, use the following interface transm command:

bandwidth*bandwidth*

bandwidth specifies the guaranteed minimum bandwidth in bps using the optional prefixes kilo, me giga, using the k, m, and g command options, respectively.

Example 7-24 illustrates configuration of 17.1 Mbps as the minimum bandwidth on transmit queue

Example 7-24. Configuring Interface with Bandwidth Parameter

```
Switch#config terminal

Switch(config)#interface GigabitEthernet 1/1

Switch(config-if)#tx-queue 4

Switch(config-if-tx-queue)#bandwidth 17.1m

Switch(config-if-tx-queue)#end
```

To demonstrate and measure the behavior of sharing on transmit queues, two packet-generator pc connected to the switch as shown in Figure 7-4. The packet-generator port connected to Gigabit Et was sending traffic with a DSCP value of zero at 1.0 Gbps. The packet-generator port connected to Ethernet 1/2 was sending traffic with a DSCP value of 40 at 1.0 Gbps. The traffic sent by both inter intended for another traffic port connected to interface Gigabit Ethernet 5/1. Connected to interface Ethernet 5/1 was a traffic-generator port measuring the traffic rate for each DSCP flow. Three trials conducted. The first trial involved the default configuration of bandwidth, whereas the remaining tr applied the bandwidth parameter at 200 Mbps, 750 Mbps, and 900 Mbps, respectively. Table 7-5 s the results of the trial. Example 7-25 displays the basic configuration used for the trial.

Figure 7-4. Network Diagram that Demonstrates Sharing

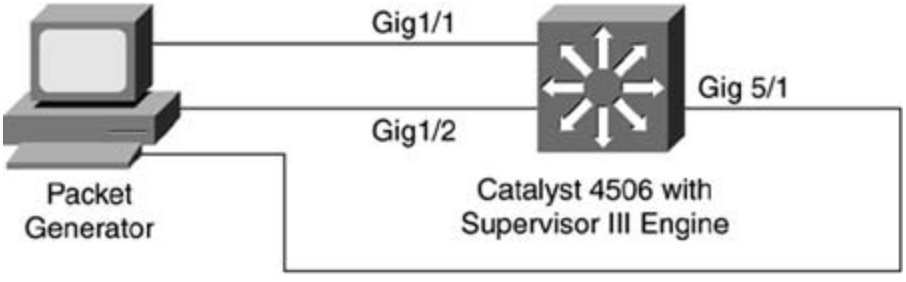


Table 7-5. Sharing on Transmit Queues

Received Rate	Default Configuration	Configured Bandwidth (200 Mbps)	Configured Bandwidth (750 Mbps)	Configur Bandwic (900/10
Rate received on Gigabit Ethernet 3/1 for traffic with DSCP value of 40	500 Mbps	480 Mbps	750 Mbps	900pps
Rate received on Gigabit Ethernet 3/1 for traffic with DSCP value of 0	500 Mbps	520 Mbps	250 Mbps	100Mbps

For the default configuration, the switch just load balances the egress traffic between the two egres containing packets for this test, queue 3 and queue 1. When the minimum bandwidth is configured Mbps for queue 3, the switch guarantees 200 Mbps to that queue. The switch must still guarantee 2 to the remaining queues and schedule traffic out of all queues using round-robin for traffic in exces configured bandwidth. As a result, a slight disparity in equality occurs when using a configured bar 200 Mbps on queue 3 and 250 Mbps for queue 1.

The second-to-last column in Table 7-5 illustrates results for configuring queue 3 for 750 Mbps and for 250 Mbps. While the last column illustrates results for configuring queue 3 for 900 Mbps and qu

100 Mbps.

Example 7-25. A Sample Sharing Configuration

```
Current configuration : 102 bytes
!
interface GigabitEthernet1/1
 no switchport
 ip address 10.0.1.1 255.255.255.0
 qos trust dscp
end
Switch#show run interface GigabitEthernet 1/2
Building configuration...
Current configuration : 102 bytes
!
interface GigabitEthernet1/2
 no switchport
 ip address 10.0.2.1 255.255.255.0
 qos trust dscp
end
Switch#show run interface gigabitEthernet3/1
Building configuration...
Current configuration : 136 bytes
!
interface GigabitEthernet3/1
 no switchport
 ip address 10.0.3.1 255.255.255.0
 qos trust dscp
 tx-queue 3
```

```
   bandwidth 200 mbps

end
```

Shaping

The Catalyst 4000 IOS Family of switches supports traffic shaping in addition to policing. Traffic sh
different characteristics than policing has. Chapter 2, "End-to-End QoS: Quality of Service at Layer
Layer 2," discusses the differences between policing and traffic shaping.

Traffic shaping configures per transmit queue to a specified rate. The switch schedules packets out
queue over time to maintain the configured rate. Packet drops occur only when a switch is unable t
packet into a full transmit queue.

All Fast Ethernet and Gigabit Ethernet support shaping. Shaping is configurable between 16 kbps a
maximum rate of the interface in 1-bps increments. Use the following interface transmit queue com
configure shaping:

shape*rate*

rate defines the traffic shaping maximum rate associated with a transmit queue.

Example 7-26 illustrates configuration of traffic shaping at 1.54 Mbps on transmit queue 1.

Example 7-26. Configuring the Bandwidth Parameter for a Transmit Queu

```
Switch#config terminal

Switch(config)#interface GigabitEthernet 1/1

Switch(config-if)#tx-queue 1

Switch(config-if-tx-queue)#shape 1.54m

Switch(config-if-tx-queue)#end
```

To demonstrate the behavior of traffic shaping against traffic-rate policing using a policer, two TCP throughput tests were conducted against a traffic policer and a traffic shaper. Two workstations rur application called TTCP were used to measure TCP/IP throughput against a policer and a traffic-sha configuration. TTCP was configured with an initial window size of only 4096.

Three trials were conducted. The first trial did not consist of any traffic policer or traffic-shaping configurations. Therefore, the two workstations were able to achieve maximum throughput. The se consisted of an individual policer similar to the one shown in Example 7-15 (except that the burst s lowered to 4000 bytes). The third trial consisted of the traffic-shaping configuration shown in Exam Figure 7-5 illustrates the simple topology used for this trial. Table 7-6 displays the results from the

Figure 7-5. Topology Used to Demonstrate Traffic Shaping Versus Traffic

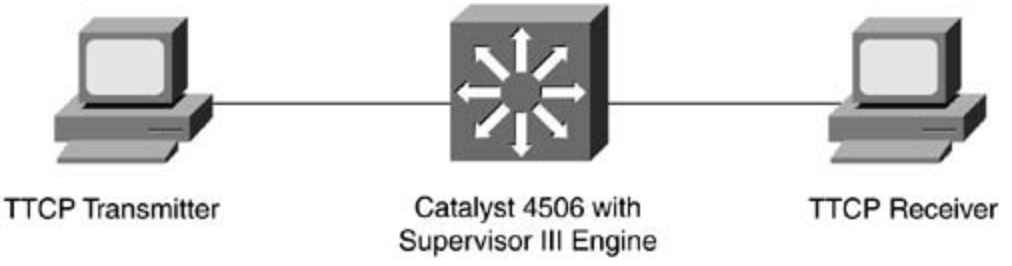


Table 7-6. Shaping on Transmit Queues

Trial	Maximum TTCP Throughput	Individual Policer at 1.54 Mbps	Traffic Shaping at 1.54 Mbps Illustrated in Example 7-27
Measured TTCP rate	5.34 Mbps	224 bps	832 bps

As evident from the trial, the behavior of TCP/IP traffic against traffic policers and traffic shaping v Although both yield TCP/IP throughput below the configured rate, the individual policer generates a significantly smaller throughput rate. Because the policers were not buffering traffic and the indivic policers were configured with a small burst size, the TCP/IP throughput measured significantly less traffic-shaping throughput. Because all network applications perform differently with different oper systems, apply these concepts carefully on a case-by-case basis.

Mapping Internal DSCP to CoS

As discussed throughout this chapter, the Catalyst 4000 IOS Family of switches uses the internal D differentiate service among packets. Because marking may occur on a packet as it traverses the sw CoS value of the packet needs to be updated on transmit. As a result, the switch employs the use c to-CoS mapping table for egress packets. The internal DSCP value determines the CoS value of the frame. Table 7-7 indicates the default DSCP-to-CoS mapping that occurs on egress frames.

Table 7-7. Default DSCP-to-CoS Mapping Table

DSCP Value	0–7	8–15	16–23	24–31	32–39	40–47	48–55	5
CoS Value	0	1	2	3	4	5	6	7

To display the current configured DSCP-to-CoS mapping, use the following command:

show qos maps dscp cos

The DSCP-to-CoS mapping table is configurable such that any internal DSCP value may map to any value. Use the following command to configure the DSCP-to-CoS mapping table:

qos map dscp*dscp-list***to cos***cos*

dscp-list represents up to eight DSCP values separated by a space. *cos* corresponds to the transmit value on the egress frame. Example 7-27 displays, configures, and verifies QoS DSCP-to-CoS mapp

Example 7-27. Displaying, Configuring, and Verifying the QoS DSCP-to-C Mapping Table

```
Switch#show qos map dscp cos

DSCP-CoS Mapping Table (DSCP = d1d2)
```

```
d1 : d2  0  1  2  3  4  5  6  7  8  9

-----------------------------------

 0 :    00 00 00 00 00 00 00 00 01 01

 1 :    01 01 01 01 01 01 02 02 02 02

 2 :    02 02 02 02 03 03 03 03 03 03

 3 :    03 03 04 04 04 04 04 04 04 04

 4 :    05 05 05 05 05 05 05 05 06 06

 5 :    06 06 06 06 06 06 07 07 07 07

 6 :    07 07 07 07

Switch#configure terminal

Switch(config)#qos map dscp 30 31 32 33 34 35 36 37 to cos 0

Switch(config)#qos map dscp 38 39 to cos 0

Switch(config)#end

Switch#show qos map dscpcos

DSCP-CoS Mapping Table (DSCP = d1d2)

d1 : d2  0  1  2  3  4  5  6  7  8  9

-----------------------------------

 0 :    00 00 00 00 00 00 00 00 01 01

 1 :    01 01 01 01 01 01 02 02 02 02

 2 :    02 02 02 02 03 03 03 03 03 03

 3 :    00 00 00 00 00 00 00 00 00 00

 4 :    05 05 05 05 05 05 05 05 06 06

 5 :    06 06 06 06 06 06 07 07 07 07

 6 :    07 07 07 07
```

Auto-QoS

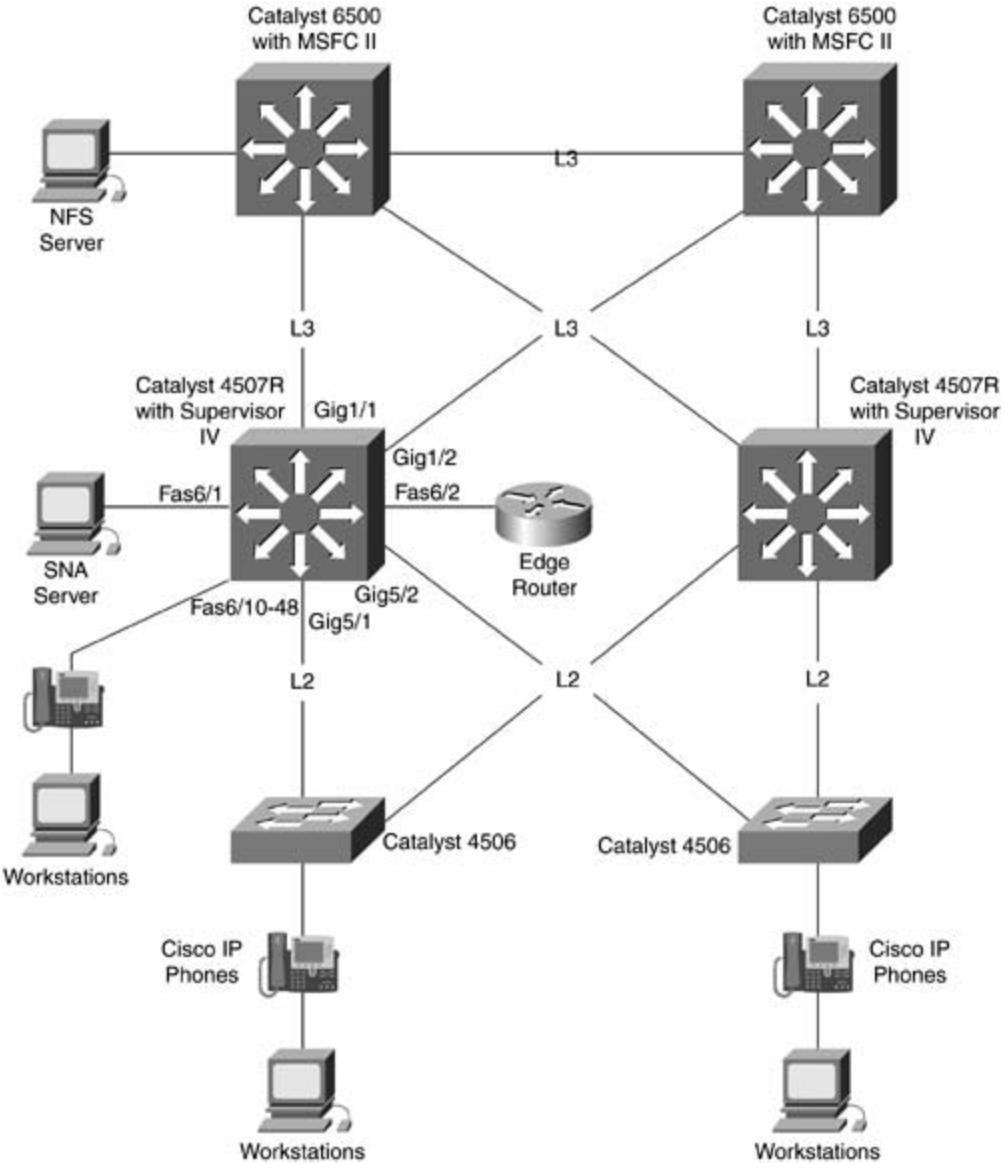
The Catalyst 4000 IOS Family of switches will support Auto-QoS in the second half of 2003. The go QoS is to provide for autoconfiguration of QoS configurations for IP Phone ports and interface QoS

configurations of access-distribution ports.

Case Study

Generally, system administrators do not configure more than a few QoS features simultaneously or switch. The topology displayed in Figure 7-6 illustrates a network that requires several QoS feature topology maximizes the use of QoS to illustrate several QoS features simultaneously. Nevertheless, features are commonly used in a variety of networks, especially those that require bandwidth restr (such as university campuses). For example, the case study illustrates restricting file-sharing traffi file-sharing profiles are becoming increasingly important in applying QoS features to campus netw

Figure 7-6. Case Study Topology



Example 7-28 illustrates several QoS features applied to the topology in Figure 7-6.

Example 7-28. Case Study Configuration

```
Switch#show running-config
Building configuration...
(text deleted)
!
qos aggregate-policer LIMIT_32KBPS 32 kbps 4000 byte conform-action transmit excee
➥drop
qos
vtp domain qos_TEST
vtp mode transparent
ip subnet-zero
!
!
class-map match-all MATCH_FILE_SHARING_FRAMES
  match access-group 100
!
!
policy-map LIMIT_FILE_SHARING
  class MATCH_FILE_SHARING_FRAMES
    police aggregate LIMIT_32KBPS
!
!
!
vlan 501-506,700
!
```

```
interface GigabitEthernet1/1
 description Uplink to Core
 no switchport
 ip address 10.1.1.2 255.255.255.0
 qos trust dscp
 tx-queue 1
   bandwidth 100 mbps
 tx-queue 2
   bandwidth 500 mbps
 tx-queue 3
   bandwidth 100 mbps
   priority high
 tx-queue 4
   bandwidth 50 mbps
!
interface GigabitEthernet1/2
 description Uplink to Core
 no switchport
 ip address 10.1.2.2 255.255.255.0
 qos trust dscp
 tx-queue 1
   bandwidth 50 mbps
 tx-queue 2
   bandwidth 500 mbps
 tx-queue 3
   bandwidth 100 mbps
   priority high
 tx-queue 4
```

```
   bandwidth 100 mbps
!
interface GigabitEthernet5/1
 description Link to Access Switch 501
 switchport trunk encapsulation dot1q
 switchport trunk allowed vlan 501,700
 qos trust dscp
 tx-queue 1
   bandwidth 100 mbps
 tx-queue 2
   bandwidth 500 mbps
 tx-queue 3
   bandwidth 100 mbps
   priority high
 tx-queue 4
   bandwidth 50 mbps
!
(text deleted)
interface FastEthernet6/1
 description SNA Server
 switchport mode access
 qos cos 2
 spanning-tree portfast
!
interface FastEthernet6/2
 description Edge_Router
 ip address 192.168.1.2 255.255.255.0
 tx-queue 1
```

```
   shape 1.0 mbps
 tx-queue 2
   shape 1.0 mbps
 tx-queue 3
   shape 500 kbps
 tx-queue 4
   shape 500 kbps
!
(text deleted)
!
interface FastEthernet6/10
 switchport mode access
 switchport voice vlan 700
 qos trust dscp
 tx-queue 3
   priority high
 spanning-tree portfast
!
(text deleted)
!
interface Vlan501
 description Data VLAN
 ip address 10.1.51.2 255.255.255.0
 no ip redirects
 service-policy input LIMIT_FILE_SHARING
 service-policy output LIMIT_FILE_SHARING
 standby 51 ip 10.1.51.1
!
```

```
(text deleted)
!
interface Vlan700
 description Voice VLAN
 ip address 10.70.1.2 255.255.255.0
 no ip redirects
 standby 70 ip 10.70.1.1
 standby 70 preempt
!
(text deleted)
!
access-list 100 permit tcp any any eq 30000
!
(text deleted)
!
end
Switch#
```

In Example 7-28, interfaces Gigabit Ethernet 1/1 and 1/2 connect directly to the core as Layer 3 int
therefore, trusting DSCP on ingress frames is desirable. The qos trust dscp configuration on each
achieves this desired configuration. Although not illustrated in the configuration, interfaces Gigabit
5/1 through 5/6 connect directly to access layer Catalyst 4506 switches. In this example, the acces
switches are responsible for determining the legitimacy of DSCP values on ingress frames. As a res
trusting DSCP on these connections is advantageous. In addition, the Cisco IP Phones connected to
Fast Ethernet 6/10 through 6/48 require trusting of DSCP to differentiate the VoIP frames.

An SNA server connected off Fast Ethernet 6/1 is sending non-IP frames. These packets control a c
application and need classification and scheduling as appropriate. The frames have no IP header, a
server transmits the frames with a CoS value of zero. As a result, configuring the interface for qos
applies an internal DSCP from the CoS-to-DSCP mapping table sufficient for prioritizing these fram
example, the SNA frames map to transmit queue 2.

The application of the policy map in the configuration intends to limit the amount of file-sharing tra
crossing VLAN boundaries. As a result, the switch applies the LIMIT_FILE_SHARING policy map to
VLANs, ingress and egress. The policy map uses an aggregate policer to limit traffic to only 32 kbp
sharing applications that match the class map clause defined in ACL 100.

For more accurate output scheduling, the interfaces connected to other switches use a nondefault s
configuration. Transmit queues 1 to 4 share traffic at rates of 50 Mbps, 500 Mbps, 100 Mbps, and 1
respectively. NFS applications are marking their traffic with a DSCP value of 20 by default. To prov
adequate sharing of traffic up to 500 Mbps, transmit queue 2 receives 500 Mbps as a share rate. Tr
queue 3 is strictly for VoIP traffic. The topology consists only of 100 Cisco IP Phones and restricting
transmit queue 3 to 100 Mbps is more than adequate for the amount of voice traffic passing betwe
switches. Because VoIP traffic maps to transmit queue 3 by default, configuring transmit queue as
priority queue forces the switch to service packets in transmit queue 3 first, and thus reduces the c
latency or jitter for VoIP traffic. All interfaces that connect to other switches and Cisco IP Phone por
transmit queue 3 as a high-priority queue.

Finally, the configuration consists of traffic shaping of transmit queues connected to an edge router
router only routes data traffic and therefore traffic-shaping works well in this example. The edge ro
connects to an ATM cloud, which can service only about 3.0 Mbps of data. As a result, the traffic-sh
configuration limits traffic to specific rates per queue.

QoS Support on the Catalyst 2948G-L3, 4908G-L3, and Catalyst 4000 Layer 3 Services Module

The following sections discuss the Catalyst 2948G-L3 and 4908G-L3 switches, and the Catalyst 4000 Layer 3 services module. These switches resemble IOS-based routers from a configuration perspective and are generally referred to as G-L3 switches. However, these switches support only a few QoS features. The supported QoS features include rate limiting, IOS-based traffic shaping, and output scheduling. These switches base classification just on IP precedence values and have no support for marking.

These features use hardware TCAM to achieve 6 million packets-per-second performance. As with other Catalyst platforms, the term *hardware switching* represents using high-speed hardware components such as TCAM for packet processing.

These Layer 3 switches provide for classification based on IP precedence only. These switches do not support classification based on DSCP or classification based on the Layer 2 CoS field of a packet. Furthermore, these Layer 3 switches do not support classification based on ACLs, reclassification, or marking. As a result, these Layer 3 switches function solely as a QoS packet-forwarding switch. In other words, the intended QoS function of these Layer 3 switches is to route and schedule packets. In an end-to-end design, these switches rely on the adjoining switches and edge routers for classification and marking.

This part this chapter covers the following topics and QoS features for the G-L3 switches:

- Architecture Overview
- Software Requirements
- Global Configuration
- Classification
- Output Scheduling
- Per-Port Traffic Shaping
- Rate Limiting
- Case Study

Catalyst 2948G-L3, 4908G-L3, and 4232-L3 Services Module QoS Architectural Overview

The 2948G-L3 and 4908G-L3 Layer 3 switches and the WS-X4232-L3 Layer 3 services module share the same architecture with different port densities. The architecture of these switches uses TCAM memory for packet processing and forwarding of IP, IPX, and Layer 2 frames. As with the Catalyst 4000 IOS Family of switches, packet processing and forwarding needs to occur via hardware switching rather than software switching. Hardware switching allows for high-rate traffic flows with several QoS features. The QoS features supported on these switches include *weighted round-robin* (WRR) scheduling, traffic policing, and traffic shaping for IP packets. QoS features do not exist for IPX and bridged traffic, and these QoS features cannot coexist with IPX

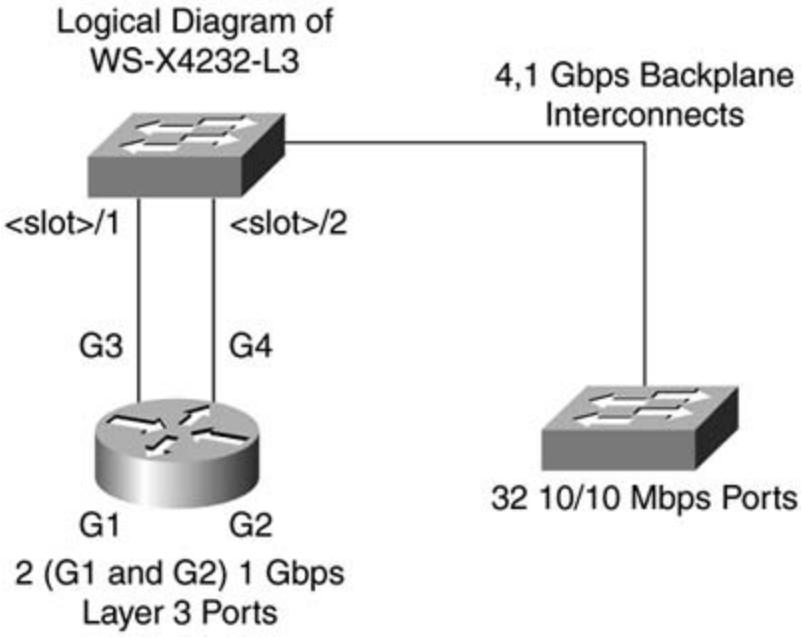
configurations.

The WS-X4232-L3 Services Module Architecture

The Catalyst 4000 Layer 3 services module, WS-X4232-L3, employs a unique architecture for integration into the Catalyst 4000 CatOS switch. At the time of publication, the Catalyst 4000 IOS Family of switches does not support the Catalyst 4000 Layer 3 services module.

The WS-X4232-L3 module consists of 4 Gigabit Ethernet ports and 32 10/100-Mbps ports. Two of the four Gigabit Ethernet ports connected directly to the supervisor interconnect on the Catalyst 4000 or 4500 chassis backplane. The other two Gigabit Ethernet ports are available as front-panel ports. The architecture of WS-X4232-L3 modules intends for the two front-panel ports to function as Layer 3 ports only. All four Gibabit Ethernet ports support port channeling (Gigabit EtherChannel). However, port-channel interfaces do not support QoS features. Furthermore, the WS-X4232-L3 does not support *integrated routing and bridging* (IRB). The 32 10/100-Mbps front panels are Layer 2-only ports. These 32 10/100-Mbps front panels are configurable only from the CatOS *command-line interface* (CLI) and not from the WS-X4232-L3 services module. Figure 7-7 provides a logical depiction of the WS-X4232-L3 module architecture when configured in a Catalyst 4000 CatOS switch.

Figure 7-7. WS-X4232-L3 in a Catalyst 4000 Switch Architecture Depiction



The configuration of the G-L3 switches is rather unique, with many restrictions because the switch configures just like an IOS router. For instance, VLAN interfaces do not exist in the configuration. Because of the unique configuration of these Layer 3 switches, consult the following technical documents at Cisco.com for more details regarding configurations, limitations, and caveats:

- "Configuration and Overview of the Router Module for the Catalyst 4000 Family (WS-X4232-L3)" Document ID: 6198
- "Catalyst 4908G-L3 VLAN Routing and Bridging Example Configuration" Document ID: 14972
- "Catalyst 2948G-L3 Sample Configurations - Single VLAN, Multi-VLAN, and Multi-VLAN Distribution Layer Connecting to Network Core" Document ID: 12020

Software Requirements

All software versions for the Catalyst 2948G-L3 and 4908G-L3 Layer 3 switches and the WS-X4232-L3 Layer 3 services support output scheduling using WRR. Cisco IOS Software versions 12.0(10)W5(18e) and later add feature support for per-port traffic shaping and rate limiting.

Catalyst 4000 CatOS switches require Cisco CatOS Software version 5.5 or higher for use of the WS-X4232-L3 module.

Global Configuration

The Catalyst 2948G-L3 and 4908G-L3 Layer 3 switches and the WS-X4232-L3 module schedule packets based on IP precedence by default. Use the following command to disable QoS on these switches:

```
[no]qos switching
```

The **show qos switching** command displays the state of global QoS configuration. The **show qos mapping** outputs the global QoS WRR mapping configuration. The **show qos mapping** [**destination***egress-interface*] command displays the QoS WRR mapping configuration for an egress interface.

Example 7-29 shows samples of the **show qos switching** and **show qos mapping** commands, respectively.

Example 7-29. Sample Use of show qos switching and show qos mapping Commands

```
Router#show qos switching
```

```
qos Based IP Switching enabled

Router#show qos mapping destination GigabitEthernet 1

Precedence WRR-Weight

     0       1

     1       2

     2       3

     3       4
```

Classification

The Catalyst 2948G-L3 and 4908G-L3 Layer 3 switches and the WS-X4232-L3 module output schedule packets using a four-queue class model. Classification differentiates packets into one of four queue classes for output scheduling.

The Catalyst 2948G-L3 and 4908G-L3 Layer 3 switches and the WS-X4232-L3 module derive classification solely based on IP precedence information. Furthermore, these switches classify frames based on the two *most-significant bits* (MSBs) of the IP precedence field. Table 7-8 illustrates how IP precedence bits map to the queue classes.

Table 7-8. Default IP Precedence–to–Queue Class Mapping

IP Precedence (Binary)	Queue Class
0 (000)	Queue 0
1 (001)	Queue 0
2 (010)	Queue 1
3 (011)	Queue 1
4 (100)	Queue 2
5 (101)	Queue 2
6 (110)	Queue 3
7 (111)	Queue 3

Output Scheduling

The Catalyst 2948G-L3 and 4908G-L3 Layer 3 switches and the WS-X4232-L3 module support output scheduling using WRR scheduling, rate limiting, and traffic-shaping features. These

switches support traffic shaping only on egress traffic flows. Rate-limiting support is applicable to either ingress or egress traffic flows. The following sections discuss WRR scheduling, rate limiting, and traffic shaping applicable to the Catalyst 2948G-L3 and 4908G-L3 Layer 3 switches and the WS-X4232-L3 module.

WRR Scheduling

WRR scheduling imparts higher bandwidth to higher-priority queues while still providing service to lower-priority queues. Maintaining service to lower-priority queues avoids queue starvation.

The premise for WRR on the Catalyst 2948G-L3 and 4908G-L3 Layer 3 switches and the WS-X4232-L3 module is to apply effective bandwidths to each of the four queues. Use the following effective bandwidth formula to determine WRR weight:

$$(W/S) \times B = n$$

W represents the WRR weight of the queue, and *S* represents the sum of all weights of the active queues.*B* is the available bandwidth of the outgoing interface(s), and *n* represents the effective bandwidth.Table 7-9 displays the default effective bandwidth per queue.

Table 7-9. Default WRR Weights Assigned to Queue Class

Queue Class	Two MSB of IP Precedence	Weight	Effective Bandwidth	Percent of Total Bandwidth
Queue 0	0	1	100 Mbps	10%
Queue 1	1	2	200 Mbps	20%
Queue 2	2	3	300 Mbps	30%
Queue 3	3	4	400 Mbps	40%

Chapter 2 discusses the behavior of WRR in more detail.

The queue mapping of the two MSBs of IP precedence to the WRR weighted value configures globally and per interface. Interface configuration overrides the global configuration. Use the following command to globally adjust the IP precedence to the WRR weighted value:

qos mapping precedence*value***wrr-weight***weight*

To adjust the mapping of IP precedence to the WRR weighted value, use the following global configuration command:

qos mapping [**destination***egress-interface*]**precedence***value***wrr-weight***weight*

For both commands, *value* defines the two MSBs of IP precedence, and *weight* represents the WRR weight. *egress-interface* defines the egress interface to apply the command configuration.

Example 7-30. Sample Configuration of Mapping IP Precedence to WRR Weights Globally and Per Interface

```
Router#show running-config

Building configuration...

(text deleted)

qos mapping precedence 1 wrr-weight 1

qos mapping destination GigabitEthernet1 precedence 2 wrr-weight 1

(text deleted)

!

end
```

To demonstrate WRR behavior, several trials were performed using the topology in Figure 7-8. In each trial, a traffic generator transmitted traffic to the switch at 1 Gbps on both supervisor ports 1/1 and 1/2. The ingress traffic on port 1/1 had an IP precedence value of 0, and ingress traffic on port 1/2 had an IP precedence value of 7. The traffic was routed from interfaces 1/1 and 1/2 out to interface Gigabit Ethernet 1 on the WS-X4232-L3 module. The traffic generated was also connected to this port to measure the egress traffic rate for both streams.

In the first trial, the default QoS mapping configuration was used. In the second trial, the QoS mapping configuration was changed such that traffic from both streams mapped to the same WRR weight. Table 7-10 summarizes the results of the trial.

Table 7-10. WRR on the Catalyst WS-X4232-L3 Module

Trial	DefaultWRR Mapping	All IP Precedence Map to the SameWRRWeight
Measured rate of traffic for frames with IP precedence (0)	200 Mbps	500 Mbps
Measured rate of traffic for frames with IP precedence (7)	800 Mbps	500 Mbps

Based on the WRR bandwidth formula, both trials yielded expected results. For the default WRR mapping trial, the router used only two queues. As a result, the sum of all weighted values was 5 (because Q1 has a weight of 1 and Q4 has a weight of 4). Therefore, the derived formula for the effective bandwidth for the IP precedence traffic of 7 was computed as follows:

$(W/S) \times B = n$

(4/5) x 1.0 Gbps = 800 Mbps for IP precedence 7 traffic

For the second trial, mapping all the IP precedence values to the same WRR weight yields an even distribution of egress traffic for both streams.

Per-Port Traffic Shaping

Traffic shaping uses the leaky token bucket algorithm. The leaky token bucket algorithm discussed earlier in this chapter for the Catalyst 4000 IOS Family also applies to the Catalyst 2948G-L3 and 4908G-L3 Layer 3 switches and the WS-X4232-L3. Although the hardware architecture and leaky token bucket algorithm implementations differ significantly between the two types of switches, the algorithm still applies for understanding the rate and burst parameters for the Catalyst 2948G-L3 and 4908G-L3 Layer 3 switches and the WS-X4232-L3 Layer 3 services switches. Note that unlike rate-limiting policers, traffic shaping actually buffers packets that exceed the specified rate.

Use the following command to configure traffic shaping in the interface configuration mode:

traffic-shape*rate burst*

rate represents the traffic-shaping rate in bps. The Catalyst 2948G-L3 and 4908G-L3 Layer 3 switches and the WS-X4232-L3 module support configuration of 32 kbps to the maximum interface speed in 1-bps increments for egress shaping. *burst* defines the burst size in bits.

Example 7-31 illustrates a sample configuration of traffic shaping.

Example 7-31. Traffic Shaping Sample Configuration

```
Router#show running-config
Building configuration...
(text deleted):
!
interface GigabitEthernet1
 ip address 192.168.1.1 255.255.255.0
 no ip directed-broadcast
 traffic-shape rate 1540000 20000
(text deleted)
!
end
```

Rate Limiting

Applications that are sensitive to delay and jitter, such as VoIP, interactive video, and stock tickers, do not tolerate the buffering that occurs when traffic shaping is active. Therefore, caution is necessary when configuring and applying traffic shaping to VoIP networks.

Rate limiting does not buffer packets. Instead, the switch drops packets over the specified rate and above burst. The Catalyst 2948G-L3 and 4908G-L3 Layer 3 switches and the WS-X4232-L3 switches do not support any other action for out-of-profile packets other than drop. These switches do support configuration of ingress and egress rate limiting in both ingress and egress applications.

Use the following interface command to configure rate limiting:

rate-limit [**input** | **output**]*rate burst*

rate defines the target rate on a per-interface basis in bps. The Catalyst 2948G-L3 and 4908G-L3 Layer 3 switches and the WS-X4232-L3 support rates from 32000 bps to the maximum link speed in 1-bps increments. *burst* represents the burst size in bits configured in 1-byte increments between 0 and 64000 bytes. Example 7-32 shows a sample interface configuration for rate limiting.

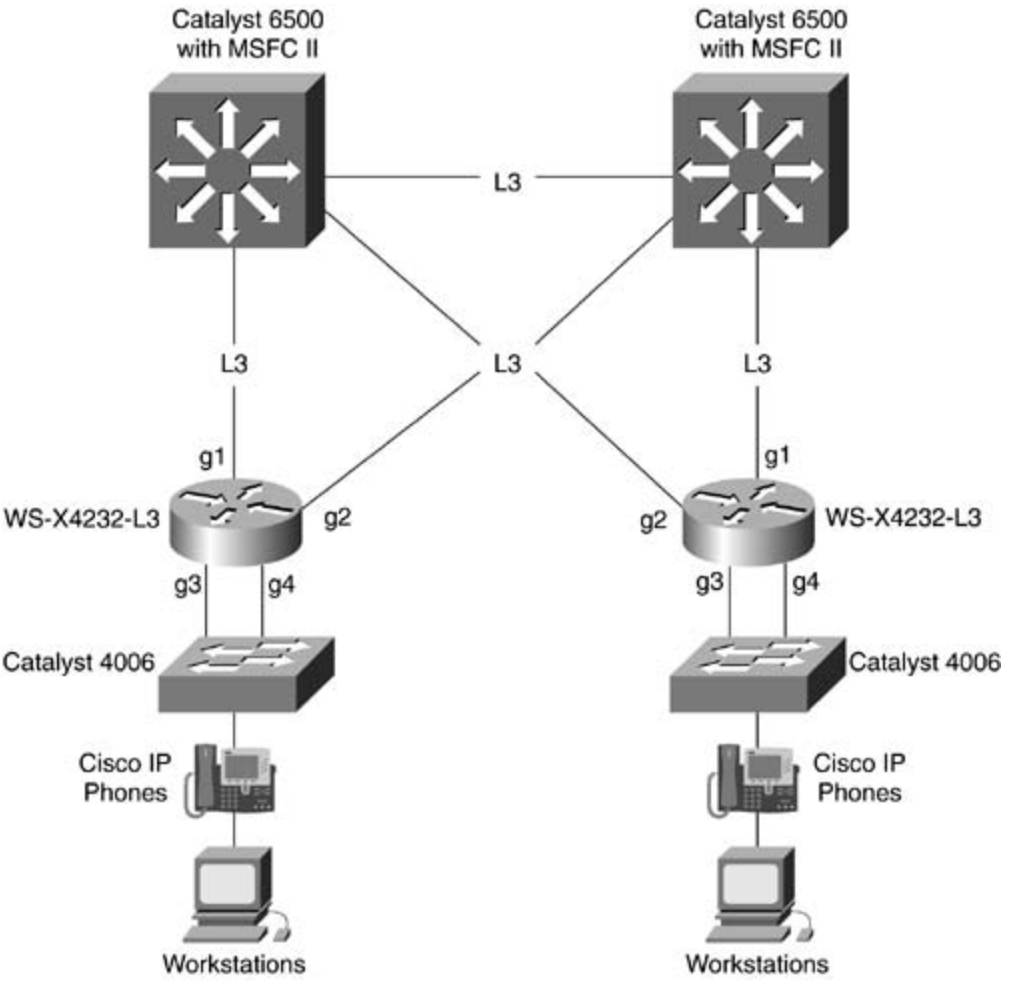
Example 7-32. Sample Configuration of Rate Limiting Applied Ingress

```
Router#show running-config

Building configuration...

(text deleted):

!

interface GigabitEthernet2

 ip address 192.168.2.1 255.255.255.0

 no ip directed-broadcast

 rate-limit input 5000000 64000

(text deleted)

!

end
```

Case Study

Example 7-33 illustrates a typical configuration of a WS-X4232-L3 in slot 2 of a Catalyst 4000 chassis as illustrated in Figure 7-8.

Figure 7-8. Case Study Topology



Example 7-33. Case Study Configuration for Figure 7-8

```
Router#show running-config
Building configuration...
(text deleted)
!
qos mapping destination GigabitEthernet1 precedence 0 wrr-weight 1
qos mapping destination GigabitEthernet1 precedence 1 wrr-weight 1
qos mapping destination GigabitEthernet1 precedence 2 wrr-weight 4
qos mapping destination GigabitEthernet1 precedence 3 wrr-weight 2
qos mapping destination GigabitEthernet3 precedence 0 wrr-weight 1
```

```
qos mapping destination GigabitEthernet3 precedence 1 wrr-weight 1
qos mapping destination GigabitEthernet3 precedence 2 wrr-weight 4
qos mapping destination GigabitEthernet3 precedence 3 wrr-weight 2
qos mapping destination GigabitEthernet4 precedence 0 wrr-weight 1
qos mapping destination GigabitEthernet4 precedence 1 wrr-weight 1
qos mapping destination GigabitEthernet4 precedence 2 wrr-weight 4
qos mapping destination GigabitEthernet4 precedence 3 wrr-weight 2
(text deleted)
!
interface GigabitEthernet1
 ip address 10.0.1.2 255.255.255.0
 no ip directed-broadcast
 rate-limit output 100000 64000
!
interface GigabitEthernet2
 ip address 10.0.2.2 255.255.255.0
 no ip directed-broadcast
 !
interface GigabitEthernet3
 no ip address
 no ip directed-broadcast
 no negotiation auto
!
interface GigabitEthernet3.1
 encapsulation dot1Q 1
 ip address 10.1.1.2 255.255.255.0
 no ip redirects
 no ip directed-broadcast
```

```
 standby 1 ip 10.1.1.1
!
interface GigabitEthernet3.3
 encapsulation dot1Q 3 native
 ip address 10.1.3.2 255.255.255.0
 no ip redirects
 no ip directed-broadcast
 standby 3 ip 10.1.3.1
!
interface GigabitEthernet4
 no ip address
 no ip directed-broadcast
 no negotiation auto
!
interface GigabitEthernet4.2
 encapsulation dot1Q 2
 ip address 10.1.2.2 255.255.255.0
 no ip redirects
 no ip directed-broadcast
 standby 2 ip 10.1.2.1
!
interface GigabitEthernet4.4
 encapsulation dot1Q 4
 ip address 10.1.4.2 255.255.255.0
 no ip redirects
 no ip directed-broadcast
 standby 4 ip 10.1.4.1
!
```

```
(text deleted)

!

end
```

The configuration applies dot1q subinterfaces to each Gigabit Ethernet interface separately to achieve load balancing without using port channeling.

The QoS mapping configuration consists of prioritizing IP precedence values 4 and 5 with 50 percent of the effective bandwidth of interfaces Gigabit Ethernet 1, 3, and 4. By default, Cisco IP Phones assign an IP precedence value of 5, which maps to queue class 2, to voice traffic (thus the reasoning for using a higher WRR weight for this queue class).

Furthermore, interface Gigabit Ethernet 1 connected to the core switch applies a rate-limiting configuration. The rate limiting limits all egress traffic flowing to the core to 100 Mbps.

Summary

The Catalyst 4000 IOS Family of switches provides for a wide range of QoS features. The switch bases classification, marking, policing, and output scheduling on not only DSCP values but also CoS values. The fine granularity of QoS features and configuration options allow for these switches to reside in either the core, distribution, or access layer of the campus network topology. You can summarize QoS feature support on the Catalyst 4000 IOS Family of switches discussed in the first part of the chapter as follows:

- At the time of publication, the Catalyst 4000 IOS Family of switches includes the Catalyst Supervisor Engines III and IV in a Catalyst 4000 or 4500 series chassis.
- Support exists for classification, reclassification, marking, and output scheduling.
- QoS packet processing occurs in hardware to achieve line-rate performance.
- 1022 ingress and 1022 egress policers applied as aggregate or individual policers are supported.
- Each interfaces uses four transmit queues for output scheduling.
- Sharing and traffic shaping are configurable per transmit queue.

The Catalyst 2948G-L3 and 4908G-L3 and the Catalyst 4000 Layer 3 services module, WS-X4232-L3, act only as QoS forwarding switches. For QoS features such as trusting and marking, these switches rely solely on externally connected switches or routers. You can summarize the QoS feature support on these switches discussed in the second part of this chapter as follows:

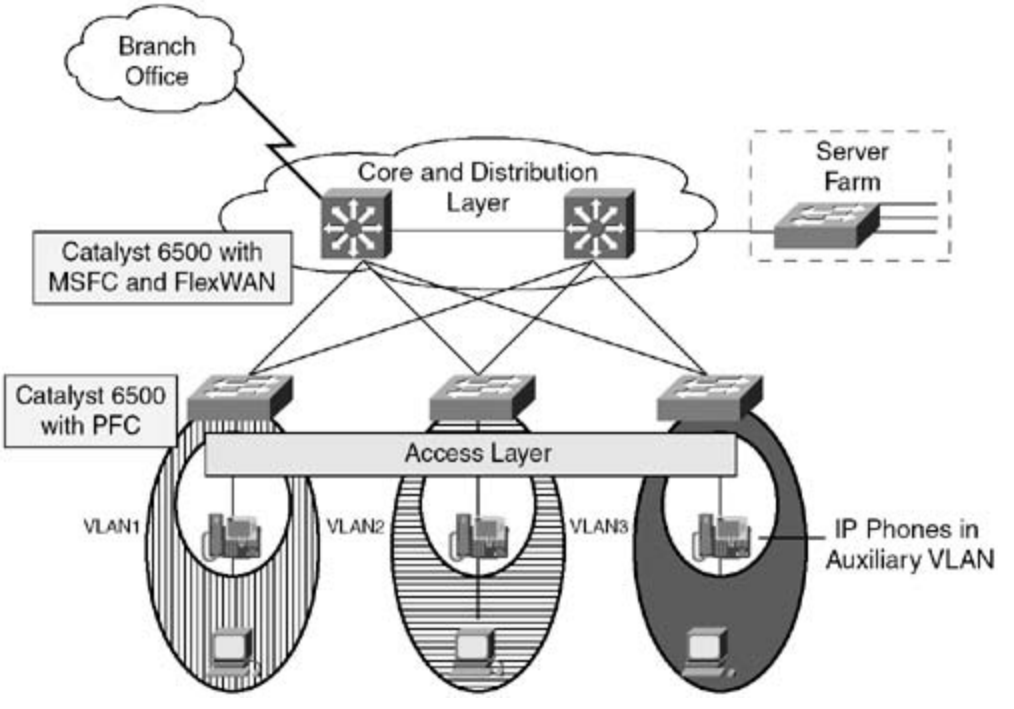
- Support exists for classification and output scheduling.
- Classification is determined by IP precedence only.
- Output scheduling uses WRR to differentiate service.
- Rate limiting is configurable per interface on ingress or egress.
- Traffic shaping is configurable per interface on egress only.
- Port-channel interfaces do not support any QoS features.

Chapter 8. QoS Support on the Catalyst 6500

The Cisco Catalyst 6500 series incorporates the industry's leading platforms, with regard to QoS features and functionality. The Catalyst 6500 Family of switches offers a broad catalogue of mechanisms to help ensure the timely end-to-end delivery of mission-critical and delay-sensitive applications. The Catalyst 6500 series is also capable of performing these mechanisms in hardware, with no impact to the overall operation of the switch.

Because of the Catalyst 6500's versatility, you can deploy it in all aspects of the campus environment. By integrating additional hardware components, the Catalyst 6500 is capable of transporting converged data, voice and real-time video across a LAN, MAN, or even WAN environment. Features such as inline power for phone support, trust, and per-port queuing and scheduling allow the Catalyst 6500 to be a viable solution for access layer deployment. Dual-rate policing,*Weighted Random Early Detection* (WRED), and marking extend the Catalyst 6500's reach to the distribution and core layers, as well as the WAN edge. Figure 8-1 shows how the Catalyst 6500 can be positioned in a small campus network.

Figure 8-1. Positioning the Catalyst 6500 Series in the Campus



Chapter 2 "End-to-End QoS: Quality of Service at Layer 3 and Layer 2," introduces many of the QoS features and capabilities found on the Catalyst 6500. This current chapter expounds on these concepts and discusses how these features specifically relate to the Catalyst 6500. The chapter opens with an architectural overview of the Catalyst 6500, and then discusses the

hardware and software requirements necessary to support QoS. The QoS discussion ensues with a quick demonstration of enabling QoS on the platform, immediately followed by the features outlined in the following list:

- Input Scheduling
- Classification and Marking
- Mapping
- Policing
- Congestion Management and Congestion Avoidance
- Automatic QoS

This chapter focuses on the campus LAN aspect of QoS for the Catalyst 6500 and addresses how the various QoS mechanisms function on this Catalyst switch without focusing on the role of the *Multilayer Switch Feature Card* (MSFC) or FlexWAN. Numerous configuration examples are provided, reinforcing the concepts discussed. CatOS versions 6.3, 6.4, and 7.5 and Cisco IOS version 12.1(13)E were used to configure the examples. The command references demonstrate how to configure the various QoS capabilities using both CatOS (Hybrid mode) and Cisco IOS (Native mode).

The Catalyst 6500 Family consists of both the Catalyst 6000 and Catalyst 6500. Despite some architectural differences, the Catalyst 6500 encompasses all features available on the Catalyst 6000. The following section lists the critical hardware components essential to QoS operation on the Catalyst 6500.

NOTE

The Catalyst 6000 chassis is "end-of-sale." However, the QoS mechanisms described in this chapter apply to both the Catalyst 6000 and the 6500 chassis. QoS capabilities depend on the installed modules.

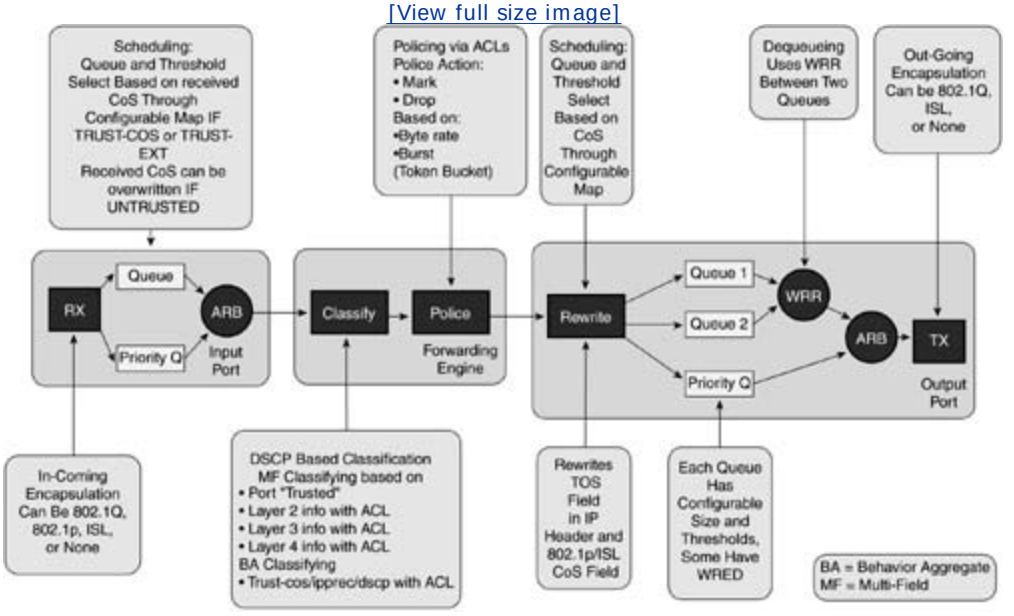
Catalyst 6500 Architectural Overview

This section introduces the various components relevant to QoS operation on the Catalyst 6500. The section covers hardware resources and terms presented throughout the chapter.

Figure 8-2 shows the QoS architecture of the Catalyst 6500. The ingress and egress ports depict the queuing architecture found on more recent Gigabit Ethernet ports. The figure demonstrates the order in which the QoS functions occur and also denotes which switch components are responsible for the different mechanisms.

Figure 8-2. Overview of QoS on the Catalyst 6500 Family Architecture

[View full size image]



Incorporating an MSFC and a FlexWAN line module further enhances the platform's QoS support. With these modules, additional QoS features include traffic shaping, Low Latency Queuing, Class-Based Weighted Fair-Queuing, and complex traffic classification based on Layer 4 through Layer 7 application recognition. For information about QoS in conjunction with the MSFC and the FlexWAN module, see Chapter 9, "QoS Support on the Catalyst 6500 MSFC and FlexWAN."

Both the Catalyst 6000 and the Catalyst 6500 chassis utilize a 32-Gbps bus for communication with non-fabric-enabled modules. Non-fabric-enabled linecards denote modules only capable of accessing the 32-Gbps bus architecture available on the Catalyst 6000 and Catalyst 6500. The 32-Gbps bus is referred to as the *data bus* or *D-bus*. The D-bus transports all frames between the various linecards. All information traversing the D-bus is viewed by all modules, including the supervisor engine. In addition to the D-bus, the Catalyst 6500 utilizes two additional buses, the*results bus* (R-bus) and the control bus or *Ethernet out-of-band channel* (EOBC). The R-bus forwards the appropriate rewrite information from the supervisor engine to the individual port *application-specific integrated circuits* (ASICs). The rewrite data includes the destination MAC and egress port information, and any QoS classification or policing policies applied to the frame. The CoS values derived from these QoS mechanisms may differ from the original *class of service*

(CoS) value present when the frame entered the switch. The EOBC provides a conduit for the supervisor engine to perform system management functions with the various linecards. Again, the bus architecture facilitates communication between traditional non-fabric-enabled modules, as well as between non-fabric-to-fabric-enabled modules.

The Catalyst 6500, unlike the Catalyst 6000, offers additional connectivity to a 256-Gbps crossbar switching fabric, accessible by incorporating a *switch fabric module* (SFM) and fabric-enabled linecards. *Fabric-enabled cards* are modules with the capability to connect to the crossbar fabric. The crossbar fabric provides high-speed communication between fabric-capable linecards only. Accessibility to the fabric is the fundamental difference between the Catalyst 6000 and the Catalyst 6500 from an architectural perspective. Incorporating a switch fabric module significantly increases the packet-forwarding rate relative to the traditional bus architecture. Table 8-2 lists the various modules and their connectivity to the backplane.

The MSFC and the *Policy Feature Card* (PFC) daughter cards of the Catalyst 6500 supervisor engines are responsible for the Layer 3 routing of packets through the switch. The MSFC and PFC have specialized functions. Not only does the PFC switch Layer 2 frames, it also routes Layer 3 packets based on Layer 3 information provided by the MSFC. Consequently, the MSFC's function is to build the Layer 3 routing and *Address Resolution Protocol* (ARP) tables, which may subsequently be passed to the PFC, in addition to handling any Layer 3 routing functionality the PFC cannot perform. Because the Layer 3 forwarding performance for the MSFC is substantially less than the PFC, the intent is not to forward all the packets using the MSFC. The goal of the Catalyst 6500 architecture is to Layer 3 route all packets via the PFC.

For CEF-based systems, the MSFC passes Layer 3 forwarding information through the EOBC to the appropriate ASICs on the PFC2. *Mulilayer switching* (MLS)-based systems do not use the EOBC to forward Layer 3 information. Instead, when a switched path is completed by the MSFC, a flow is created and cached and the PFC1 forwards all subsequent packets for that flow. *Cisco Express Forwarding* (CEF) and MLS allow the PFC to switch packets in hardware.

Layer 3 forwarding depends on the supervisor engine. For Supervisor I Engines, the forwarding information is based on MLS flows, whereas the PFC2 on Supervisor II Engines utilizes hardware-based CEF. MLS operation is covered in Chapter 4, "QoS Support on the Catalyst 5000 Family of Switches," in the section titled "MLS Fundamentals." Regardless of the Layer 3 switching method, QoS features are applied to the packet prior to it being Layer 3 switched.

NOTE

For additional information on MLS-based switching on the Supervisor I Engine, refer to the following technical document at Cisco.com:

"Configuring IP Unicast Layer 3 Switching on Supervisor Engine 1"

For additional information on CEF-based switching on the Supervisor II Engine, refer to the following technical document at Cisco.com:

"Configuring CEF for PFC2"

In addition to the PFC, which is centrally located on the supervisor engine, fabric-enabled linecards may incorporate a *Distributed Forwarding Card* (DFC). At the time of writing, the WS-X6816 linecard is the only module that comes equipped with a DFC by default. However, other fabric-enabled linecards can be upgraded to accept a DFC. The DFC is a daughter card that sits

on a fabric-enabled linecard. The DFC's architecture and operation is exactly the same as the PFC2. As a result, the DFC is capable of performing distributed Layer 3 CEF-based forwarding, Layer 2 bridging, *access-control lists* (ACLs), and QoS. Therefore, by adding a DFC, forwarding decisions are localized to the linecard. Again, similar to the PFC2, the MSFC is responsible for building the CEF information that is distributed out to the DFC.

Finally, the *ternary content addressable memory* (TCAM) is a finite portion of memory resident on the PFC1 and PFC2. The TCAM is essentially a table that stores ACL entries and masks used to apply defined QoS policies. The TCAM allows multiple *access-control entries* (ACEs) to share a single mask. Storing the ACL information in memory on the PFC ensures high-speed lookups are performed, and thus maximizes the throughput and minimizes the latency for processing packets and applying QoS policies. Because QoS ACL lookups are performed in hardware, applying the policies results in no impact to system switching performance. TCAM is discussed in more detail in Chapter 6, "QoS Features Available on the Catalyst 2950 and 3550 Family of Switches," as well as later in this chapter.

For more detailed information regarding the architecture of the Catalyst 6500, consult the following technical document at Cisco.com:

"Catalyst 6000 and 6500 Series Architecture"

Software and Hardware Requirements

QoS feature support for the Catalyst 6500 began with software version 5.1. Initially, with the Supervisor I Engine, the platform was limited in QoS functionality and only capable of performing Layer 2 functions. Specifically, it was only capable of supporting port-based CoS, as well as assigning a CoS based on destination MAC address. With the introduction of the PFC1 in CatOS Software version 5.3, QoS support on the Catalyst 6500 broadened to include policing, marking, and classification based on QoS ACLs for IP, IPX, and MAC layer traffic. QoS is also fully supported in Cisco IOS (Native mode). Initially, QoS support only included IP traffic in the first Cisco IOS Software version release 12.0(7)XE. Marking, policing, classification and congestion avoidance were the features included for IP traffic. Cisco IOS release 12.1(1)E expanded QoS support for Native mode to include IPX and MAC layer traffic. Table 8-1 depicts the different QoS processes covered in this chapter.

The table specifies the hardware responsible for the different operations and the software capable of supporting the various features.

Table 8-1. Hardware Support for QoS

QoS Operation	Hardware Responsible for QoS Operation	Supported Software
Input queue scheduling[*]	Linecards (port ASIC)—PFC not required	CatOS/Cisco IOS
Classification	Supervisor—Responsible for Layer 2 (CoS) PFC—Responsible for Layer 2 and 3 (CoS/IP precedence/DSCP)	CatOS/Cisco IOS
Policing	Layer 3 switching engine in PFC	CatOS/Cisco IOS
Marking/rewrite	Linecards (port ASIC)—Based on classification/policing performed by supervisor or PFC	CatOS/Cisco IOS
Output queue scheduling	Linecards (port ASIC)—Based on priorities established during classification/policing	CatOS/Cisco IOS

[*] Input queue scheduling is contingent on the trust state of the inbound port. If the inbound trust policy is set for untrusted, traffic is sent to the default queue and is serviced FIFO.

As demonstrated in Table 8-1, the port ASICs on the linecards play a significant role in the end-to-end QoS implementation within the Catalyst 6500. Table 8-2 shows the different modules available for the platform and the default queuing architecture for both receive and transmit ports.

Table 8-2. Overview of Modules Supporting QoS

Module	Linecard Composition	Receive Ports	Transmit Ports	Priority Queue	Architecture to Backplane
WS-X6K-Sup1	2 X 1000	1q4t	2q2t	No	Bus
WS-X6K-Sup1A	2 X 1000	1p1q4t	1p2q2t	RX/TX	Bus
WS-X6K-Sup2	2 X 1000	1p1q4t	1p2q2t	RX/TX	Bus/Fabric
WS-X6024	24 X 10	1q4t	2q2t	No	Bus
WS-X6148	48 X 10/100	1q4t	2q2t	No	Bus
WS-X6224	24 X 100	1q4t	2q2t	No	Bus
WS-X6248	48 X 10/100	1q4t	2q2t	No	Bus
WS-X6316	16 X 1000	1p1q4t	1p2q2t	RX/TX	Bus
WS-X6324	24 X 100	1q4t	2q2t	No	Bus
WS-X6348	48 X 10/100	1q4t	2q2t	No	Bus
WS-X6408	8 X 1000	1q4t	2q2t	No	Bus
WS-X6408A	8 X 1000	1p1q4t	1p2q2t	RX/TX	Bus

WS-X6416	16 X 1000	1p1q4t	1p2q2t	RX/TX	Bus
WS-X6501	1 X 10GE	1p1q8t	1p2q1t	RX/TX	Bus/fabric
WS-X6502	1 X 10GE	1p1q8t	1p2q1t	RX/TX	Bus/fabric
WS-X6516	16 X 10/100/1000	1p1q4t	1p2q2t	RX/TX	Bus/fabric
WS-X6524	24 X 100	1p1q0t	1p3q1t	RX/TX	Bus/fabric
WS-X6548	48 X 10/100	1p1q0t	1p3q1t	RX/TX	Bus/fabric
WS-X6816[*]	16 X 1000	1p1q4t	1p2q2t	RX/TX	Fabric only

[*] Modules are only supported with Supervisor II.

As mentioned earlier, the Catalyst 6500 is capable of supporting QoS utilizing multiple modes of software. One software option is to support both the supervisor engine and MSFC with two separate images; this configuration is known as *Hybrid mode*. The alternative is to load one single image to support the entire platform. This software option is referred to as *Cisco IOS* (*Native mode*).

When running in Hybrid mode, the administrator must load one version of code for the supervisor module, referred to as the *Catalyst Operating System* (CatOS). If an MSFC exists on the supervisor, a separate version of IOS supporting the MSFC also needs to be loaded. The IOS on the MSFC is also responsible for supporting any FlexWAN modules and installed port adapters.

Native mode allows for the seamless integration of both the supervisor and MSFC, offering the administrator one command line to configure both components. This chapter provides numerous examples of the support for QoS in both Hybrid and Cisco IOS. These examples demonstrate QoS configurations utilizing both modes of operation.

Identifying the Catalyst Software

After the desired operational mode has been determined, it is necessary to download the appropriate software version. When running Hybrid, as already noted, the administrator must download separate images for the supervisor and MSFC components. For the supervisor engine, two software versions are available. One version supports the Supervisor I module, the other supports the Supervisor II module.

The fundamental difference between the two images is based on the numeric value prefaced by "cat6000-sup", which indicates the software is a supervisor image. If the string cat6000-sup is not immediately followed by a number, the software supports Supervisor I and IA engines. If the string is followed by a 2, however, the software supports all Supervisor II Engines. The same naming convention applies to images available for the MSFC. For 6500s utilizing an MSFC 1, the image name is preceded by "c6msfc". For switches equipped with an MSFC 2, the "c6msfc" string is immediately followed by the numeral 2.

The supervisor engine naming convention for Hybrid images is as follows:

- "cat6000-sup" indicates image supports Supervisor I/IA Engines.
- "cat6000-sup2" indicates image supports Supervisor II Engines.

The MSFC naming convention for Hybrid images is as follows:

- "c6msfc" indicates image supports MSFC 1.
- "c6msfc2" indicates image supports MSFC 2.

When running in Native mode, the image is bundled, incorporating the software essential for operating both the supervisor and the MSFC. In this instance, the string used to identify the version of code as a Cisco IOS image is "c6sup". Again, the numeric values immediately following the string specify the supervisor and MSFC versions supported by the image, respectively.

The naming convention for bundled Cisco IOS images is as follows:

- "c6sup11" indicates the image supports Supervisor I/IA engines and MSFC 1.
- "c6sup12" indicates the image supports Supervisor I/IA engines and MSFC 2.
- "c6sup22" indicates the image supports Supervisor II engine and MSFC 2.

NOTE

"c6sup" was the original designation used to identify the initial release of Cisco IOS for the Catalyst 6500. Images with this naming convention supported platforms with a Supervisor I Engine and MSFC 1.

Table 8-3 summarizes some of the fundamental differences applicable to operating in Hybrid and Native modes on the Catalyst 6500. For additional comparative information about CatOS and Cisco IOS, consult the following technical document at Cisco.com:

"White Paper: Comparison of the Cisco Catalyst and Cisco IOS Operating Systems for the Cisco Catalyst 6500 Series Switch"

www.cisco.com/warp/public/cc/pd/si/casi/ca6000/tech/catos_wp.pdf

Table 8-3. Default Differences Between Hybrid and Cisco IOS

Feature	Hybrid	Cisco IOS
Required software images	2 software images: Supervisor and MSFC.	1 software image: bundled Supervisor and MSFC.
Required configuration files	2 configuration files: Supervisor and MSFC.	1 configuration file: combined Supervisor and MSFC.
Default interface/port state	All ports default to Layer 2 switch ports.	All interfaces default to Layer 3 routed interfaces.
Default interface/port status	All ports default to enabled.	All interfaces default to administratively shut down.

Enabling QoS on the Switch

QoS first needs to be enabled globally on the switch, before this discussion turns to an explanation of the different QoS features. Example 8-1 shows how to initialize QoS on the Catalyst 6500 and shows the commands to verify the configuration.

Example 8-1. Enabling QoS on the Switch

```
(Hybrid)

hybrid(enable)>set qos enable

Command Verification:

hybrid(enable)>show qos status

   QoS is enabled on this switch.

(Native)

native(config)#mls qos

Command Verification:

native#show mls qos

   QoS is enabled globally

!

(text omitted)
```

After QoS has been enabled on the switch, the features can be configured. This chapter now focuses on the individual QoS mechanisms available on the Catalyst 6500. The first topic is input scheduling.

Input Scheduling

As shown in Table 8-1, the Catalyst 6500, through the use of individual port ASICs, can support inp
scheduling. This feature is beneficial if contention for the bus or switching fabric exists. At the time
instance, fabric-enabled cards have an 8-Gbps full-duplex connection to the switching fabric. Voice
application performance may be severely impacted if the receiving module is a WS-X6516, with all
receiving traffic at close to line rate. To ensure that voice traffic and other real-time applications m
required service levels, you can use input scheduling to assign traffic to the appropriate queues an
The recommendation is to assign voice traffic to the priority queue to minimize the end-to-end dela
To take advantage of this feature, however, the inbound port must be configured to trust the arrivi
view the scheduling capabilities of a particular port, use one of the following commands:

Example 8-2. Viewing Input Scheduling Capabilities

```
(Hybrid)

Show port capabilities [mod[/port]]

hybrid (enable) show port capabilities 4/1

Model                    WS-X6408A-GBIC

(text omitted)

QOS scheduling           rx-(1p1q4t),tx-(1p2q2t)

(text omitted)
```

or

```
(Native)

show interface {type num}capabilities

native#show interface gigabitEthernet 2/1 capabilities

  Model:                 WS-X6516-GBIC

(text omitted)

QOS scheduling:        rx-(1p1q4t), tx-(1p2q2t)

(text omitted)
```

or

```
show queueing interface {type num} [| include Receive]
```

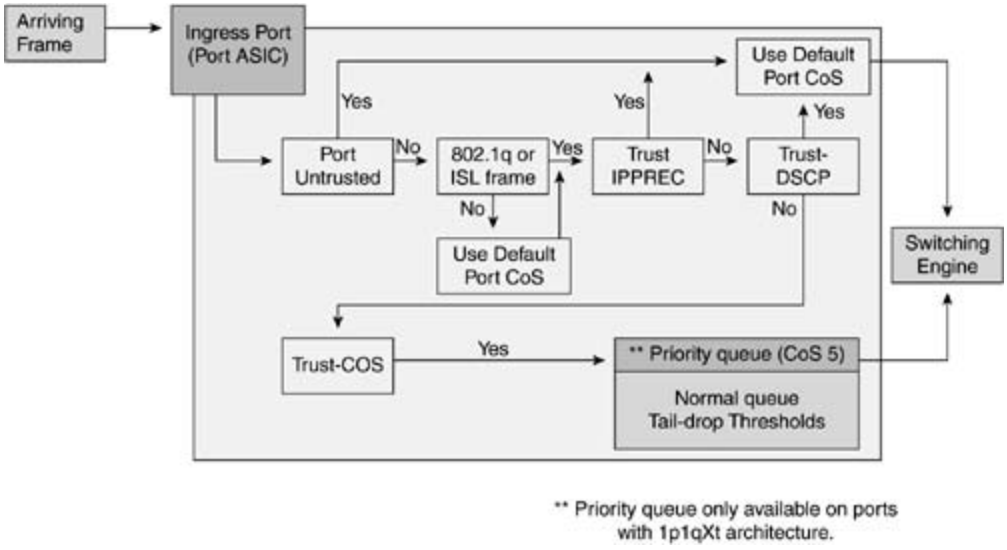
```
native#show queueing interface fastEthernet 6/1 | include Receive
  Receive queues [type = 1q4t]:
```

NOTE

The following section on classification covers the concept of trust and how it operates on the C 6500.

All other settings result in the arriving frames being placed in a default queue and forwarded direct switching engine based on FIFO. Figure 8-3 represents the decision path for an arriving frame.

Figure 8-3. Decision Path for an Ingress Frame



Because input scheduling is only applicable when the port is set to trust an ingress frame's CoS val the appropriate configuration has been made. If the arriving frame has an 802.1q or *Inter-Switch L* header, the port logic maintains the CoS value specified in the user priority field or the user field fc respective frames. For all other frames, the port ASIC uses the default CoS setting specified by the

for the given port.

NOTE

802.1q headers have a user priority field, and ISL headers contain a user field; these fields en
network administrator to specify that a certain stream of traffic should be handled more expec
However, frames not tagged with a trunk header lack any additional fields that accommodate
setting. As a result, all untagged frames utilize the default CoS value specified on the ingress
acquiring the default port CoS value, the derived value determines how the frame is processed
the switch. This setting may also impact how the frame is handled on an end-to-end basis acr
network.

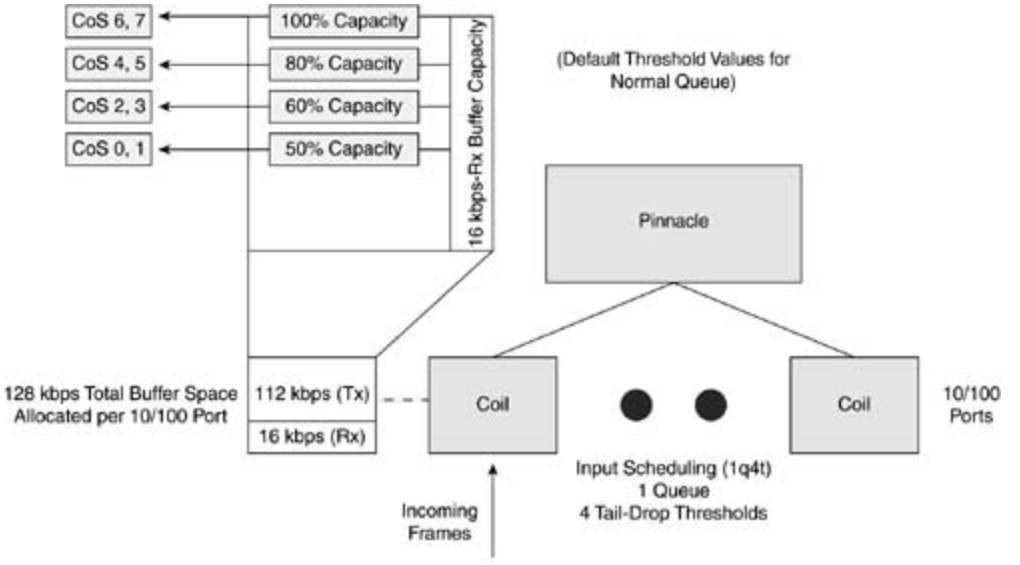
After the CoS value of an inbound frame has been determined, the frame is placed into the approp
For most 10/100 ports, there is one queue with four configurable tail-drop thresholds, designated a
additional information on queue nomenclature, refer to the section "Catalyst Feature Overview" in
"Overview of QoS Support on Catalyst Platforms and Exploring QoS on the Catalyst 2900XL, 3500X
Catalyst 4000 CatOS Family of Switches." Each CoS is mapped to one of these four thresholds. Tab
the exact CoS-to-threshold assignment for 1q4t port types. The exception to this is the fabric-enab
modules, whose ports offer an additional strict-priority queue for frames marked with CoS 5. These
are designated as 1p1q0t. This module is covered later in the section "Input Scheduling and Conge
Avoidance for 1p1q0t and 1p1q8t."

Similar to the non-fabric-enabled 10/100 ports, earlier Gigabit Ethernet ports could only provide a
with congestion avoidance handled by the four configurable tail-drop thresholds. (Refer back to Ta
view the queue architecture for the various modules.) However, recent Gigabit linecards offer the a
of a strict-priority queue, noted as 1p1q4t. The priority queue is responsible for transmitting delay-
critical network traffic, namely voice. Because the priority queue can starve remaining traffic, it is r
that only low-bandwidth traffic be placed in the strict-priority queue. By default, frames marked wi
mapped to the strict-priority queue. This ensures the expeditious handling of voice traffic, because
Phones mark voice frames with CoS 5. To minimize the delay and jitter, which impact voice quality
priority queue is provided immediate access to the switch backplane. If traffic exists in the strict-pr
it is serviced prior to any other queues. Contrary to the normal queue, however, the priority queue
possess a configurable threshold. When the strict-priority queue reaches 100-percent capacity, fra
discarded due to buffer exhaustion.

Figure 8-4 shows input scheduling on 10/100 ports from an architectural perspective.

Figure 8-4. Ingress QoS on the Coil ASIC

[View full size image]



NOTE

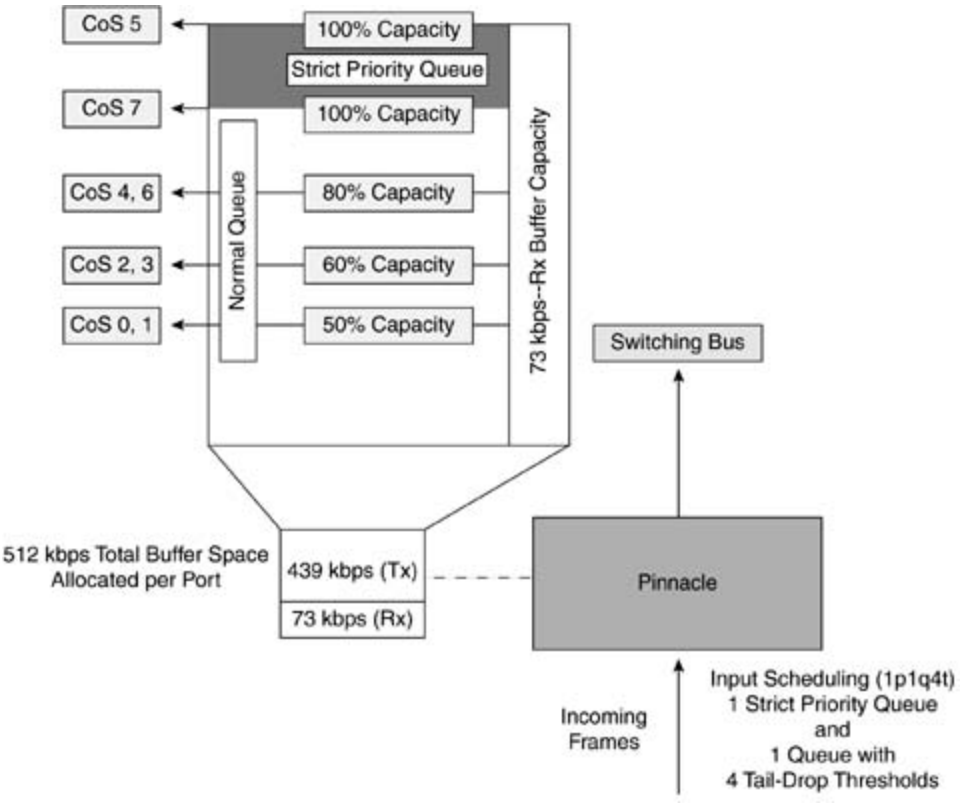
Figure 8-4 does not reflect the architecture of the fabric- enabled 10/100 modules. They use d ASICs and utilize a centralized buffer accessible to each port on the linecard. Although the arc is centralized, the net result is a finite buffer being statically allocated to each port.

As depicted in Figure 8-4, input queue scheduling and congestion management on the 10/100 mod accomplished by using the Coil ASIC.

The Pinnacle serves as the connection to the switching bus, and branches out to four Coil ASICs. Ea is responsible for servicing twelve 10/100 ports on a single linecard. The Coil is also responsible fo finite amount of buffer space to each port. As shown in the figure, each individual port receives 128 space. The figure depicts the architecture for both the WS-X6348 and WS-X6148. Each block of me further subdivided, leaving 112 KB for the transmit buffer and 16 KB for the receive buffer. Within space allocated for the receive queue, four tail-drop thresholds are specified. The different CoS valu mapped to these threshold levels.

Figure 8-5 shows the ASIC responsible for Gigabit Ethernet ports. In this case, the Pinnacle ASIC c Gigabit ports. Each port receives 512 KB of buffer space, which is subdivided into 439 KB for the tr and 73 KB for the receive side. For earlier Gigabit linecards that do not have a strict-priority queue architecture differs slightly and the default threshold settings and CoS mappings match the default the 10/100 ports, shown in Table 8-4. For ports with a strict-priority queue, the default differs sligh of being mapped to the third tail-drop threshold setting of 80 percent, all frames arriving with CoS to the priority queue by default. It is up to the network administrator to ensure that CoS 5 is, in fac traffic. The strict-priority queue then ensures that all arriving critical traffic is forwarded prior to tra normal queue. When the priority queue is depleted, the normal queue is serviced. Again, the priori only intended to accommodate low-bandwidth, low-latency traffic, such as voice. Continuous traffic the priority queue can potentially starve out lower queues. The priority queue does not have a conf threshold. Therefore as soon as the queue is full, it tail drops any excess frames. Figure 8-5 depicts queue, the default thresholds, and the CoS mappings. Now that input queue scheduling has been in focus turns to assigning receive queue drop thresholds and how to map CoS values to the threshol queues.

Figure 8-5. Ingress QoS on the Pinnacle ASIC



Configuring Receive Queue Tail-Drop Thresholds

To configure the different receive queue tail-drop thresholds and verify the configuration, use the f
commands.Table 8-4 provides the default threshold drop values and the CoS values mapped to the

(Hybrid)

set qos drop-threshold {*port-type*}**rx queue** {*queue#*} {*thr1*} {*thr2*} {*thr3*} {*thr4*}

show qos info {**runtime** {*mod/port*}}**|** {**config** {*mod/port*}**|** {{*port-type*}**rx**}}

(Native)

rcv-queue threshold 1 {*thr1*} {*thr2*} {*thr3*} {*thr4*}

show queueing interface {*type num*} [**| begin Receive**]

Table 8-4. Default Receive Queue Threshold Configuration

Standard Receive Queue Thresholds	Default Mapped CoS Values	Buffer Capacity Fille
Threshold 1	0,1	50
Threshold 2	2,3	60
Threshold 3	4,5[*]	80
Threshold 4	6[*],7	100
CoS 6 is mapped to threshold 3 if a strict-priority queue is present. All other threshold mappings r unchanged.		

[*] CoS 5, by default, is mapped to the strict-priority queue if present.

When configuring the receive queue thresholds in Hybrid, *port-type* is either 1q4t or 1p1q4t. Regar presence of a priority queue, the *queue#* is always 1, which indicates the normal queue. The tail-d for the strict-priority queue, which is represented by *queue#* 2, is set for 100 percent, which is not In both cases, *thr1* represents the first drop percentage, which by default is 50 percent and corresp values 0 and 1. *thr2* by default is set to 60 percent and maps to CoS 2 and 3. *thr3* defaults to 80 p corresponds to CoS 4 and 5, unless the port has a priority queue, in which case the CoS values are Finally,*thr4* defaults to 100 percent and maps to CoS 6 and 7; in the presence of a priority queue, default CoS 7 is the only value mapped to the final tail-drop threshold. rcv-queue threshold, whi configured under interface configuration mode, is the equivalent command in Cisco IOS. Verifying t configuration in Hybrid using the config*{port-type}* rx option enables the administrator to more ac display the desired results. Examples 8-3 and 8-4 demonstrate how to configure the drop threshol the configurations.

Example 8-3. Configuring Receive Queue Tail-Drop Thresholds in Hybrid

```
hybrid (enable) set qos drop-threshold 1p1q4t rx queue 1 65 75 85 100

Receive drop thresholds for queue 1 set at 65% 75% 85% 100%

hybrid (enable) show qos info config 1p1q4t rx

QoS setting in NVRAM for 1p1q4t receive:

QoS is enabled

Queue and Threshold Mapping for 1p1q4t (rx):

Queue Threshold CoS
```

```
----- --------- ---------------

1     1         0 1

1     2         2 3

1     3         4 6

1     4         7

2     -         5

Rx drop thresholds:

Queue #  Thresholds - percentage

-------  ------------------------------------

1        65% 75% 85% 100%

(text omitted)
```

Example 8-4. Configuring Receive Queue Tail-Drop Thresholds in Native I

```
native(config)#interface gigabitEthernet 1/1

native(config-if)#rcv-queue threshold 1 65 75 85 100

  threshold configured on:  Gi1/1 Gi1/2

native#show queueing interface gigabitEthernet 1/1 | begin Receive

  Receive queues [type = 1p1q4t]:

    Queue Id    Scheduling  Num of thresholds

    -----------------------------------------

       1         Standard            4

       2         Priority            1


    queue tail-drop-thresholds

    --------------------------

    1     65[1] 75[2] 85[3] 100[4]
```

```
(text omitted)
```

At the time of writing, only Gigabit Ethernet interfaces support configuring the receive queue thres running Native mode. Fast Ethernet interfaces default to 100 percent for all thresholds in the receiv Gigabit ports on the linecard must be configured to mls qos trust [cos], prior to altering the defa settings. After the thresholds have been adjusted, all ports inherit the configuration. For 16-port G ports 1 through 8 receive the same threshold and map configuration, whereas 9 through 16 receive settings. For switches running Hybrid, it is necessary to use set port qos [*mod*/*port*]trust [trust- activate and use the receive queue thresholds.

Mapping CoS to Queues and Drop Thresholds

After the different queue thresholds have been configured, you can map the various CoS values to tail-drop threshold. To map a CoS to a particular threshold and verify the configuration, use the fol commands:

```
(Hybrid)

set qos map {port-type}rx {queue#} {thr #}cos {cos-list}

show qos info {runtime {mod/port}} | {config {mod/port} | {{port-type} rx}}

(Native)

wrr-queue cos-map {queue-id} {thr #} {cos1 [cos2] [cos3] [cos4] [cos5] [cos6] [cos

rcv-queue cos-map {queue-id} {thr #} {cos1 [cos2] [cos3] [cos4] [cos5] [cos6] [cos

show queueing interface {type num} [ | begin Receive]

(Mapping CoS to the priority queue)

priority-queue cos-map 1 {cos1 [cos2] [cos3] [cos4] [cos5] [cos6] [cos7] [cos8]}
```

After all the ports have been configured to trust the ingress CoS, the default threshold mappings ta described in the preceding section. When mapping to receive thresholds in Hybrid mode, *port-type* 2q2t, 1p1q4t, 1p1q0t, or 1p1q8t. (1p1q0t and 1p1q8t are addressed in the next section). The 2q2t keyword is used to configure the CoS mappings for both 1q4t receive and 2q2t transmit queues. In queue 1 and thresholds 1 and 2 represent queue 1 and thresholds 1 and 2 for 1q4t port types. Hov and thresholds 1 and 2 represent queue 1 and thresholds 3 and 4. This behavior is discussed later

in the section titled "Mapping CoS Values to Transmit Queues and Thresholds" and demonstrated i
35.

The equivalent command in Cisco IOS for configuring 1q4t port types is wrr-queue cos-map. Aga
2q2t configuration principle applies. With the exception of the 2q2t port types, *queue#* or *queue-id*
CoS is to be mapped to either the standard normal queue, specified as 1, or the priority queue, spe
thr # designates the threshold within the applicable queue the CoS is mapped to. Because there is
threshold for the priority queue, the appropriate parameter is 1. Finally, the *cos-list* enables the ad
specify a specific CoS or range. When using the priority-queue cos-map command, the administ
specify the keywords receive or transmit. Therefore, all CoS values mapped to the priority queue
directions.Example 8-5 demonstrates mapping a CoS to the normal queue and threshold and the p
and also verifies the configuration in Hybrid. Example 8-6 shows the same configuration with Nativ

Example 8-5. Mapping a CoS to a Queue and Threshold in Hybrid Mode

```
hybrid (enable) set qos map 1p1q4t rx 1 4 cos 6

QoS tx priority queue and threshold mapped to cos successfully.

hybrid (enable) set qos map 1p1q4t rx 2 1 cos 7

QoS tx priority queue and threshold mapped to cos successfully.

hybrid (enable) show qos info config 1p1q4t rx

QoS setting in NVRAM for 1p1q4t receive:

QoS is enabled

Queue and Threshold Mapping for 1p1q4t (rx):

Queue Threshold CoS

----- --------- ---------------

1     1         0 1

1     2         2 3

1     3         4

1     4         6

2     -         5 7

(text omitted)
```

InExample 8-5, CoS 6 is changed from its default mapping in the normal queue, previously thresh
mapped to threshold 4. CoS 7 is mapped to the priority queue. The following example demonstrate
parameters in Native mode.

Example 8-6. Mapping a CoS to a Queue and Threshold in Native Mode

```
native(config-if)#rcv-queue cos-map 1 4 6
  cos-map configured on:  Gi2/1 Gi2/2 Gi2/3 Gi2/4 Gi2/5 Gi2/6 Gi2/7 Gi2/8
native(config-if)#priority-queue cos-map 1 7
  cos-map configured on:  Gi2/1 Gi2/2 Gi2/3 Gi2/4 Gi2/5 Gi2/6 Gi2/7 Gi2/8
native#show queueing interface gigabitEthernet 2/1 | begin Receive
  Receive queues [type = 1p1q4t]:
    Queue Id    Scheduling  Num of thresholds
    -----------------------------------------
       1         Standard            4
       2         Priority            1

    queue tail-drop-thresholds
    --------------------------
    1     50[1] 60[2] 80[3] 100[4]
    queue thresh cos-map
    ---------------------------------------
    1     1      0 1
    1     2      2 3
    1     3      4
    1     4      6
    2     1      5 7
(text omitted)
```

The receive queue size ratio enables the administrator to set a ratio between the amount of traffic the strict-priority queue and the amount received by the normal queue. Currently, only two module 6500 support altering this feature. They are the fabric-enabled 10/100 modules and the 10 Gigabit

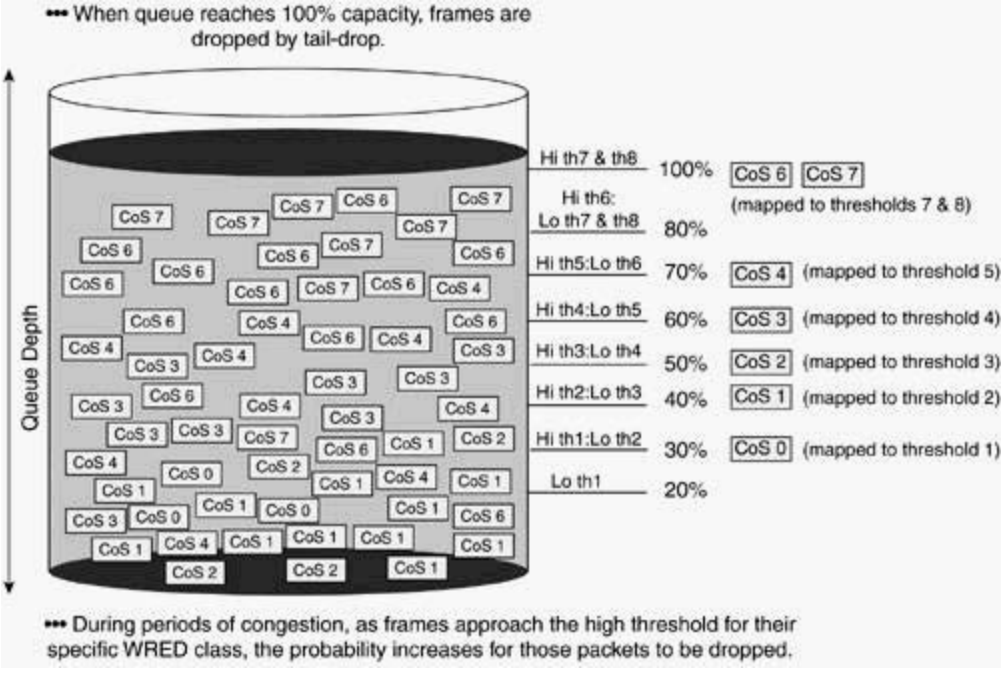
modules. The port types for these modules are 1p1q0t and 1p1q8t, respectively. The following sect input scheduling and congestion avoidance for these port types and discusses how to alter the rece ratio.

Input Scheduling and Congestion Avoidance for 1p1q0t and 1p1q8t

In addition to the two port types 1p1q4t and 1q4t, two additional port types are available on other linecards, 1p1q0t and 1p1q8t. These two options are available to ports on the fabric-enabled 10/1(and the 10 Gigabit Ethernet modules, respectively. Mentioned earlier, the 1p1q0t port type, similar 1p1q4t, has a strict-priority queue and one normal queue. Unlike the 1p1q4t, however, the 1p1q0t only a single nonconfigurable tail-drop threshold. The 10/100 fabric-enabled module has a nonbloc connection to the switch fabric. This implies there is less of an opportunity for congestion to occur (because contention for the switching fabric is reduced. As a result, these ports do not possess a cor drop threshold and start discarding traffic when the buffers reach 100-percent capacity.

The 1p1q8t port type on the 10 Gigabit modules offers an additional feature not available on any o previously mentioned. Although it has two queues, both a strict-priority queue, and a configurable queue, this port type offers the option of configuring eight WRED thresholds as a congestion avoida mechanism within the normal receive queue. The port ASIC available on the 10 Gigabit module ma possible. WRED offers a more deterministic method of dropping lower-priority frames, and ensurin priority flows are given preference, as frames accumulate within the queue. Refer to Chapter 2 for detailed description of WRED operation. Figure 8-6 shows WRED operation on 1p1q8t port types.

Figure 8-6. WRED Operation for 1p1q8t Port Types



The 1p1q8t port type has eight instances of WRED, each instance having both a lower and an uppe The lower threshold, also referred to as the *low-water mark*, ranges from 0 percent to 100 percent normal queue size. The upper threshold, also referred to as the *high-water mark*, ranges from 1 pe

percent of the total queue. When configuring the values, the lower threshold must be less than or e
higher threshold. The low-water mark enables the administrator to specify a lower bound, indicatir
to randomly drop frames in times of congestion for a particular CoS mapped to that specific WRED
high-water mark, on the other hand, specifies at what percentage point during congestion it is perı
drop all traffic with the appropriate CoS. During congestion, if the queue fills to a point that falls w
configured WRED range, frames with the applicable CoS are randomly dropped. The drop probabili
CoS values progressively increases as the queue fills, until the level of congestion reaches the uppe
range, at which point all frames with the corresponding CoS are tail dropped. WRED is discussed ir
during the section on output scheduling. The following commands are used to configure the WRED
1p1q8t port types:

(Hybrid)

set qos wred {**1p1q8t**}**rx queue** {*queue#*} {[*thr1Lo*:]*thr1Hi* [*thr2Lo*:]*thr2Hi ... [thr8L*

show qos info {**runtime** {*mod/port*}}**|** {**config** {*mod/port*}**|** {{*port-type*}**rx**}}

(Native)

wrr-queue random-detect {**min-threshold**} {*queue#*} {*thr1% thr2% thr3% thr4% ... thr8*

wrr-queue random-detect {**max-threshold**} {*queue#*} {*thr1% thr2% thr3% thr4% ... thr8*

show queueing interface tengigabitethernet {*slot/port*}

In Hybrid, you can configure all the thresholds with one command. Because there is only one norm
queue# is {1}. When configuring the threshold percentages, if a lower threshold is not explicitly cc
lower bound defaults to zero. Therefore, if only a single value is specified for a threshold, that valu
configured as the upper bound. The thresholds depicted in Figure 8-6 are configured as follows:

set qos wred 1p1q8t rx queue 1 20:30 30:40 40:50 50:60 60:70 70:80 80:100 80:100

In Native mode, the administrator uses two separate commands, one to configure all eight of the n
thresholds for the various WRED classes and the other to configure the eight upper-bound threshol

both instances the *queue#* is configured as {1}.Examples 8-7 and 8-8 demonstrate how to configu
receive queue WRED thresholds using Hybrid and Native modes, respectively.

Example 8-7. Configuring Receive Queue WRED Thresholds for 1p1q8t in Mode

```
hybrid (enable) set qos wred 1p1q8t rx queue 1 20:35 30:45 40:55 50:65 60:75 70:85
➥80:100
   WRED thresholds for queue 1 set to 20:35 and 30:45 and 40:55 and 50:65 and 60:7
➥5 and
   80:100 and 80:100 on all WRED-capable 1p1q8t ports.
hybrid (enable) show qos info config 1p1q8t rx
(text omitted)
Rx WRED thresholds:
Queue #  Thresholds - percentage
-------  ---------------------------------------
1        20%:35% 30%:45% 40%:55% 50%:65% 60%:75% 70%:85% 80%:100% 80%:100%
(text omitted)
```

Example 8-8. Configuring Receive Queue WRED Thresholds for 1p1q8t in Mode

```
native(config-if)#rcv-queue random-detect min-threshold 1 20 30 40 50 60 70 80 80
native(config-if)#rcv-queue random-detect max-threshold 1 35 45 55 65 75 85 100 10
native#show queueing interface tenGigabitEthernet 8/1 | begin Receive
  Receive queues [type = 1p1q8t]:
    Queue Id    Scheduling  Num of thresholds
    -----------------------------------------
    queue random-detect-min-thresholds
```

```
    ---------------------------------

      1    20[1] 30[2] 40[3] 50[4] 60[5] 70[6] 80[7] 80[8]

    queue random-detect-max-thresholds

    ---------------------------------

      1    35[1] 45[2] 55[3] 65[4] 75[5] 85[6] 100[7] 100[8]

(text omitted)
```

After the WRED thresholds for the receive queue have been established, the CoS values are then m
desired threshold. The same configuration commands demonstrated at the beginning of the sectior
CoS to Queues and Drop Thresholds" are applicable in this situation. Unlike the previous examples,
number of available thresholds has now increased to eight. Therefore, the valid values for the *thr #*
range from 1 to 8. Additionally, when configuring the commands in Hybrid, the *port-type* in this ins
1p1q8t. Finally, the configuration for priority queue mappings is identical to that specified in the pr
section.

Configuring the Receive Queue Size Ratio

The 1p1q0t and 1p1q8t port types support configuring the receive queue size ratio for the normal a
priority queues. By default, the normal queue is assigned 80 percent of the total buffer space, and
priority queue is provided the remaining 20 percent. To alter and verify the percentage of memory
the normal and priority queues, use the following commands:

```
(Hybrid)

set qos rxq-ratio {1p1q0t | 1p1q8t} {queue1-ratio} {queue2-ratio}

show qos info {runtime {mod/port}}| {config {mod/port}| {{port-type}rx}}

(Native)

receive-queue queue-limit {normal-queue ratio} {priority-queue ratio}

show queueing interface {type num} [ | include  queue-limit ratios]
```

When configuring the command in Hybrid mode, *queue1-ratio* represents the percentage of total b

assigned to the normal queue. *queue2-ratio* is the percentage assigned to the strict-priority queue. using Hybrid or Native mode, the values assigned to the normal queue and the priority queue mus 100 percent, and both individual values must be greater than zero. Prior to modifying default recei ratios, carefully consider how a change will impact traffic behavior. Example 8-9 shows configurati receive queue ratio size in Hybrid mode; Example 8-10 shows the same with Cisco IOS.

Example 8-9. Configuring the Receive Queue Size Ratio in Hybrid Mode

```
hybrid (enable) set qos rxq-ratio 1p1q0t 75 25

QoS rxq-ratio is set successfully.

hybrid (enable) show qos info config 1p1q0t rx

(text omitted)

Rx queue size ratio:

Queue #  Sizes - percentage

-------  -----------------------------------

1        75%

2        25%
```

Example 8-10. Configuring the Receive Queue Size Ratio in Native Mode

```
native(config-if)#rcv-queue queue-limit 75 25

  queue-limit configured on all 48 ports on slot 3.

native#show queueing interface fastEthernet 3/1 | include queue-limit

    queue-limit ratios:     75[queue 1]  25[queue 2]
```

Classification and Marking

Classification and marking on the Catalyst 6500 is performed by port ASICs and the PFC. Given the different capabilities, this section is broken down into the following topics:

- Port-Based Versus VLAN-Based QoS
- Classification Based on Destination MAC Address and VLAN
- Trust
- Classification and Marking Using ACLs and ACEs

In addition to the these listed topics, further attention is directed toward Cisco's *Modular QoSCLI* (MQC), which was introduced in Chapter 5, "Introduction to the Modular QoS Command-Line Interface." As before, numerous examples are provided demonstrating how these features are configured when operating in both Hybrid and Native modes.

Port-Based Versus VLAN-Based QoS

On the Catalyst 6500, QoS policies can be deployed at the individual port level or they can be applied on a per-VLAN basis. By default, all ports are configured for port-based QoS. This implies all classification policies, and marking and policing, configured for the port are only applicable to the port to which they are applied. Parameters are not shared by neighboring interfaces. On the contrary, VLAN-based QoS enables the administrator to apply all settings to all ports configured for a specific VLAN. VLAN-based QoS may be preferred in certain situations. When deploying voice in a campus environment, for example, consideration needs to be given to voice-control traffic. Although not as important as the actual voice frames, the control streams are responsible for establishing, maintaining, and terminating calls. Therefore, it is recommended to assign the control traffic a higher priority than normal user data. Because voice traffic is assigned to auxiliary VLANs, marking the control traffic is easily accomplished using VLAN-based QoS. You can use ACLs to identify the appropriate Layer 3 and 4 parameters, and then map them to the appropriate auxiliary VLAN. Additional instances when VLAN-based QoS can be used include when policing inter-VLAN traffic for specific VLANs and when establishing initial QoS policies for access layer devices with multiple VLANs. The latter scenario may be required, depending on the QoS capabilities of the access layer device. To alter the port policy to either port based or VLAN based, use the following commands:

```
(Hybrid)

set port qos {mod/port} {port-based | vlan-based}

(Native)
```

```
[no]mls qos vlan-based
```

> **NOTE**
>
> Linecards equipped with a DFC support only port-based policing. VLAN-based policing is currently not supported on DFCs. It will however, be supported on the DFC3, on a per-DFC3 basis.

If a port is configured for VLAN-based QoS, any ACLs applied at the port level are ignored. The opposite applies to ports configured for port-based QoS; ACLs applied at the VLAN level are ignored. As a result, it is vital to properly set the port to port-based or VLAN-based QoS in an effort to achieve the desired behavior. When configuring a port to be VLAN based in Cisco IOS, it is necessary to configure the port as a Layer 2 port. Therefore, prior to altering the interface policy,`switchport` must be entered while in interface configuration mode. The command `switchport` alters the port state from the default Layer 3 setting to Layer 2. By default, all ports on a 6500 with Cisco IOS are routed interfaces. The next two examples show how to successfully change a port's setting to VLAN based and how to verify the configuration.

Example 8-11. Configuring a Port to VLAN Based in Hybrid Mode

```
hybrid (enable) set port qos 5/30 vlan-based

QoS interface is set to vlan-based for ports 5/30

hybrid (enable) show qos info config 5/30

(text omitted)

Tx port type of port 5/30 : 2q2t

Rx port type of port 5/30 : 1q4t

Interface type: vlan-based

(text omitted)
```

Example 8-12. Configuring a Port to VLAN Based in Native Mode

```
native(config)#interface fastEthernet 6/35
```

```
native(config-if)#mls qos vlan-based

vlan-based mode is only for Layer 2 LAN interfaces.

native(config-if)#switchport

native(config-if)#mls qos vlan-based

native#show mls qos | begin vlan-based

  QoS is vlan-based on the following interfaces:

    Gi1/1 Gi5/1 Fa6/1 Fa6/2 Fa6/34 Fa6/35

(text omitted)
```

Classification Based on Destination MAC Address and VLAN

A Catalyst 6500 without a PFC or an MSFC, but equipped with a Layer 2 switching engine, can classify and mark based on the destination MAC address and VLAN of an arriving frame. With this hardware configuration, the switching engine can support only Layer 2 CoS values. Therefore, this feature enables the administrator to map a specific CoS value to a host destination MAC address and VLAN pair. The following command is used to map the appropriate CoS (available in Hybrid only):

```
set qos mac-cos {dest MAC-addr} {dest VLAN} {CoS value}

show qos mac-cos {all |dest MAC-addr}
```

Unfortunately, this feature does not scale very well. Due to the vast number of MAC addresses in a given network, the configuration process could be administratively intensive and difficult to manage and maintain. If you include a PFC in the hardware profile of the switch, however, you can use the set qos acl command as an alternative to the preceding commands. The following example demonstrates how to map a CoS value to a destination MAC address and VLAN.

Example 8-13. Mapping the CoS Value to a Destination MAC Address and VLAN

```
hybrid (enable) set qos mac-cos 00-02-a5-07-83-0e 1 3

CoS 3 is assigned to 00-02-a5-07-83-0e vlan 1.

hybrid (enable) show qos mac-cos all

Number  VLAN  Dest MAC            CoS

------  ----  ------------------  ---

1       20     11-22-33-44-55-66  5

2       1      00-02-a5-07-83-0e  3
```

Trust

Trust is a concept covered in Chapter 2, in the section "Catalyst QoS Trust Concept." Trust is a classification method that enables the administrator to maintain a frame's CoS/ToS (*type of service*) settings. Upon ingress, the administrator can determine to further utilize the predetermined priority settings or can apply new settings to arriving frames on a port-by-port or VLAN basis. Trust is extremely important in determining a frame's behavior as it traverses the switch. The trust state of a given port determines how frames are classified, marked, and, as previously discussed, scheduled as they proceed from the ingress to egress port within the switch.

On the 6500, all ports by default are configured as untrusted. The exception to this is FlexWAN ports, which are trusted by default. For dot1q- and ISL-tagged frames arriving at an untrusted port, even if they possess a CoS or ToS setting greater than zero, those frames are reset with the default CoS configured for the respective port of entry. By default, this value is zero, but can be changed at the discretion of the administrator. To modify the default CoS setting for a port, use the following commands:

```
(Hybrid)

set port qos {mod/port}cos {CoS value}

show qos info {runtime | config} {mod/port}

(Native)

mls qos cos {CoS value}

show queueing interface {type num} [ | include Default COS]
```

The CoS setting derived from the port is then mapped to a corresponding internal DSCP value, which is used internally to determine the handling of the frame through the switch. For more detailed information, see the section titled "CoS-to-DSCP Mapping" in this chapter. (Refer to Table 8-12 for the default mapping tables within the Catalyst 6500.)

Similar to the Catalyst 3550, the 6500 is capable of trusting incoming frames based on their CoS, IP precedence, or DSCP values. Table 8-5 shows the trust states supported on the Catalyst 6500. The table correlates each module with the supported trust states.

Table 8-5. Overview of Supported Trust States and Modules

Module	Trust CoS	Trust IP Precedence	TrustDSCP
WS-X6024	Yes	No	No
WS-X6148	Yes	Yes	Yes
WS-X6224	Yes	No	No
WS-X6248	Yes	No	No
WS-X6316	Yes	Yes	Yes
WS-X6324	Yes	No	No
WS-X6348	Yes	No	No
WS-X6408	Yes	Yes	Yes
WS-X6408A	Yes	Yes	Yes
WS-X6416	Yes	Yes	Yes
WS-X6501	Yes	Yes	Yes
WS-X6502	Yes	Yes	Yes
WS-X6516	Yes	Yes	Yes
WS-X6524	Yes	Yes	Yes
WS-X6548	Yes	Yes	Yes
WS-X6816	Yes	Yes	Yes

trust-cos enables the administrator to trust the inbound CoS setting contained within the dot1q or ISL header. For all untagged frames or non-IP packets entering the switch, if a port is set to trust-cos all traffic entering the switch is set with the default CoS configured for the port. In addition, as mentioned in the preceding section, to take advantage of input scheduling on the 6500 the port must be configured to trust the inbound CoS. To enable a port to trust the inbound CoS, use the following commands:

```
(Hybrid)

set port qos {mod/port}trust {trust-cos}

show qos info {runtime | config} {mod/port}

(Native)

mls qos trust {cos}

show queueing interface {type num}
```

Under some circumstances, the administrator must perform additional configuration steps, beyond what is noted in the preceding commands. The WS-X6224/6248 and WS-X6324/6348 have a hardware restriction that prevents them from passing the learned CoS to the switching engine. Although the linecards recognize the arriving CoS, and use that value for input scheduling, when the frame header is passed to the switching engine for forwarding, the CoS value for the frame is not preserved, and as a result the CoS is rewritten to zero. Therefore, it is necessary to configure additional commands.

> NOTE
>
> The WS-X6148 linecard is not subject to the same hardware limitations as the WS-X6224/6248 and WS-X6324/6348. It is capable of trusting inbound CoS values without configuring additional commands. Also unlike the WS-X6224/6248 and Ws-X6324/6348, the WS-X6148 supports inbound trust for IP precedence and for DSCP values. At the time of writing, this support is found in CatOS Release 6.4(1) and subsequent 6.4 releases.

Example 8-14 shows the resulting message after setting the port to trust-cos. As described, the receive thresholds are enabled on the port. This allows inbound frames to utilize the ingress port scheduling. However, the port remains in the untrusted state. The example further includes the additional commands required to ensure the arriving CoS is sustained through the switch.

Example 8-14. Enabling trust-cos on WS-X6224/6248 and WS-X6324/6348 in Hybrid Mode

```
hybrid (enable) set port qos 6/1 trust trust-cos

Trust type trust-cos not supported on this port.
```

```
Receive thresholds are enabled on port 6/1.

Port  6/1 qos set to untrusted.
```

set qos acl {**ip**} {*acl-name*} {**trust-cos**} {**any**}

commit qos acl {*acl-name*}

set qos acl map {*acl-name*} {*mod/port*|*vlan*}

For the fabric-enabled 10/100 modules, it is not necessary to make additional configuration steps. In addition, the fabric-enabled 10/100 linecard is the only 10/100 card that supports port-based trust in Native mode. The default setting for all ports is untrusted, which can only be modified on Gigabit Ethernet and fabric-enabled 10/100 ports when running Cisco IOS. The same situation applies to ports configured for VLAN-based QoS.

trust-ipprec and trust-dscp instruct the switch to analyze the ToS field in the IP header of arriving packets. When configured, the IP precedence option accepts the three least significant bits in the precedence field within the ToS byte, whereas the DSCP option utilizes the 6 bits reserved for the codepoint value within the ToS byte. Configuring a port to trust the IP precedence or DSCP is accomplished as follows:

(Hybrid)

set port qos {*mod/port*}**trust** {**trust-ipprec | trust-dscp**}

show qos info {**runtime | config**} {*mod/port*}

(Native)

mls qos trust {**ip-precedence | dscp**}

show queueing interface {*type num*} [**| include Trust state**]

Modules with 1q4t port types, with the exception of Gigabit modules, do not support the IP precedence or DSCP trust option. This restriction applies to Catalyst 6500s running both Hybrid and Native modes. After the arriving frame or packet has been classified, using either the predetermined IP precedence or DSCP setting, that value is then used to derive the internal DSCP utilized by the switch. Once again, if the ingress port is configured to trust IP precedence, DSCP, or is instructed not to trust any Layer 2 or 3 priority settings (untrusted), the frame is passed directly to the switching engine. As a result, the arriving frame is not processed through

the input scheduling and queuing mechanisms configured for the port. To revert a port to the defaultuntrusted state, in Hybrid it is necessary to configure set port qos {*mod*/*port*}trust untrusted. In Native mode, the command is no mls qos trust.

NOTE

Regardless of the port trust state, each arriving frame is assigned a CoS value, which is forwarded to the switching engine within the D-bus header. If the incoming frame arrives on a dot1q or ISL trunk, the CoS value is derived from the priority of the marked frame. Otherwise, the default port CoS is assigned to the D-bus header. This applies to all traffic types.

The following examples demonstrate how to modify the trust state on Gigabit Ethernet and non-1q4t port types.

Example 8-15. Altering the Default Trust State in Hybrid Mode

```
hybrid (enable) set port qos 4/4 trust trust-cos

Port  4/4 qos set to trust-cos.

hybrid (enable) set port qos 4/5 trust trust-dscp

Port  4/5 qos set to trust-dscp.

hybrid (enable) show qos info config 4/4

QoS setting in NVRAM:

QoS is enabled

Policy Source of port 4/4: COPS

Tx port type of port 4/4 : 1p2q2t

Rx port type of port 4/4 : 1p1q4t

Interface type: port-based

ACL attached:

The qos trust type is set to trust-cos

(text omitted)

hybrid (enable) show qos info config 4/5

QoS setting in NVRAM:
```

```
QoS is enabled

Policy Source of port 4/5: COPS

Tx port type of port 4/5 : 1p2q2t

Rx port type of port 4/5 : 1p1q4t

Interface type: port-based

ACL attached:

The qos trust type is set to trust-dscp.

(text omitted)
```

Example 8-16. Altering the Default Trust State in Native Mode (WS-X6548)

```
native(config)#interface fastEthernet 3/12

native(config-if)#mls qos trust ip-precedence

native#show queueing interface fastEthernet 3/12 | include Trust state

  Trust state: trust IP Precedence
```

Examples 8-17 and 8-18 show how to configure a port back to its default untrusted state. The commands apply to all ports, regardless of interface port type.

Example 8-17. Reverting to the Default Trust State in Hybrid Mode

```
hybrid (enable) set port qos 4/4 trust untrusted

Port  4/4 qos set to untrusted.

hybrid (enable) show qos info config 4/4

QoS setting in NVRAM:

QoS is enabled

Policy Source of port 4/4: COPS
```

```
Tx port type of port 4/4 : 1p2q2t

Rx port type of port 4/4 : 1p1q4t

Interface type: port-based

ACL attached:

The qos trust type is set to untrusted.

(text omitted)
```

Example 8-18. Reverting to the Default Trust State in Native Mode (WS-X6548)

```
native(config)#interface fastEthernet 3/12

native(config-if)#no mls qos trust

native#show queueing interface fastEthernet 3/12

Interface FastEthernet3/12 queueing strategy:  Weighted Round-Robin

  Port QoS is enabled

  Port is untrusted

(text omitted)
```

Classification and Marking Using ACLs and ACEs

In addition to port trust states, the Catalyst 6500 utilizes ACLs to classify traffic flows, to enforce administratively defined QoS policies. Equipped with a PFC, the Catalyst 6500 can classify, and subsequently mark and police traffic, based on Layer 2, 3, and 4 characteristics. Policing is covered later in the policing section.

QoS ACL lookups are performed in hardware. The TCAM is the portion of memory responsible for storing all defined masks and ACL components for the switch. QoS ACLs are composed of individual ACEs. In turn, each ACE is composed of a pattern value and a mask value. The pattern value specifies the actual source and destination address, including any additional Layer 3- and 4-specified parameters. The mask in this case is defined as the portions of the pattern value that are specifically matched, or ignored.

Because the TCAM is a finite memory space, the number of configured masks and ACEs is also finite. For the Supervisor I Engine, the number of masks is limited to 2000. However, each mask can be shared among a maximum of eight ACEs. As a result, there are 16,000 possible supportable entries for the Supervisor I, which are shared among QoS and security ACLs. With a

Supervisor II Engine the number of masks and entries increases. The Supervisor II supports 4000 different masks, and is still capable of associating 8 ACEs with 1 mask entry. Therefore, the number of entries increases to 32,000. Unlike the Supervisor I, however, these entries are dedicated to QoS.

Each supervisor engine can support up to 512 different ACL labels system wide. Labels are divided among QoS and security ACLs. ACL label resources are consumed based on ACL type and how they are applied. QoS ACLs can only be applied to a port/interface or VLAN. In either instance, the result is the use of one label. Prior to CatOS Software Release 6.3, 250 labels are reserved for QoS type ACLs. Software Release 6.3 and later increases the number of supported QoS ACLs from 250 to 500. For additional information regarding the TCAM, refer to the section "Catalyst 2950 and Catalyst 3550 Family of Switches QoS Architectural Overview" in Chapter 6.

Default ACLs

Aside from administratively defined ACLs, the Catalyst 6500 utilizes three default ACLs to ensure all ingress traffic is classified. These ACLs are applied to all ports/interfaces on the switch. Any traffic not matching a named ACL applied to a port or VLAN interface, matches one of the default ACLs. One ACL is reserved for IP traffic, designated with ethertype 0x800, another for IPX traffic, ethertype 0x8137 and 0x8138, and finally an ACL for all MAC traffic, which encompasses all other ethertype values, including AppleTalk, DECnet, Banyan VINES and, *Xerox Network Systems* (XNS). For all arriving frames, the ethertype value is used to determine which ACL is applied to the arriving frame. Each default ACL has only one ACE. Although the match criteria are not configurable, the default marking value can be altered. The exception to this rule applies to Catalyst 6500s equipped with a Supervisor II Engine. With a PFC2, it is not possible to the default IPX or MAC value. This limitation does not impact IP traffic. By default, all traffic not matched by a named ACL is matched by one of the default ACLs and assigned an internal DSCP value of zero. Example 8-19 demonstrates how to modify the default DSCP value.

Example 8-19. Modifying the Default Marking Value with a PFC2

```
hybrid (enable) set qos acl default-action ip dscp 16 rx

IP ACL is set successfully.

hybrid (enable) show qos acl info default-action all

QoS default ACL on rx side:

set qos acl default-action

-------------------------

ip dscp 16

ipx

mac

(text omitted)
```

InExample 8-19 all ingress IP traffic to the 6500, which does not match a named ACL, is marked with a DSCP value of 16. Because the configuration in Example 8-19 was performed on a switch equipped with a PFC2, it was not possible to modify the DSCP values for IPX and MAC traffic. Example 8-20 shows the same procedures with a PFC1. In this instance, however, the default value for IPX and MAC traffic is modified.

Example 8-20. Modifying the Default Marking Value with a PFC1

```
hybrid (enable) set qos acl default-action mac dscp 8

QoS default-action MAC ACL is set successfully.

hybrid (enable) set qos acl default-action ipx dscp 32

QoS default-action IPX ACL is set successfully.

hybrid (enable) show qos acl info default all

QoS default ACL on rx side:

set qos acl default-action

--------------------------

ip dscp 0

ipx dscp 32

mac dscp 8
```

When running Cisco IOS, it is not possible to alter the default ACL marking parameters. The values for all traffic types are statically defined to zero. This is verified by performing a show mls qos {ip | ipx | mac} at the command line:

```
native#show mls qos ip

 QoS Summary [IP]:      (* - shared aggregates, Mod - switch module)

       Int Mod Dir  Class-map DSCP  AgId Trust FlId   AgForward-Pk   AgPoliced-Pk
```

```
--------------------------------------------------------------------------------
        All  1    -    Default    0    0*    No    0        8920394             0
native#
```

As shown in the preceding command output, a default class map is applied to all interfaces and marks all unmatched traffic with an internal DSCP value of zero. The same default behavior applies to all traffic types: IP, IPX, or MAC.

Defining and Configuring QoS ACLs in Hybrid Mode

As stated earlier, it is possible to classify and mark traffic on the Catalyst 6500 by applying administratively defined named ACLs. Each ACL contains one or more ACEs, which define the Layer 2, 3, and 4 parameters used to classify and subsequently mark or police network traffic. Similar to the default ACLs, the administrator has the option of configuring three ACL types: IP, IPX, or MAC. Each inbound frame is compared against the defined classification criteria specified in the ACEs. After a match has been established, the configured action is taken and no further ACEs are processed. Again, all traffic not matched by any of the named ACLs is matched by one of the default ACLs and marked with the associated default internal DSCP value. This section addresses configuring QoS ACLs in Hybrid mode, and then transitions to configuring ACLs with Native mode using class maps and policy maps.

Classification with ACLs in Hybrid mode is accomplished with the command set qos acl {ip | ipx |mac}. When performing classification utilizing IP ACLs, it is possible to classify traffic based on the following characteristics: Layer 3 source and destination address, Layer 3 precedence values, Layer 4 protocol IDs, Layer 4 TCP and UDP port numbers, Layer 4 *Internet Control Message Protocol* (ICMP) type and code values, and Layer 4 *Internet Group Management Protocol* (IGMP) message types. When configuring the various Layer 3 and 4 parameters, if a value or keyword is not specified for the protocol ID, port numbers, or message types, the PFC classifies all traffic matching the generic Layer 3 or 4 property indicated in the ACE. If an IP ACE is configured to match TCP or UDP traffic, for instance, if a particular port number or keyword is not specified, the ACE is matched against all TCP or UDP traffic, regardless of port. The same behavior applies to IP ACEs configured with the icmp or igmp keywords. If a particular message or code type is not indicated, all ICMP or IGMP traffic is matched. Additionally, if IGMP snooping is enabled on the switch, QoS does not support IGMP type traffic. Tables 8-6 through 8-8 identify the most common criteria used to classify IP traffic flows.

Table 8-6. Layer 4 IP Protocol Criteria

Protocol ID Keyword	Protocol ID Keyword	Protocol ID Keyword
Value	Value	Value
AHP	IGMP	PCP
(51)	(2)	(108)
EIGRP	IGRP	PIM
(88)	(9)	(103)
ESP	IP	TCP
(50)	(0)	(6)
GRE	IpinIP	UDP
(47)	(4)	(17)
IGRP	NOS	Protocol ID
(9)	(94)	Range (0–255)
ICMP	OSPF	
(1)	(89)	

Table 8-7. Layer 4 UDP Criteria

Port Keyword	Port Keyword	Port Keyword	Port Keyword
Port	Port	Port	Port
Biff	Echo	RIP	Talk
512	7	520	517
Bootpc	Mobile-IP	SNMP	TFTP
	434		
68		161	69
	Name Server		
Bootps	42	SNMPtrap	Time
67	NetBIOS-DGM	162	37
	138		
Discard		SunRPC	Who
	NetBIOS-NS		
9	137	111	513

DNS	NTP	Syslog	XDMCP
	123		177
53		514	
			Port range (0–65535)
DNSIX		TACACS	
195		49	

Table 8-8. Layer 4 TCP Criteria

Port Keyword	Port Keyword	Port Keyword	Port Keyword
Port	Port	Port	Port
BGP	FTP	LDP	Telnet
179	21	515	23
Chargen	FTP-Data	NNTP	Time
19	20	119	37
Daytime	Gopher	POP2	UUCP
13	70	109	540
Discard	Hostname	POP3	Whois
9	101	110	43
Domain	IRC	SMTP	WWW
53	194	25	80
Echo	Klogin	SunRPC	Port range (0–65535)
7	543	111	
Finger	Kshell	TACACS	
79	544	49	

Although not as granular as IP, IPX and MAC ACLs can classify traffic based on specific protocol numbers and ethertype values, as well as based on the source and destination network and host numbers. Administrators can identify particular IPX protocols based on keywords—(ncp (17), rip (1),sap (4), spx (5))—or a protocol number in the range from 0 to 19 or 21 to 255. MAC ACLs enable the administrator to identify traffic based on ethertype value, utilizing one of the keywords or values shown in Table 8-9. Once again, if a specific protocol or ethertype value is not included in the ACL configuration, the resulting behavior is to classify all IPX or MAC traffic based on the indicated network or node addresses.

Table 8-9. MAC Classification Criteria

Keyword	Keyword
Ethertype	Ethertype
AARP	DEC-Mumps
0x80F3	0x6009
Banyan-Vines-Echo	DEC-NetBIOS
0x0baf	
	0x8040
DEC-Amber	
0x6008	DEC-Phase-IV
	0x6003
DEC-DSM	
0x8039	Ethertalk
	0x809B
DEC-Diagnostic-Protocol	
0x6005	XEROX-NS-IDP
DEC-LANBridge	0x0600
0x8038	
	Valid Configurable Ethertype Values
DEC-LAT	0x809B,0x80F3
0x6004	
DEC-LAVC-SA	0x6000–0x6009
0x6007	
DEC-MOP-Dump	0x8038–0x8039
0x6001	
DEC-MOP-Remote-Console	0x8040–0x8042
0x6002	
DEC-MSDOS	0x0BAD,0x0BAF,
0x8041	0x0600

For all ACL types, after the named ACLs have been configured, they are temporarily placed into an edit buffer within memory. As a result, if changes are made to an ACL, or a new ACL is created, it is necessary to commit the changes before they can be implemented. Committing the ACL copies the commands from the temporary edit buffer to the PFC in hardware. Commits are performed with the following command, `commit qos acl {ACL_name | all}`. Prior to committing any changes, it is possible to "roll back" any modifications without impacting the performance of currently operational ACLs. This eliminates any ACL alterations or additions present within the temporary edit buffer. This is accomplished with `rollback qos acl {ACL_name | all}`. When modifying default ACL parameters, it is not necessary to commit any changes. All changes take affect immediately. To verify whether an ACL has been committed to hardware, or if there are outstanding "not committed" changes, use `show qos acl editbuffer`. Refer to Example 8-23 in the "Classification and Marking in Hybrid Mode" section for a demonstration on configuring and applying QoS ACLs using Hybrid mode.

Class Maps and Policy Maps with Cisco IOS

With Cisco IOS, classification is performed utilizing Cisco's MQC, as described in Chapter 5. The modular CLI enables the administrator to classify all interesting traffic by defining named or numbered IOS ACLs and referencing them within class maps. Class maps are then applied to policy maps. Policy maps identify the actions performed on traffic corresponding to the predefined class maps, which are referenced within the policy map. Finally, the defined policies are then applied to the appropriate port or interface with the `service-policy {input | output}` command. For further information regarding the Cisco MQC, refer to Chapter 5.

Cisco IOS supports the classification of IP, IPX, and MAC type traffic. With Cisco IOS, both IP and IPX flows can be classified utilizing standard-numbered ACLs, extended-numbered ACLS, or named ACLs. For MAC layer traffic, network data is classified using named ACLs only.

> NOTE
>
> With Release 12.1(1)E and later, it is possible to classify IPX and MAC layer traffic. However, QoS support for IPX classification can be based on source network, and optionally destination network and node parameters. Classifying IPX type traffic based on socket numbers, source node, protocol, or service type is not supported.

When defining interesting traffic, you can configure multiple class maps. Potentially, one class map can be specified for each type of inbound traffic for a designated receiving interface. QoS on the Catalyst 6500 only supports one `match` statement when defining classification criteria. You can match traffic based on Layer 3 precedence values by using `match ip`, or administratively defined ACLs by using `match access-group`. All other class map options are not supported for QoS. The exception to this limitation is the `match protocol` option. With 12.1(13)E and later releases, `match protocol` can be used to support *network-based application recognition* (NBAR) on the Catalyst 6500 with an MSFC 2. At the time of writing, however, NBAR is implemented only in software. Refer to Chapter 9 for further details regarding NBAR support on the Catalyst 6500. After the traffic flows have been delineated, you can reference each class map within a policy map. You can assign only one policy map to each interface. For an example of configuring QoS policies using the MQC, refer to Example 8-24. Policy maps, like class maps, have certain configuration limitations. The following class {*class-name*} keywords are not supported under

the QoS policy map configuration:

- class {*class-name*}destination-address
- class {*class-name*}input-interface
- class {*class-name*}qos-group
- class {*class-name*}source-address

Finally, remember when configuring QoS policies in Cisco IOS that the console does not immediately notify the administrator when an unsupported command is used. When the command is applied to the target interface using the service-policy input command, the administrator is notified of any discrepancies.

ACL-Based Classification and Marking in Hybrid Mode

Classification and marking are extremely important in determining how a frame is processed within the 6500, as well as for determining what preference it is given when forwarded on to the network. These actions determine the internal DSCP value chosen for a frame, which in turn translates to an egress DSCP and CoS value. After the classification parameters have been established, you must determine what action to apply to all traffic meeting those specifications. This section focuses on classification and marking in Hybrid mode, tying together all previously discussed topics. When a frame is received, it is forwarded out on the D-bus, where it is seen by all ports and the PFC. At the PFC, hardware-logic processes configured ACLs, which interact with port trust states to determine the proper classification or marking settings to derive the internal DSCP value.

Configuring classification and marking with QoS ACLs in Hybrid mode involves four supported rules. These rules are implemented using the following four keywords: trust-cos, trust-ipprec, trust-dscp, and dscp.Table 8-10 summarizes the corresponding behavior resulting from configuring one of the four ACE keywords and describes how each interacts with existing ingress port trust states.

Table 8-10. Summary of Marking Rules in Hybrid Mode

Port Trust State ACE Keyword	untrusted	trust-cos	trust-ipprec	trust-dscp
trust-cos	Port CoS value: default value (0)	Port CoS or CoS value of arriving frame	Port CoS or CoS value of arriving frame[1]	Port CoS or CoS value of arriving frame[1]
trust-ipprec	IP precedence value of arriving frame	IP precedence value of arriving frame	IP precedence value of arriving frame[1]	IP precedence value of arriving frame[1]
trust-dscp	DSCP value of arriving frame	DSCP value of arriving frame	DSCP value of arriving frame[1]	DSCP value of arriving frame[1]
dscp	DSCP value specified in ACE	Port CoS or CoS value of arriving frame[2]	IP precedence value of arriving frame[1]	DSCP value of arriving frame[1]

[1] Configuration settings do not apply to WS-X6224/6248 and WS-X6324/6348 series linecards.

[2] The internal DSCP value is derived from the DSCP value specified in the ACL for WS-X6224/6248 and WS-X6324/6348 series linecards.

As you can see in Table 8-10, when an ACE keyword is specified within a QoS ACL, the keyword overrides the port trust policy applied to the ingress interface or VLAN. The exception is when using the dscp keyword. Instead of always marking the frame header using the DSCP priority configured in the QoS ACE, the keyword operates in conjunction with the port trust state. If the ingress port is configured to trust the arriving CoS, IP precedence, or DSCP value, dscp instructs the switch to maintain the QoS setting derived from the ingress port's classification policy. If the ingress port trust is left at its default setting (untrusted), however, dscp marks the frame header with the value specified in the ACE. Other notable behavior includes configuring the trust-cos keyword within an ACE. trust-cos is not a recommended setting when the inbound port's trust policy is set to untrusted. This combination may result in unexpected behavior. Traffic arriving on an interface configured as untrusted is immediately labeled with the default port CoS setting, thus overwriting the existing CoS priority. The frame header is then forwarded directly to the switching engine. As a result, the value maintained by the trust-cos command, within the ACE, may not be the expected value. Therefore, when using the trust-cos keyword, it is recommended the port be configured for trust-cos as well.

NOTE

Because the WS-X6148 is not subject to the same hardware limitations as the WS-X6224/6248 and the WS-X6324/6348, all the marking rules provided in Table 8-10 apply.

Not all marking rules listed in the Table 8-10 apply to WS-X6224/6248 or WS-X6324/6348 linecards. As mentioned earlier, trust-ipprec and trust-dscp are not supported configurations at the port level for these modules. Also due to hardware limitations, the port ASIC for these linecards cannot preserve the inbound priority without the administrator entering additional

commands. Therefore, if the desired action is to maintain the inbound CoS setting, a QoS ACL must be configured with the trust-cos keyword. Furthermore, as a result of the same port ASIC limitation, if the dscp keyword is specified rather than trust-cos, the arriving frame is marked based on the specified codepoint rather than the trusted inbound CoS priority.

Example 8-21 demonstrates how to configure a QoS ACL in Hybrid mode. In this example, a named ACL called VideoConf is created. For this ACL, the intent is to mark all traffic destined for TCP ports 1720, 1731, and 1503 with DSCP 26 and to maintain the classification established by the default port value for all other traffic. The TCP ports identified are control ports used in some video conferencing applications. Therefore, DSCP 26, equivalent to precedence 3, is applied to the matching traffic. In this instance, port 4/2 is assumed to be a trunk port.

Example 8-21. Configuring and Applying an IP QoS ACL in Hybrid Mode

```
hybrid (enable) set qos acl ip VideoConf dscp 26 tcp any any eq 1720

VideoConf editbuffer modified. Use 'commit' command to apply changes.

hybrid (enable) set qos acl ip VideoConf dscp 26 tcp any any eq 1731

VideoConf editbuffer modified. Use 'commit' command to apply changes.

hybrid (enable) set qos acl ip VideoConf dscp 26 tcp any any eq 1503

VideoConf editbuffer modified. Use 'commit' command to apply changes.

hybrid (enable) set qos acl ip VideoConf trust-cos ip any any

Warning: ACL trust-cos should only be used with ports that are also configured

with port trust=trust-cos.

VideoConf editbuffer modified. Use 'commit' command to apply changes.
```

After the ACL has been created, a message is sent to the console. It reminds the administrator to commit the changes to the PFC, removing them from the temporary edit buffer. Looking at the edit buffer, you can confirm the ACL has not yet been committed.

```
hybrid (enable) show qos acl editbuffer

ACL                              Type Status

-------------------------------  ---- ----------

VideoConf                        IP   Not Committed

hybrid (enable) commit qos acl VideoConf
```

```
QoS ACL 'VideoConf' successfully committed.

hybrid (enable) show qos acl editbuffer

ACL                              Type Status

-------------------------------  ---- ----------

VideoConf                        IP   Committed
```

After the ACL has been committed to hardware, you can verify the ACL configuration with the following command:

```
hybrid (enable) show qos acl info config VideoConf

set qos acl ip VideoConf

---------------------------------------------

1. dscp 26 tcp any  any  eq 1720

2. dscp 26 tcp any  any  eq 1731

3. dscp 26 tcp any  any  eq 1503

4. trust-cos ip any  any
```

After verifying the ACL configuration, you can apply the ACL to the desired port or VLAN interface. This is accomplished by issuing the command set qos acl map {*ACL_name*} {{*mod*/*port*} | {*VLAN*}}. In the example, the ACL is mapped to a port. Recall that to successfully map the ACL to the desired port, the port must be set for port-based QoS. After the ACL has been successfully mapped to the desired port or interface, you can confirm the configuration with show qos acl map {runtime | config} {*ACL_name* | all |*mod*/*port*|*VLAN*}.

```
hybrid (enable) set qos acl map VideoConf 4/2

ACL VideoConf is successfully mapped to port 4/2.

hybrid (enable) show qos acl map config VideoConf

QoS ACL mappings on rx side:

ACL name                         Type Vlans

-------------------------------- ---- --------------------------------
```

```
VideoConf                          IP

ACL name                         Type Ports

-------------------------------- ---- --------------------------------

VideoConf                          IP 4/2
```

Finally, because the ACL has been created, committed to hardware, and applied to the desired port, it is possible to verify that the configured behavior is the desired behavior. The verification is based on the port's QoS configuration information. `show qos info` yields the following results:

```
hybrid (enable) show qos info config 4/2

QoS setting in NVRAM:

QoS is enabled

Policy Source of port 4/2: COPS

Tx port type of port 4/2 : 1p2q2t

Rx port type of port 4/2 : 1p1q4t

Interface type: port-based

ACL attached: VideoConf

The qos trust type is set to untrusted.

Default CoS = 2

(text omitted)
```

As displayed in the output, the port is configured for port-based QoS, and the QoS trust type is set to untrusted. Also the show command verifies that an ACL named VideoConf is attached to the port and that the default port CoS is 2. You may recognize behavior described in Table 8-10. Although all traffic arriving on port 4/2 is assigned the default CoS and forwarded directly to the PFC, because of the attached ACL and `dscp` keyword, all traffic destined to TCP ports 1720, 173, and 1503 are marked with DSCP 26. This behavior is attributed to the ingress port trust being set to untrusted. Furthermore, because the desire is to maintain the default CoS setting, it is necessary to configure the additional ACE with the `trust-cos` keyword. If the ACE is not included, all traffic not matching the first ACE is matched by the default IP ACL and assigned a DSCP value of zero.

ACL-Based Classification and Marking in Native Mode

The concepts for classification and marking with Native mode are similar to those discussed for Hybrid mode. In conjunction with the ACLs used to classify interesting traffic, however, Native mode incorporates the use of the MQC introduced in Chapter 5. As compared to operation in Hybrid mode, Native mode has its own set of rules for marking traffic. Table 8-11 summarizes these rules.

Table 8-11. Summary of Marking Rules in Native Mode

Port Trust State Policy Map Keyword	untrusted	trust cos	trust precedence	trust dscp
trust cos	Port CoS value: default value (0)	Port CoS or CoS value of arriving frame[1]	Port CoS or CoS value of arriving frame[1]	Port CoS or CoS value of arriving frame[1]
trust precedence	IP precedence value of arriving frame	IP precedence value of arriving frame[1]	IP precedence value of arriving frame[1]	IP precedence value of arriving frame[1]
trust dscp	DSCP value of arriving frame	DSCP value of arriving frame[1]	DSCP value of arriving frame[1]	DSCP value of arriving frame[1]
set ip precedence	Precedence value derived from value in policy map	Port CoS or CoS value of arriving frame[1]	IP precedence value of arriving frame[1]	DSCP value of arriving frame[1]
set ip dscp	DSCP value derived from value in policy map	Port CoS or CoS value of arriving frame[1]	IP precedence value of arriving frame[1]	DSCP value of arriving frame[1]

[1] Configuration settings do not apply to WS-X6224/6248 and WS-X6324/6348 series linecards.

Similar to the behavior in Hybrid mode, when the trust keyword is utilized in a policy map class, and subsequently applied to an interface with the service-policy input statement, the trust state of the policy map class supersedes the trust state specified at the interface. Also as specified in Table 8-11, it is not possible to configure the interface trust state for WS-X6224/6248 and WS-X6324/6348 linecards. On these linecards, all ports default to the nonconfigurable untrusted state. This behavior differs slightly from options available in Hybrid mode. Also trust cos is not available when using these series of modules in Native mode.

Prior to 12.1(12c)E1, marking was only possible through policing. Therefore, marking required the administrator to configure a policer, which did not police but just marked priority traffic and transmitted it. With the release of 12.1(12c)E1, set ip precedence and set ip dscp were made available as actions within a class, under the policy map configuration. Therefore, for traffic arriving on an untrusted interface, it became possible to set the IP precedence or DSCP value for incoming frames. Again, similar to the behavior demonstrated in Hybrid mode with the ACE dscp keyword,set ip precedence and set ip dscp, when specified, do not supersede the configured port trust state. The PFC maintains the ingress CoS, IP precedence, or DSCP value, as long as the correspondingmls qos trust keyword is configured at the interface.

The following example demonstrates how to classify and subsequently mark traffic using Native

mode. The same criteria used in the Hybrid examples is used here. Again, the intent is to mark all traffic destined for TCP ports 1720, 1731, and 1503 with a DSCP value of 26, whereas all other traffic is marked with the default CoS setting for the interface. The first step in the QoS policy creation is creating an ACL for all relevant traffic. When configured, the ACL is then mapped to an administratively defined class map. You can verify these configuration steps with the`show access-list` and `show class-map` commands.

Example 8-22. Configuring and Applying a QoS Policy in Native Mode

```
native(config)#access-list 110 remark Control traffic for Video Conferencing App

native(config)#access-list 110 permit tcp any any eq 1720

native(config)#access-list 110 permit tcp any any eq 1731

native(config)#access-list 110 permit tcp any any eq 1503

native(config)#class-map match-any Control-traffic

native(config-cmap)#match access-group 110

native#show access-list 110

Extended IP access list 110

    permit tcp any any eq 1720

    permit tcp any any eq 1731

    permit tcp any any eq 1503

native#show class-map Control-traffic

 Class Map match-any Control-traffic (id 12)

   Match access-group  110
```

When the desired traffic has been identified, and referenced in a class map, that class map is then mapped to a defined policy map. In this instance, the policy map name corresponds to a broader topic, which encompasses the traffic identified in the class map. This was done to demonstrate the modular aspect of the relationship between class maps and policy maps. The policy map name chosen is VideoConf. Because there are potentially different identifiable traffic streams, you can reference multiple class maps within one policy map. Each class map, in turn, may have its own marking parameter. For this example, the actual audio and video streams can be identified and placed within their own class map and marked appropriately. Therefore, the recommendation is to name the policy map something related to the class map, but in a broader sense. Depending on how the policies are formulated, for instance, the policy map may be named after a department or after an application (such as VideoConf). In this scenario, the goal is to mark all traffic destined for TCP ports 1720, 1731, and 1503 with DSCP 26. Therefore, the `set ip dscp {`*`dscp value`*`}` command is utilized within the policy map class, along with the desired value.

```
native(config)#policy-map VideoConf
native(config-pmap)#class Control-traffic
native(config-pmap-c)#set ip dscp 26
native#show policy-map VideoConf
 Policy Map VideoConf
  class  Control-traffic
   set ip dscp 26
```

After the policy map parameters have been established, you must apply the QoS policy to the appropriate interface to complete the process. Here, the desired interface is Gigabit 5/1, and the service-policy input {*name*} command is configured referencing policy map VideoConf.

```
native(config)#interface gigabitEthernet 5/1
native(config-if)#service-policy input VideoConf
native#show policy-map interface gigabitEthernet 5/1
 GigabitEthernet5/1
  service-policy input: VideoConf
    class-map: Control-traffic (match-any)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      match: access-group 110
 (text omitted)
native#show mls qos ip ingress
 QoS Summary [IP]:      (* - shared aggregates, Mod - switch module)
Int            Mod Dir Cl-map DSCP AgId Trust FlId   AgForward-Pk   AgPoliced-Pk
--------------------------------------------------------------------------------
Gi5/1            1 I Control-t 26    3     No    0              0              0
(text omitted)
```

```
native#show queueing interface gigabitEthernet 5/1

Interface GigabitEthernet5/1 queueing strategy:  Weighted Round-Robin

  Port QoS is enabled

  Port is untrusted

  Default COS is 2

(text omitted)
```

Finally, after the appropriate steps for MQC have been completed, you can verify behavior. As demonstrated,show policy-map interface {*type num*} and show mls qos ip ingress detail the policy map class information. Specifically, the outputs include the ACL referenced in the class map and the DSCP value used to mark all conforming traffic. Based on the information in Table 8-11, and the output extracted from show queueing interface {*type num*}, note that the port is configured as untrusted. This implies all traffic matching the ACL is marked with DSCP 26. Recall that set ip dscp overrides interfaces with an untrusted port trust state. All remaining traffic is marked with the default CoS value of 2. As mentioned previously, this configuration is possible with Cisco IOS Release 12.1(12c)E1 and later. For earlier versions, it is necessary to configure a policer to mark all desired traffic. Policing is covered later in the chapter.

Mapping

After a frame header has been forwarded from the ingress port to the switching engine, and the appropriate priority value has been established, the PFC determines an internal DSCP value for the This value is used to signify how a frame is handled while it traverses the switch, and ultimately ho scheduled upon egress. A DSCP value is applied to all frames processed by the switching engine, regardless of traffic type. This behavior is also independent of the operating system. However, the for configuring the tables is different between Native mode and Hybrid mode. Table 8-12 shows the mapping values used by the 6500.

Table 8-12. Default Mapping Values

CoS-to-DSCPMapping	Precedence-to-DSCPMapping	DSCP-to-CoS Mappin
CoS 0 = DSCP 0	IP precedence 0 = DSCP 0	DSCP 0–7 = CoS 0
CoS 1 = DSCP 8	IP precedence 1 = DSCP 8	DSCP 8–15 = CoS 1
CoS 2 = DSCP 16	IP precedence 2 = DSCP 16	DSCP 16–23 = CoS 2
CoS 3 = DSCP 24	IP precedence 3 = DSCP 24	DSCP 24–31 = CoS 3
CoS 4 = DSCP 32	IP precedence 4 = DSCP 32	DSCP 32–39 = CoS 4
CoS 5 = DSCP 40	IP precedence 5 = DSCP 40	DSCP 40–47 = CoS 5
CoS 6 = DSCP 48	IP precedence 6 = DSCP 48	DSCP 48–55 = CoS 6
CoS 7 = DSCP 56	IP precedence 7 = DSCP 56	DSCP 56–63 = CoS 7

The Catalyst 6500 determines the internal DSCP value based on the trust state specified either at t or interface level, or based on the trust keyword or DSCP/IP precedence value specified in an ACL. following sections detail how the internal DSCP values are derived from the appropriate sources an internal DSCP values map to egress CoS. These topics are covered in the following order:

- CoS-to-DSCP Mapping
- Precedence-to-DSCP Mapping
- DSCP-to-CoS Mapping
- Policed DSCP-to-Mark Down Mapping

CoS-to-DSCP Mapping

If the ingress port or interface has been configured to trust-cos, the internal DSCP value is derive the arriving CoS on the 802.1q or ISL trunk. Likewise, because an ACL or policy map class superse port trust state, if they are configured to trust the incoming CoS value, the internal DSCP is again d from the inbound CoS. One caveat to consider is traffic arriving on an untrusted port. Although inc frames are marked with the default CoS setting configured for the port or interface, in this case the

DSCP value is derived from the default ACL. By default, the default ACL applies a DSCP value of ze
result, if the desire is to maintain the CoS established by the default CoS value at the interface, it i
necessary to use an ACL or policy map class to trust CoS, or modify the default DSCP value in the d
ACL.

```
(Hybrid)

set qos cos-dscp-map {DSCP1 DSCP2 DSCP3 DSCP4 DSCP5 DSCP6 DSCP7 DSCP8}

show qos map config cos-dscp-map

(Native)

mls qos map cos-dscp {DSCP1 DSCP2 DSCP3 DSCP4 DSCP5 DSCP6 DSCP7 DSCP8}

show mls qos map [ | begin Cos-dscp map]
```

Table 8-12 shows the default mapping settings for the CoS-to-DSCP table. To modify and verify the
or modified mapping settings for the CoS-to-DSCP table, use the preceding commands. These com
enable an administrator to map one DSCP value to a CoS value. As a result, referencing the preced
commands, {*DSCP1*} maps to CoS 0 and {*DSCP2*} maps to CoS 1. This pattern applies to all DSCP
concluding with {*DSCP8*}, which maps to CoS 7. Examples 8-23 and 8-24 demonstrate how to mo
default mapping values in both Hybrid and Native modes, respectively.

Example 8-23. Modifying the CoS-to-DSCP Mapping Table in Hybrid Mode

```
hybrid (enable) set qos cos-dscp-map 56 48 40 32 24 16 8 0

QoS cos-dscp-map set successfully.

hybrid (enable) show qos map config cos-dscp-map

CoS - DSCP map:

CoS   DSCP

---   ----

  0   56

  1   48
```

```
  2   40
  3   32
  4   24
  5   16
  6   8
  7   0
```

Example 8-24. Modifying the CoS-to-DSCP Mapping Table in Native Mode

```
native(config)#mls qos map cos-dscp 56 48 40 32 24 16 8 0
native#show mls qos map | begin Cos-dscp map
   Cos-dscp map:
        cos:   0  1  2  3  4  5  6  7
     ------------------------------------
       dscp:  56 48 40 32 24 16  8  0
(text omitted)
```

The values mapped in the preceding examples are not recommended settings. Instead, they demo the capability of the CoS-to-DSCP mapping feature. It is normally not necessary to modify the CoS DSCP mapping table from its defaults. If you do make alterations, however, carefully consider how values are mapped and propagate the modifications across the entire network for consistency. Cha performed to this table impact the performance of the DSCP-to-CoS table, which needs to be consi When configuring the mapping feature on WS-X6224/WS-X6248 and WS-X6324/WS-X6348 linecar additional steps are required. In Hybrid, based on limitations previously discussed, a port configur trust CoS sustains the inbound CoS setting by applying an ACE with the `trust-cos` keyword. The sv then derives the DSCP value from the CoS of the incoming tagged frame. With Native mode, trust i configurable option for these modules at the port or interface level. All ports default to untrusted. A result, to maintain the local port CoS setting and use it to derive the internal DSCP, a policy map c must be configured and applied to the applicable interface. Otherwise, the value associated with th ACL is utilized to obtain the internal DSCP value, which is statically defined as zero.

Precedence-to-DSCP Mapping

Similar to the CoS-to-DSCP mapping, internal DSCP values can also be derived from arriving IP precedence values. The default IP precedence-to-DSCP mapping is shown in Table 8-12. Frames ar

ports or interfaces configured with the trust-ipprec or trust ip-precedence keywords obtain thei internal DSCP value from the precedence of the ingress traffic. This same feature applies to ACLs a policy map classes. The feature is also available for WS-X6224/WS-6248 and WS-X6324/WS-X634 modules. Because this configuration is not supported at the physical interface or port level for thes linecards, using either Hybrid or Native modes, however, an ACL or a policy map class must be cor The following commands enable the administrator to modify the default mappings and verify the configuration:

```
(Hybrid)

set qos ipprec-dscp-map {DSCP1} {DSCP2} {DSCP3} {DSCP4} {DSCP5} {DSCP6} {DSCP7} {D

show qos maps config ipprec-dscp-map

(Native)

mls qos map ip-prec-dscp {DSCP1} {DSCP2} {DSCP3} {DSCP4} {DSCP5} {DSCP6} {DSCP7} {

show mls qos maps [ | begin IpPrec ]
```

As seen with CoS-to-DSCP mapping, {*DSCP1*} is associated with IP precedence 0 and {*DSCP2*} corresponds with IP precedence 1. The pattern then concludes with {*DSCP8*}, which maps to IP pre 7.Examples 8-25 and 8-26 demonstrate how to alter the default IP precedence-to-DSCP mapping t and how to verify the configuration.

Example 8-25. Modifying the IP Precedence-to-DSCP Mapping Table in H Mode

```
hybrid (enable) set qos ipprec-dscp-map 56 48 40 32 24 16 8 0

QoS ipprec-dscp-map set successfully.

hybrid (enable) show qos maps config ipprec-dscp-map

IP-Precedence - DSCP map:

IP-Prec   DSCP

-------   ----

      0   56
```

```
      1   48

      2   40

      3   32

      4   24

      5   16

      6   8

      7   0

hybrid (enable)
```

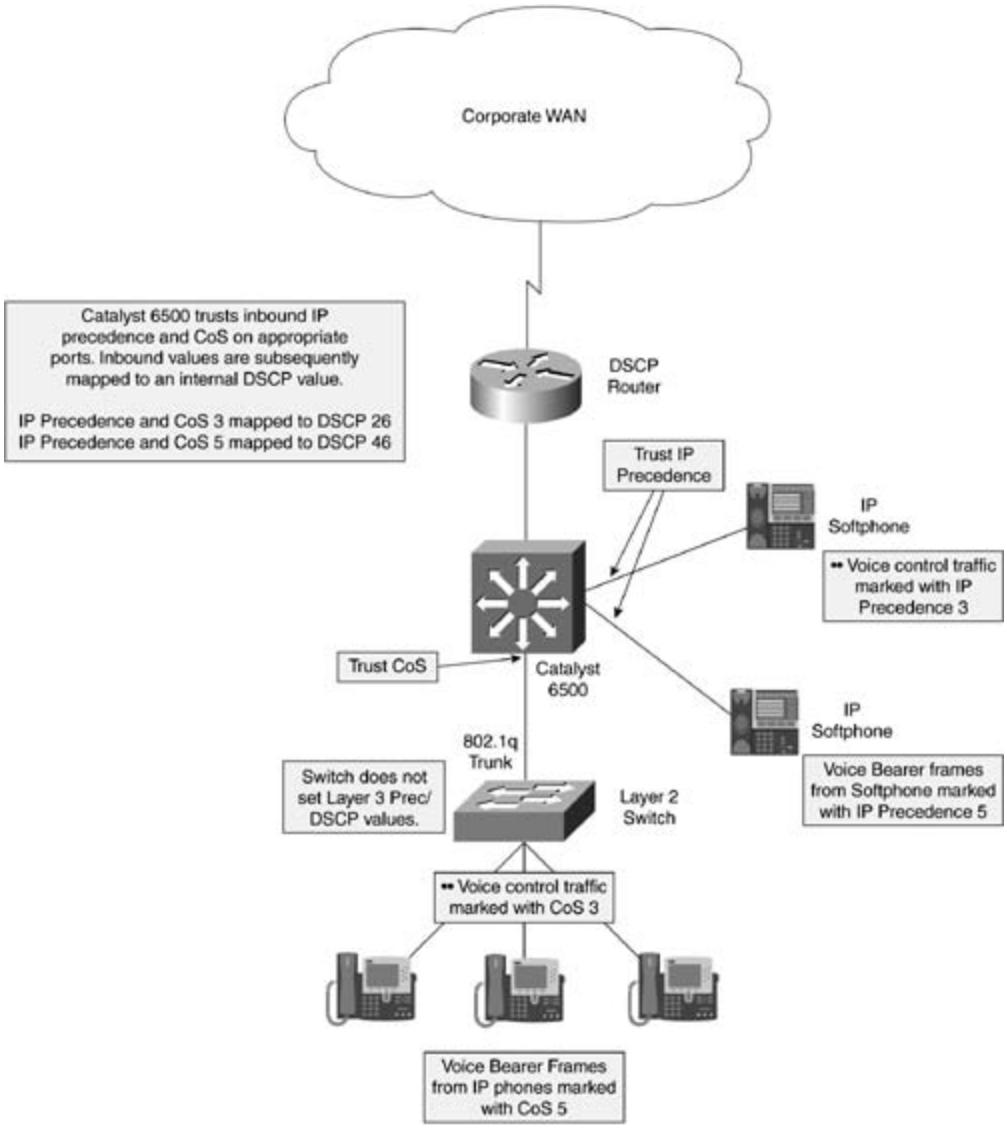
Example 8-26. Modifying the IP Precedence-to-DSCP Mapping Table in Na Mode

```
native(config)#mls qos map ip-prec-dscp 56 48 40 32 24 16 8 0

native#show mls qos maps | begin IpPrec

   IpPrecedence-dscp map:

     ipprec:   0  1  2  3  4  5  6  7

     -----------------------------------

       dscp:  56 48 40 32 24 16  8  0
```

Once again, the values mapped in the preceding examples are not recommended settings. They demonstrate the capability of the IP precedence-to-DSCP mapping feature. The default settings are sufficient for most instances, unless specific requirements dictate otherwise. If the default settings modified, however, it is imperative to maintain consistency across the entire network. One example it may be necessary to modify the default map settings is when deploying voice in the campus. For streams, DSCP 26 is recommended for control traffic, whereas DSCP 46 is recommended for the ac voice packets. This is critical in environments where QoS policies are implemented using DSCP valu demonstrate this scenario, consider the network in Figure 8-7.

Figure 8-7. When to Change the Default Map Settings

[View full size image]



The Catalyst 6500 in Figure 8-7 is trunked to a Layer 2 switch, which is trusting the incoming CoS s the IP Phones. Also two laptops configured with IP Softphone are attached directly to the Catalyst (Softphones are capable of marking voice packets with an IP precedence of 5. The router attached t corporate WAN implements QoS using DSCP-based policies. In this case, to ensure the proper end- handling for the voice traffic, it is necessary to alter the default values in the CoS-to-DSCP and IP precedence-to-DSCP mapping tables. For both tables, CoS 3 and IP precedence 3 are mapped to D and CoS 5 and IP precedence 5 are mapped to DSCP 46. Modifying the mapping tables to account voice traffic ensures the voice streams receive the appropriate service levels when forwarded to DS capable devices.

Because the sources for the internal DSCP values have been determined, the discussion can now tu how the internal DSCP values are mapped back into a frame upon egress.

DSCP-to-CoS Mapping

When the frame header is passed from the switching engine to the egress port or interface, the int DSCP value is mapped back to a CoS value. The internal DSCP is also written to the ToS field of the

IP header. The CoS value allows the frame to utilize the egress queue scheduling as it exits the swi
Output scheduling is covered in a later section. In addition to scheduling, if the outbound port is ar
802.1q trunk, the CoS value is written into the trunk header and transmitted to the network. The C
is reflected in the user priority field. To modify and verify the configuration of the DSCP-to-CoS ma
table, use the following commands:

(Hybrid)

set qos dscp-cos-map {*DSCP1* [*DSCP2...DSCP64*]}:{*CoS*}

show qos maps config dscp-cos-map

(Native)

mls qos map dscp-cos {*DSCP1* [*DSCP2 DSCP3 DSCP4 DSCP5 DSCP6 DSCP7 DSCP8*]}**to** {*CoS*}

show mls qos map [**| begin Dscp-cos**]

In Hybrid, a range of DSCP values can be mapped to a single CoS. If the intent is to map DSCP 24
35 to CoS 3, for example, the command can be configured as follows: set qos dscp-cos-map 24-
Native mode does not permit the administrator to enter a range of values. A maximum of eight
independent DSCP values can be specified per command. In addition, when entering individual DS
values in Hybrid, you can enter multiple values; however, you must separate the values with comn
Native mode requires the DSCP values to be separated with spaces. The following two examples
demonstrate how to configure the DSCP-to-CoS mapping table. They also demonstrate the differen
command syntax. Note that to demonstrate dependency, the following examples were configured t
the changes made to the CoS-to-DSCP mapping table in previous examples.

Example 8-27. Modifying the DSCP-to-CoS Mapping Table in Hybrid Mode

hybrid (enable) **set qos dscp-cos-map 56-63:0**

QoS dscp-cos-map set successfully.

hybrid (enable) **set qos dscp-cos-map 48-55:1**

QoS dscp-cos-map set successfully.

hybrid (enable) **set qos dscp-cos-map 40-47:2**

QoS dscp-cos-map set successfully.

```
hybrid (enable) set qos dscp-cos-map 32-39:3
QoS dscp-cos-map set successfully.
hybrid (enable) set qos dscp-cos-map 24-31:4
QoS dscp-cos-map set successfully.
hybrid (enable) set qos dscp-cos-map 16-23:5
QoS dscp-cos-map set successfully.
hybrid (enable) set qos dscp-cos-map 8-15:6
QoS dscp-cos-map set successfully.
hybrid (enable) set qos dscp-cos-map 0,2,4,6:7
QoS dscp-cos-map set successfully.
hybrid (enable) set qos dscp-cos-map 1,3,5,7:7
QoS dscp-cos-map set successfully.

hybrid (enable) show qos maps config dscp-cos-map
DSCP - CoS map:
DSCP                              CoS
--------------------------------  ---
                           56-63  0
                           48-55  1
                           40-47  2
                           32-39  3
                           24-31  4
                           16-23  5
                            8-15  6
                             0-7  7
```

Example 8-28. Modifying the DSCP-to-CoS Mapping Table in Native Mode

```
native(config)#mls qos map dscp-cos 56 57 58 59 60 61 62 63 to 0
native(config)#mls qos map dscp-cos 48 49 50 51 52 53 54 55 to 1
native(config)#mls qos map dscp-cos 40 41 42 43 44 45 46 47 to 2
native(config)#mls qos map dscp-cos 32 33 34 35 36 37 38 39 to 3
native(config)#mls qos map dscp-cos 24 25 26 27 28 29 30 31 to 4
native(config)#mls qos map dscp-cos 16 17 18 19 20 21 22 23 to 5
native(config)#mls qos map dscp-cos 8 9 10 11 12 13 14 15 to 6
native(config)#mls qos map dscp-cos 0 1 2 3 4 5 6 7 to 7

native#show mls qos map | begin Dscp-cos
   Dscp-cos map:                                (dscp= d1d2)
     d1 :  d2 0  1  2  3  4  5  6  7  8  9
     ------------------------------------
      0 :    07 07 07 07 07 07 07 07 06 06
      1 :    06 06 06 06 06 06 05 05 05 05
      2 :    05 05 05 05 04 04 04 04 04 04
      3 :    04 04 03 03 03 03 03 03 03 03
      4 :    02 02 02 02 02 02 02 02 01 01
      5 :    01 01 01 01 01 01 00 00 00 00
      6 :    00 00 00 00
(text omitted)
```

Policed DSCP Mark-Down Mapping

The Catalyst 6500 has an additional feature that allows it to mark down DSCP values, based on int
policed DSCP mark-down tables. Contingent on configured policing parameters, instead of droppin
profile frames, the mark-down table enables the administrator to define DSCP translations. This m
previous internal DSCP value to another defined "marked-down" DSCP value for frames violating t
configured policing contract. The PFC hardware version, either a PFC1 or PFC2, determines how ma
mark-down tables are available.

The Catalyst 6500 with a PFC1 only provides support for the single-rate policing function. As a resu
option of configuring the policed DSCP mark-down table for frames violating the normal traffic rate
available with the PFC1. On the other hand, the PFC2 provides an additional enhancement beyond
available with the PFC1. The PFC2 can support a dual-rate policer. With the PFC2, two mark-down
can be altered, a table for traffic violating the normal rate and streams violating the excess rate. Ta
13 lists the default mapping values for both normal and excess rate policed DSCP mark-down table

Table 8-13. Normal And Excess Rate Default Policed DSCP Mark-Down V

Internal DSCP	Mark-Down DSCP	Internal DSCP	Mark-Down DSCP	Internal DSCP	Mark-Down DSCP	Internal DSCP	Mar Dow DSC
0	0	16	16	32	32	48	48
1	1	17	17	33	33	49	49
2	2	18	18	34	34	50	50
3	3	19	19	35	35	51	51
4	4	20	20	36	36	52	52
5	5	21	21	37	37	53	53
6	6	22	22	38	38	54	54
7	7	23	23	39	39	55	55
8	8	24	24	40	40	56	56
9	9	25	25	41	41	57	57
10	10	26	26	42	42	58	58
11	11	27	27	43	43	59	59
12	12	28	28	44	44	60	60
13	13	29	29	45	45	61	61
14	14	30	30	46	46	62	62
15	15	31	31	47	47	63	63

To alter how internal DSCP values are mapped to policed DSCP mark-down values, use the followir
commands:

(Hybrid)

```
set qos policed-dscp-map [excess-rate | normal-rate] {DSCP1 [DSCP2...DSCP64]}:{DSC

show qos maps config policed-dscp-map [excess-rate | normal-rate]

(Native)

mls qos map policed-dscp {max-burst | normal-burst} {DSCP1 [DSCP2 DSCP3 DSCP4 DSCP
➥DSCP7 DSCP8]}to {DSCP}

show mls qos map [ | begin Policed-dscp]
```

With the PFC2, the additional excess-rate and normal-rate keywords are available in Hybrid mo themax-burst and normal-burst keywords are available in Native mode. If neither of the keywor specified, the configured DSCP mappings apply to the normal rate table. With the PFC1, the precec keywords are not available.

When configuring the excess rate table on the PFC2, if changes have been made to the normal rate those alterations are independent from policies created for the excess rate table. This is attributed mark-down operation occurring after the frame has been processed through the policing function. the scenario where a dual-rate policer is configured, and traffic streams violating the configured co rates have their DSCP values marked down in both instances. If a frame violates only the normal t rate profile, it is marked down according to how the original internal DSCP value is mapped to othe values in the normal rate table. When the same frame reaches the excess rate policer, it still posse original internal DSCP value. Therefore, if the frame exceeds the excess rate, the mark-down value determined based on how the original DSCP value is mapped in the excess rate table. The followin examples demonstrate how to modify and verify the policed DSCP tables in Hybrid and Native mod Example 8-29 shows how to map DSCP 24 to DSCP 8 in the mark-down table in Hybrid, for a switc PFC1.Example 8-30 shows how to map DSCP 24 to DSCP 8 for the normal rate policer and DSCP 2 DSCP 0 for the excess rate policer in Native mode.

Example 8-29. Modifying the Policed DSCP Mark-Down Table in Hybrid M

```
hybrid (enable) set qos policed-dscp-map 24:8

QoS policed-dscp-map set successfully.


hybrid (enable) show qos map config policed-dscp-map

DSCP - Policed DSCP map:

DSCP                              Policed DSCP

--------------------------------  ------------

                                0  0
```

```
                                1  1
                                2  2
                                3  3
                                4  4
                                5  5
                                6  6
                                7  7
                              8,24  8
(text omitted)
```

Example 8-30. Modifying the Policed DSCP Mark-Down Table in Native Mo

```
native(config)#mls qos map policed-dscp normal-burst 24 to 8
native(config)#mls qos map policed-dscp max-burst 24 to 0

native#show mls qos map | begin Policed-dscp
   Normal Burst Policed-dscp map:                                  (dscp= d1d2)
     d1 :  d2 0  1  2  3  4  5  6  7  8  9
     ------------------------------------
      0 :    00 01 02 03 04 05 06 07 08 09
      1 :    10 11 12 13 14 15 16 17 18 19
      2 :    20 21 22 23 08 25 26 27 28 29
      3 :    30 31 32 33 34 35 36 37 38 39
      4 :    40 41 42 43 44 45 46 47 48 49
      5 :    50 51 52 53 54 55 56 57 58 59
      6 :    60 61 62 63
   Maximum Burst Policed-dscp map:                                  (dscp= d1d2)
```

```
     d1 :  d2 0  1  2  3  4  5  6  7  8  9
     ------------------------------------
      0 :    00 01 02 03 04 05 06 07 08 09
      1 :    10 11 12 13 14 15 16 17 18 19
      2 :    20 21 22 23 00 25 26 27 28 29
      3 :    30 31 32 33 34 35 36 37 38 39
      4 :    40 41 42 43 44 45 46 47 48 49
      5 :    50 51 52 53 54 55 56 57 58 59
      6 :    60 61 62 63
(text omitted)
```

Policing

Policing is another feature available on the Catalyst 6500. Policing enables the administrator to cor utilization for certain applications. This guarantees the necessary bandwidth for voice and video an mission-critical applications. Policing is performed in hardware on the PFC without impacting switcl Policing cannot occur on the 6500 platform without a PFC and is currently only supported for ingres PFC version within the platform determines the extent of the policing functionality. The Catalyst 65 single-rate and two-rate policing.

The purpose of this section is to explain policing operation, as it pertains to the Catalyst 6500. It in discussion of microflow and aggregate policers, single-rate policing on the PFC1 and PFC2, and two on the PFC2. Finally, the section includes numerous configuration examples and various show comı verify operation.

Microflow and Aggregate Policers

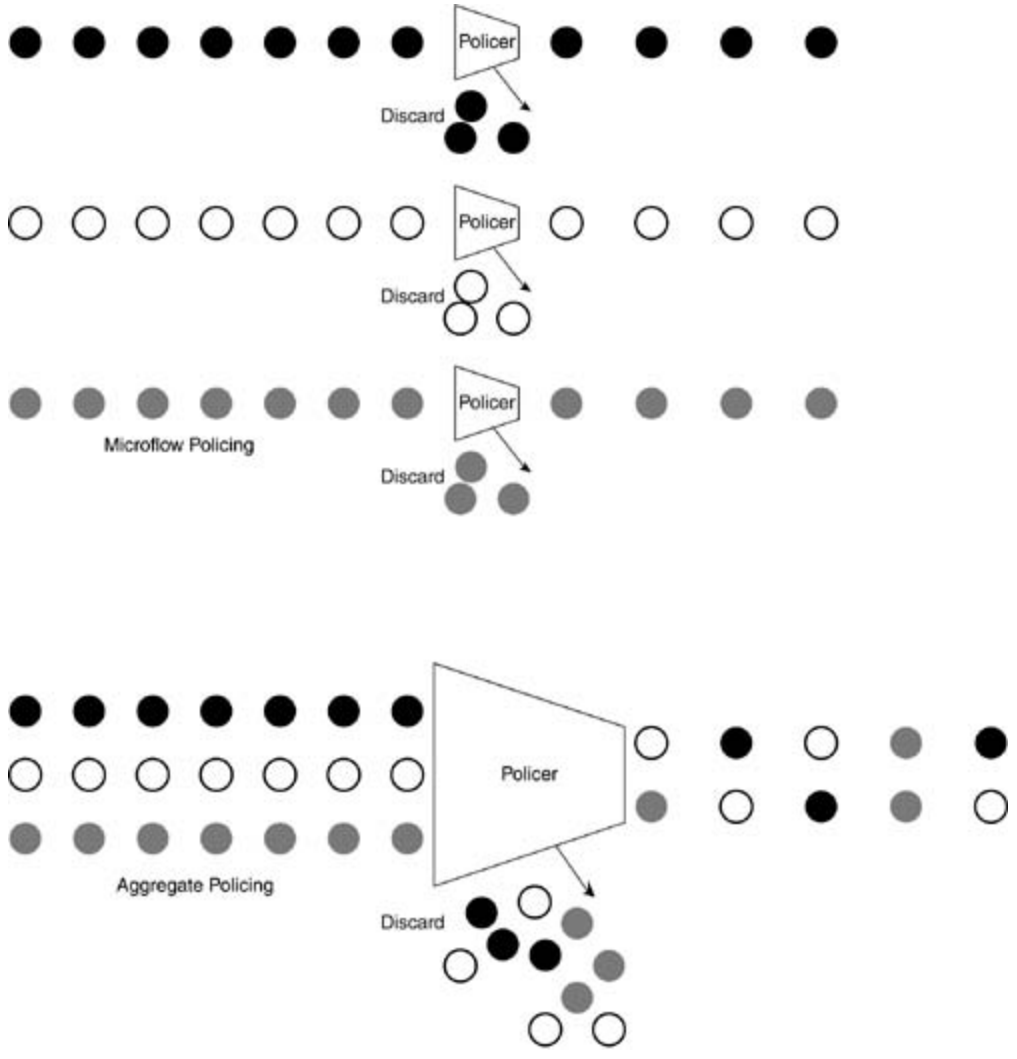
Two types of policers are available on the Catalyst 6500: microflow and aggregate policers.

Microflow policers limit the bandwidth consumed by individual flows on a port-by-port or interface very specific and can be defined using Layer 3 source and destination addresses, Layer 4 protocol t 4 source and destination port numbers. The bandwidth limitation is applied to each flow matching defined in the ACLs separately. The Catalyst 6500 can support up to 63 microflow policers.

Aggregate policers limit an aggregate of individual flows across multiple ports or interfaces, or on a interface, to one specified rate. The shared aggregate policer polices all traffic to a configured rate which the policer is applied. On the other hand, the interface aggregate policer polices all flows for interface. Up to 1023 aggregate policers can be defined on the Catalyst 6500.

Consider the following example. Assume there is a microflow policer defined limiting the bandwidth for certain IPTV streams to 1.5 Mbps. After the policer has been defined, it is applied to the approp in this case ports 5/1 and 5/3. In this case, although one microflow policer has been defined and a two separate ports, all IPTV traffic matching the configured criteria is limited to 1.5 Mbps per flow. one flow exists on port 5/1 and two flows exist on port 5/3, each flow is limited to 1.5 Mbps, for a c of 4.5 Mbps. Aggregate policers differ slightly. Consider the same situation, but now the configured aggregate policer. In this case, the combined inbound flows for port 5/1 and port 5/3 are considere a total of 1.5 Mbps. Any traffic exceeding this rate is classified as nonconforming and subsequently

Figure 8-8. Microflow Versus Aggregate Policer



By default, microflow policing only impacts routed traffic. To police bridged traffic, set qos bridge policing enable {*VLAN*} must be configured in Hybrid mode. The equivalent command for policing in Cisco IOS is mls qos bridged, which is configured under the VLAN interface.

Single-Rate Policing

Single-rate policing is supported with the PFC1 starting with Release 5.3(1) for Hybrid mode and R 12.0(7)XE for Native mode. Release 6.1(1) for Hybrid mode and 12.1(5c)EX, and 12.1(8a)E for Na introduced support for the PFC2. This is notable because the PFC2 incorporates an ASIC, allowing p at dual rates. The PFC2 offers the excess rate and burst configuration options as an additional enha result, single-rate microflow and aggregate and two-rate aggregate policing are supported on the I minimum versions of code. This section focuses on the operation and configuration of the single-ra Hybrid and Native modes.

The policing function on the Catalyst 6500 controls the data after it has completely entered a port a the packet buffer. When the data arrives, it is compared to the QoS ACL entries configured. If the d matches an ACL entry that is mapped to a policer, the policer allocates one token for every data bi of the IP and IPX packets entering the switch. Sequentially, the policing function occurs after the in value has been derived for the respective frame. If the data is not IP or IPX, the entire Layer 2 frar

the D-bus, is used by the policing counter. This includes the source and destination MAC addresses Layer 2 encapsulation, including packet padding and *cyclic redundancy check* (CRC) information. IS encapsulation information is excluded. The size of the frame is important to consider when configu size. Ensure that enough tokens are available in the token bucket to sustain the maximum-sized fr by the policer.

> NOTE
>
> The ACLs for the policers are defined in the same manner as previously outlined. However, it i important to remember the effects the various trust keywords have on deriving the internal DS The trust keywords impact any policies defined in the policed DSCP tables.

After the switch has determined that the arriving frame meets the criteria defined in the QoS ACL, to establish whether the frame conforms or violates the configured contract. When configuring the administrator defines the type, name, rate, and burst depth.

The refresh interval, on the other hand, is already defined and fixed for the Catalyst 6500. The tok refreshed every .25 ms, which equates to 4000 updates every second. The number of tokens place bucket during every interval depends on the configured contract rate. If enough tokens are present bucket to accommodate an arriving frame, the data is considered conforming. As a result, no actio the frame, which is then forwarded to the egress port or interface. Because the frame was in profile also charged the appropriate amount of tokens, equal to the size of the transmitted frame in bytes. policer, microflow or aggregate, determines a frame is out of profile, that frame is marked down or according to the policy configured for that specific policer. If only a microflow policer is configured, profile action is derived from the microflow policy. If only an aggregate policer is utilized, it specifi profile action. If both policer types are defined, the most stringent decision is the one applied. The summarizes these rules:

- If the microflow policer returns an out-of-profile decision, mark or drop according to the micr rule.
- If the aggregate policer returns an out-of-profile decision, mark or drop according to the aggr rule.
- If either policer returns an out-of-profile decision, mark or drop according to the policing rule that returned the out-of-profile decision.

If both policers return an out-of-profile decision and the rule of either one is to drop, the packet is otherwise the packet is marked down and transmitted. The new DSCP value is taken from the polic mapping table. By default, frames are not marked down. They are mapped to the equivalent value values can be modified by the administrator, and customized to meet specific network requirement demonstrated earlier, the mapping table is applied globally to the switch. For further information, r section on mapping. Neither the microflow nor the aggregate policer is charged for the packet if eit a policing decision.

A Theoretical Example of Policing Behavior

As discussed, tokens in the token bucket are replenished at a regular fixed interval. The rate at wh placed into the bucket is once every .25 ms. As a result, every second has 4000 refresh intervals. D .25-ms interval, (.00025(s) * rate(bps)) tokens are placed into the bucket. If 10 Mbps is the config

tokens are placed into the token bucket every .25 ms, assuming tokens need to be replaced. All tol the capacity of the bucket are discarded. When selecting a burst size, it is important to ensure it is be equal to or higher than the desired policing rate.

If you set the rate to 10 Mbps, an acceptable bucket depth, based solely on rate, is as follows:

10,000,000 (bps) * .25 (ms) = 2500, which you would round up to 3 kb

However, packets that are transmitted through the network are never fixed in size; instead, they v The possibility exists that a maximum frame size (1518 bytes) can occur at any interval. Therefore must be at least equal to the average or largest packet size. Consider the following example to der behavior of the policer.

In this scenario

Time = .25 ms; therefore the following example represents 10 refresh cycles.

Assume there is a 100-Mbps interface, and the intent is to police all ingress traffic for the port dow and drop all traffic in excess of this rate. The bucket starts at full capacity, and the bucket depth is 13 kb, which accounts for large packets arriving on the interface (1518 * 8 = 12144; rounded = 13 results in the following configuration:

```
set qos policer aggregate Test rate 10000 burst 13 drop
```

or

```
mls qos aggregate-policer Test 10000000 1625 violate-action drop
```

Table 8-14. Operation of Aggregate Policer

Time	Packet Size (Bytes)	Tokens Added[*]	Token Bucket Size (Bits)	Tokens Removed (Bits)	Final Bucke Size
0	0	0	13000	0	13000
1	751	0	13000	6008	6992
2	600	2500	9492	4800	4692
3	950	2500	7192	0	7192
At t=3 No tokens are removed from the bucket, because there are not enough tokens to entire packet. As a result, a packet arriving at t=3 is dropped due to policing.					
4	700	2500	9692	5600	4092
5	830	2500	6592	0	6592
At t=5 the ingress packet is dropped due to policing.					
6	1050	2500	9092	8400	692
7	450	2500	3192	0	3192
8	675	2500	5692	5400	292
9	350	2500	2792	0	2792
10	725	2500	5292	0	5292

[*] Note No tokens are added at t=1 because the token bucket is at full capacity.

Due to the flow-control mechanisms utilized by TCP, the burst parameter setting based on the calc demonstrated in the example is too low. The result is a "sawtooth" effect seen when TCP attempts rate. The policer throttles the TCP stream by dropping frames, causing TCP to initialize its slow sta As a result, the average rate achieved by the TCP session is considerably lower than the configured policer. Therefore, when policing TCP streams, as an initial step, it is recommended to double the b (2 * burst) to effectively police at the configured rate. As described section "Burst Size" in Chapter QoS Features Available on the Catalyst 4000 IOS Family of Switches and the Catalyst G-L3 Family the round-trip time is known for a particular TCP stream, that value can be used to determine a bu Therefore, the preceding formula can be substituted with (2 * RTT * rate) to determine a baseline However, it is important to consider that policing is not designed or intended to be application-frier either transmits a conforming frame, or polices based on the configured contract. The (2 * burst) c rate) recommendation is meant to be only a starting point. To obtain the desired results, you may modify the burst parameter after observing the behavior of the traffic,

Configuring the Single-Rate Policer

To configure and verify single-rate policing in Hybrid and Native modes, use the following comman

(Hybrid)

```
set qos policer {microflow | aggregate} {name}rate {rate}burst {burst} {policed-ds
show qos policer {config | runtime} {all | aggregate | microflow} [name]
show qos statistics { l3stats | {aggregate-policer [name]}}
```

(Native)

```
mls qos aggregate-policer {name} {rate} [{burst}]violate-action {drop |
➥policed-dscp-transmit | transmit}
police aggregate {name}
show mls qos aggregate-policer [name]
show mls qos ip {type num}
```

When the aggregate policer is configured in Native mode, the command is applied under the policy aggregate policer is referenced by configuring police aggregate {*name*}. The {*name*} parameter 31 characters long and is case sensitive. The policer name may include a–z, A–Z, 0–9, the dash ch underscore character (_), and the period character (.). Policer names must start with an alphabetic They must also be unique across all microflow and aggregate policers. Keywords cannot be utilized command as a policer name.

The valid range for the rate and erate parameters is 32 kbps through 32 Gbps, for Hybrid. For Nati maximum rate is 4 Gbps. To classify all traffic as out of profile, set the rate parameter to zero (0). range is consistent, the values are inputted slightly different between operating systems. In Hybrid entered in kbps. As a result, if the desired rate were 1 Mbps, the configured value would be 1000. I mode the rate is configured in bps. Therefore, 1 Mbps is configured as 1000000. Table 8-15 lists sc values and levels of granularity for the various ranges.

Table 8-15. Rate Configuration Table

Rate Range	Granularity	Rate Range
1 to 1024 (1 Mbps)	32768 (32 kbps)	65537 to 131072 (128 Mbps)
1025 to 2048 (2 Mbps)	65536 (64 kbps)	131073 to 262144 (256 Mbps)
2049 to 4096 (4 Mbps)	131072 (128 kbps)	262145 to 524288 (512 Mbps)
4097 to 8192 (8 Mbps)	262144 (256 kbps)	524289 to 1048576 (1 Gbps)
8193 to 16384 (16 Mbps)	524288 (512 kbps)	1048577 to 2097152 (2 Gbps)
16385 to 32768 (32 Mbps)	1048576 (1 Mbps)	2097153 to 4194304 (4 Gbps)
32769 to 65536 (64 Mbps)	2097152 (2 Mbps)	4194305 to 8000000 (8 Gbps)

Within each range, the hardware is configured with rate values that are multiples of the granularity valid range for the burst parameters is 1 kb through 32 Mb. Again, the burst values are configured different depending on the operating system. With Hybrid, burst is entered in *kilobits* (kb), whereas configures the burst in *bytes* (B). To maintain consistent QoS results, when applying a single police ports, standardize the trust state across all affected ports.

Notice the previous commands do not include configuration for the microflow policer in Native mod aggregate policer, which is configured in global configuration mode, the microflow policer is entere MQC. To enable microflow policing in Native mode, mls qos flow-policing must be enabled in glo configuration mode. By default, microflow policing is globally enabled in Native mode. You can veri issuing the command show mls qos. In addition to the microflow policer, it is also possible to con interface aggregate policer. Native mode offers the option of configuring two different aggregate p previously defined aggregate policer behaves the same as the policer defined in Hybrid mode. The aggregate policer, on the other hand, applies to ingress traffic only on the port to which it is applie microflow and per-interface aggregate policers are configured under the policy map class using the commands:

(Native IOS)

(Microflow Policer)

```
police flow {rate} [{burst}] [conform-action {drop | set-dscp-transmit {DSCP}|
set-prec-transmit{prec}| transmit}] [exceed-action {drop | policed-dscp-transmit
```

(Aggregate Policer)

```
police {rate} [{burst}] [conform-action {drop | set-dscp-transmit {DSCP}|
set-prec-transmit{prec}| transmit}] [exceed-action {drop | policed-dscp-transmit
```

As mentioned in the "Classification And Marking" section, prior to 12.1(12c)E1 set ip precedence dscp were not available options. As a result, to mark incoming traffic, a policer was required. Mark accomplished using the conform-action option. Under this configuration option, it is possible to m existing DSCP or IP precedence value. The frame is then forwarded to the appropriate egress port according to the CoS derived from the new DSCP setting. The following examples demonstrate how single-rate policing in both Hybrid and Native mode. The intent of these examples is to demonstrat lower-priority applications, ensuring sufficient bandwidth exists for mission-critical applications, pa and video.

In the Hybrid mode example provided, an aggregate policer controls bandwidth consumption for w traffic is limited to 1 Mbps. All HTTP traffic exceeding the normal rate is dropped. Figure 8-9 shows configured in Hybrid mode. In the Native example, the same match criteria and limitations apply. however, a microflow policer is used. Figure 8-10 shows the example configured for Native mode. the behavior of the different policers, a traffic generator is configured. The generator is attached vi to the port to which the policer is applied. Three separate flows are created on the traffic generator for TCP port 80. In addition, each flow is set to generate packets at a rate of 2 Mbps, for a total of

Figure 8-9. Single-Rate Aggregate Police

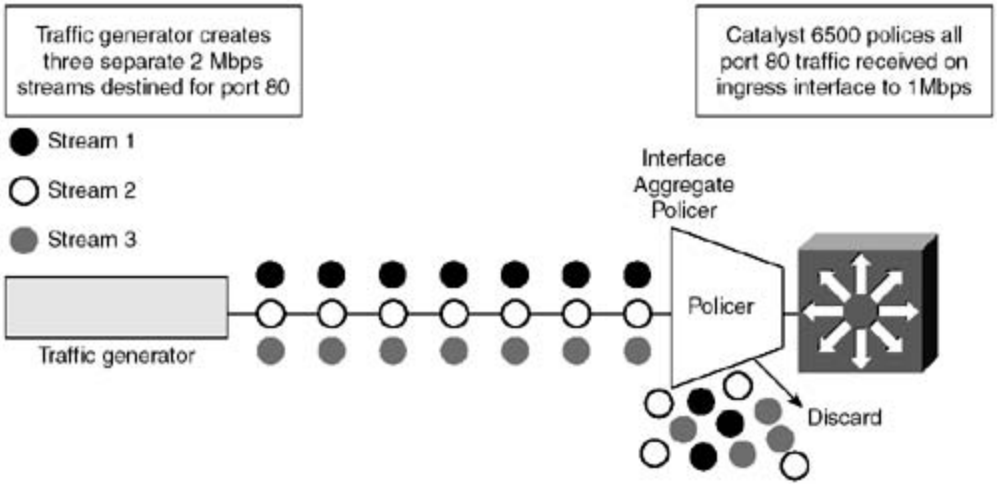
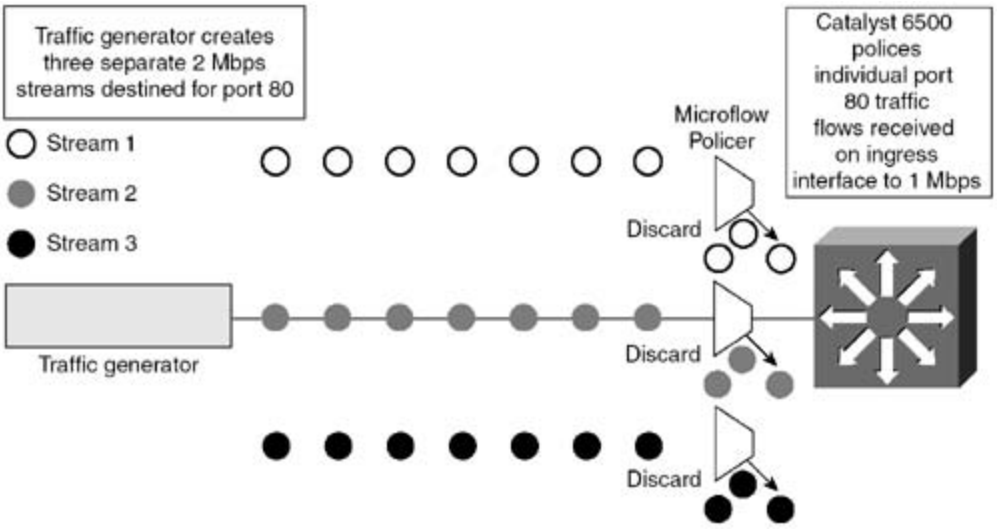


Figure 8-10. Single-Rate Microflow Policer



Example 8-31. Configuring a Single-Rate Aggregate Policer in Hybrid Mod PFC2

```
hybrid (enable) set qos policer aggregate HTTP-police rate 1000 burst 13 drop

QoS policer for aggregate HTTP-police updated successfully.

Rate is set to 992, erate is set to 992 burst is set to 13 and eburst is set to 13

ware due to hardware granularity.

hybrid (enable) set qos acl ip HTTP-traffic dscp 0 aggregate HTTP-police tcp any a

HTTP-traffic editbuffer modified. Use 'commit' command to apply changes.

hybrid (enable) commit qos acl HTTP-traffic

QoS ACL 'HTTP-traffic' successfully committed.

hybrid (enable) set qos acl map HTTP-traffic 5/10

ACL HTTP-traffic is successfully mapped to port 5/10.
```

The first step is to define the policer with the appropriate parameters. Although 1 Mbps is the confi value is rounded down to the closest 32-kbps increment to conform to the hardware granularity for range. Next, to account for the largest potential frame size (1518 bytes), the value 13000 is chose The policer is then applied to the appropriate ACL. When the ACL is applied to the desired port, it i verify configuration and performance.

```
hybrid (enable) show qos info config 5/10

QoS setting in NVRAM:

QoS is enabled

Policy Source of port 5/10: COPS

Tx port type of port 5/10 : 2q2t

Rx port type of port 5/10 : 1q4t

Interface type: port-based

ACL attached: HTTP-traffic

The qos trust type is set to untrusted.

Default CoS = 1

(text omitted)
```

Based on the show qos info output for port 5/10, it is confirmed the interface is set for port-based ACL HTTP traffic has been applied. Further, note that the port has been left at the default untrusted discussed in the previous sections, frames arriving on port 5/10 are marked with the port default C scenario, however, this value is overwritten. Remember, the internal DSCP value is derived from th which equates to DSCP 0. Fortunately, because the port is untrusted, even if the default ACL DSCP modified, all traffic matching the ACL configured in the example is set to DSCP 0.

```
hybrid (enable) show qos policer config aggregate HTTP-police

QoS aggregate policers:

Aggregate name                   Avg. rate (kbps) Burst size (kb) Normal action

------------------------------ ---------------- --------------- -------------

HTTP-police                                 1000              13 policed-dscp

                           Excess rate (kbps) Excess burst size (kb) Excess actio

                               ------------------ ---------------------- --------

                                            1000                     13 drop

                               ACL attached

                               ------------------------------------

                               HTTP-traffic
```

show qos policer verifies the configuration for the defined policer HTTP-police. Notice that becaus
rate policer is configured on a Catalyst 6500 equipped with a PFC2, excess rate and burst are visib
Because the excess rate and burst are equal to the normal rate and burst, however, the excess pol
not invoked.

```
hybrid (enable) show qos statistics aggregate-policer HTTP-police

QoS aggregate-policer statistics:

Aggregate policer               Allowed packet Packets exceed Packets exceed

                                count          normal rate    excess rate

------------------------------- -------------- -------------- --------------

HTTP-police                              42768         208811         208811
```

Finally, reviewing show qos statistics for the aggregate policer reveals the desired performance.
251,579 packets were forwarded to the port. This is confirmed by adding the total number of packe
the excess rate (208,811) to the total number of packets conforming to the policing contract (42,7
percentage of conforming packets is computed (42,768 / 25,1579), it is evident the aggregate poli
as expected. Approximately, 1/6th of the total traffic conforms to the specified rate. The next exam
demonstrates the configuration and behavior of a microflow policer in Native mode.

Example 8-32. Configuring a Single-Rate Microflow Policer in Native Mod PFC2

```
native(config)#mls qos flow-policing

native(config)#access-list 101 permit tcp any any eq www

native(config)#class-map HTTP-traffic

native(config-cmap)#match access-group 101

native(config)#policy-map HTTP-police

native(config-pmap)#class HTTP-traffic
```

```
native(config-pmap-c)#police flow 1000000 2000
```

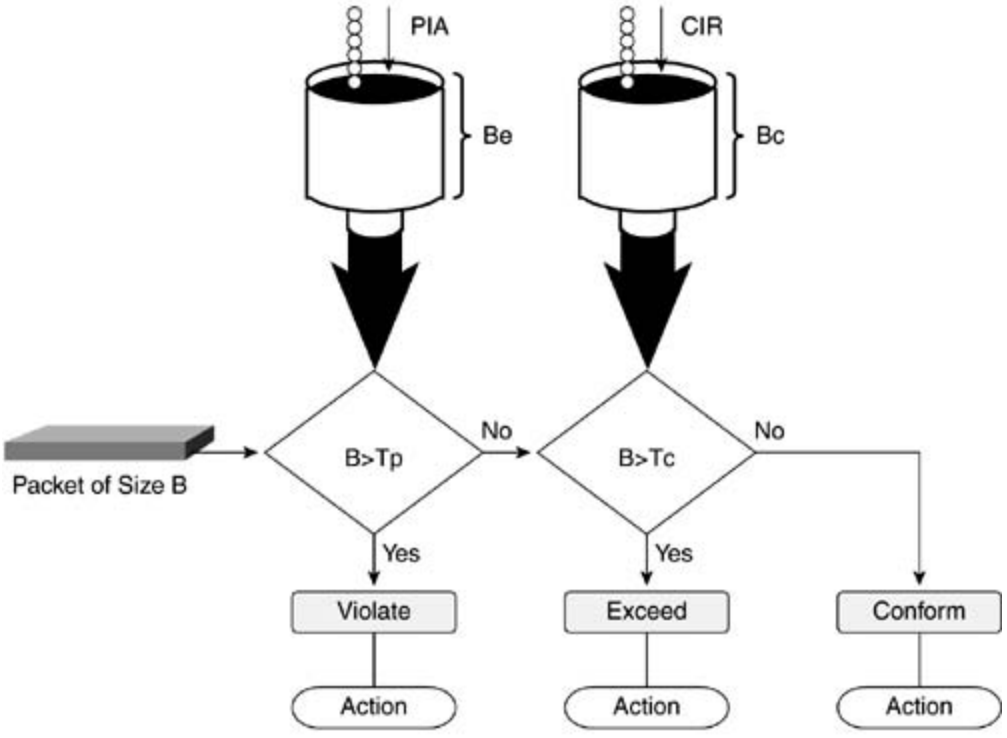
After the interesting traffic has been defined, in addition to the class maps and policy maps, the res
applied to the appropriate interface with the command service-policy input {*name*}. At this poin
now operational.

```
native#show mls qos ip fastEthernet 6/10

   [In] Policy map is HTTP-police   [Out] Default.

 QoS Summary [IP]:      (* - shared aggregates, Mod - switch module)

Int            Mod Dir Cl-map DSCP AgId Trust FlId   AgForward-Pk   AgPoliced-Pk

--------------------------------------------------------------------------------

Fa6/10           1 I HTTP-traf  0    3*  dscp    2         682858              0

native#show mls ip detail

Displaying Netflow entries in Supervisor Earl

DstIP           SrcIP           Prot:SrcPort:DstPort  Src i/f:AdjPtr

-----------------------------------------------------------------

20.20.20.2      192.168.10.1    tcp :www    :www      0   : 0

20.20.20.1      192.168.20.1    tcp :www    :www      0   : 0

20.20.20.3      192.168.30.1    tcp :www    :www      0   : 0

Pkts         Bytes       Age   LastSeen  Attributes

-------------------------------------------------

62605        30426030    273   15:07:31   L3 - Dynamic

62675        30460050    273   15:07:31   L2 - Dynamic

62677        30461022    273   15:07:31   L2 - Dynamic

    QoS     Police Count Threshold    Leak     Drop Bucket  Use-Tbl Use-Enable

-----------+------------+---------+-----------+----+-------+-------+----------+

  0x0          60278           0        0       NO   1798      NO       NO

  0x0          60207           0        0       NO   1857      NO       NO

  0x0          60206           0        0       NO   1640      NO       NO
```

Two-Rate Policing

The PFC2 provides additional policing enhancements beyond the capabilities of the PFC1. The PFC2 aggregate policing at dual rates. Dual-rate aggregate policing was introduced for the Catalyst 650(Software Release 6.1(1) and in Cisco IOS Release 12.1(8a)E. In addition to the traditional normal burst size, the PFC2 introduces an excess rate and an excess burst for the policer. With this configu drop indication applies to the excess rate, as opposed to the normal rate. Packets exceeding the nc always marked down, unless they also exceed the excess rate. If they exceed the excess rate, the p either marked down or dropped, depending on the administratively defined policy. Figure 8-11 sho operation of the two-rate policer.

Figure 8-11. Two-Rate Policer



To complement the policing enhancements offered by the PFC2, there are now two policed DSCP m corresponding to each rate for marking down out-of-profile packets, as demonstrated in the previo section. Operation of the two-rate policer on the Catalyst 6500 is analogous to the "two-rate three- described in IETF RFC 2698. As depicted in Figure 8-11, the two-rate policer adds a second bucket operation. Similar to the single-rate policer, both buckets use a token bucket mechanism; however operates independently of the other. The following commands are used to configure and verify ope two-rate policer on the Catalyst 6500 with a PFC2:

(Hybrid)

set qos policer aggregate {*name*}**rate** {*rate normal*}**policed-dscp erate** {*rate excess*}

  {**policed-dscp | drop**}**burst** {*burst normal*}**eburst** {*burst excess*}

show qos policer {**config | runtime**} {{**aggregate** [*name*]}**| all**}

show qos statistics { **l3stats | {aggregate-policer** [*name*]}}

64

(Native)

mls qos aggregate-policer {*name*} {*rate CIR*} [*burst CIR*] [*burst PIR*]**pir** {*rate PIR*}

[**conform-action** {**drop | set-dscp-transmit** {*DSCP*}**| set-prec-transmit** {*prec*}**| tran**

exceed-action{**drop | policed-dscp-transmit | transmit**}] [**violate-action** {**drop |**

policed-dscp-transmit |transmit}]]

police aggregate {*name*}

police {*rate CIR*} [[*burst CIR*] [*burst PIR*]]**pir** {*rate PIR*} [[**conform-action** {**drop |**

set-dscp-transmit {*DSCP*}**| set-prec-transmit** {*prec*}**| transmit**}] [**exceed-action** { **d**

set-dscp-transmit {*DSCP*}**| set-prec-transmit** {*prec*}**| policed-dscp-transmit | trans**

violate-action {**drop | policed-dscp-transmit | transmit**}]]

show mls qos

show mls qos aggregate-policer [*name*]

show mls qos ip {*type num*}

For Native mode, two aggregate policers are depicted. Recall from configuring the single-rate polic
aggregate-policer, when applied to multiple ports through a policy map class, polices all ports to
rate. The alternative aggregate policer polices traffic to the configured rate on a per-port basis. Wi
aggregate policer, the normal rate, or *committed information rate* (CIR), determines how many tol
in the bucket every .25 ms. The excess rate, or *peak information rate* (PIR), determines how many
placed into the excess bucket based on the same frequency. Both token buckets operate independe
other.

Unlike the single-rate policer, which offers two policing actions, the two-rate policer provides suppo
policing action. These three policing actions correspond to the green, yellow, and red transactions
2698.

The first policing action matches all traffic conforming to the normal and excess policing rates. Bec
packets conform to both rates, no action is required from the switch. In Hybrid, the switch maintai
QoS parameters assigned to the packet, which is forwarded to the egress scheduler. In Native mod
exists to transmit the conforming packet, assign a new IP precedence or DSCP value, or drop the p
drop keyword is specified for the conform-action, the exceed-action and violate-action are au
configured to drop.

The second policing action adheres to traffic exceeding the normal rate, but conforming to the exce
Hybrid mode, packets exceeding the normal rate, but conforming to the excess, rate are marked d
the normal rate policed DSCP table. In Native mode, the administrator can instruct the policing log
mark down, or forward traffic matching the criteria for the second policing action. In addition, in N
per-port aggregate policer also allows a new DSCP or IP precedence value to be assigned. The dro
not available in Hybrid for traffic conforming to the excess rate, but violating the normal rate. In N
however, if the drop keyword is specified, the resulting behavior can be compared to the single-ra

The third policing action defines the policy applied to all traffic violating the normal and excess rate
and Native modes, packets can either be dropped or marked and forwarded with a policed DSCP va
addition, Native mode provides the transmit keyword as an option for violating traffic. When defin
policies for the three different policing levels, subsequent policing policies cannot be less stringent
defined policies. If the policed-dscp-transmit keyword is specified for the exceed-action, for ex
transmit cannot be specified for the violate-action. The configured action must be equal or more

With the release of CatOS 7.2 and later versions, the eburst parameter is configured independentl
parameter. This provides additional autonomy when configuring the burst parameters. Prior to the
the administrator is limited to only configuring the burst option. The eburst value is derived from
value. The burst and eburst values define the bucket depth for the respective token buckets. The
buckets, controlled by the policing ASIC on the PFC, operate independently of each other. Tokens c
the normal bucket are only allocated to packets conforming to the normal rate, whereas tokens cor
excess bucket are only allocated to packets conforming to the excess rate. Tokens from the two bu
combined to service a packet. Enough tokens must be present in either bucket to service an entire
optionaleburst value is not explicitly configured, the burst size is set to the same value for both no
excess rate policers.

When configuring the two-rate policer, ensure the configured burst values are at least equal to the
largest serviced packet size. In addition, set the excess rate value to be greater than or equal to th
If the two policing rates are configured equally, the resulting behavior is a single-rate policer. In th
although there are two separate token buckets, one for normal rate and the other for excess rate,
are charged for a successfully transmitted packet. Therefore, if a packet arrives, which does not ex
normal rate, and the normal rate equals the excess rate, tokens amounting to the size of the packe
from both buckets. The following examples demonstrate configuring an aggregate two-rate policer
Native modes.

Example 8-33. Configuring an Aggregate Two-Rate Policer in Hybrid Mod

```
hybrid (enable) set qos policer aggregate HTTP-police rate 1000 policed-dscp erate
➥drop burst 13 eburst 13
QoS policer for aggregate HTTP-police updated successfully.
```

```
Rate is set to 992, erate is set to 1984 burst is set to 13 and eburst is set to 1
➥hardware due to hardware granularity.
hybrid (enable) set qos acl ip HTTP-traffic dscp 0 aggregate HTTP-police tcp any a
HTTP-traffic editbuffer modified. Use 'commit' command to apply changes.
hybrid (enable) commit qos acl HTTP-traffic
QoS ACL 'HTTP-traffic' successfully committed.
hybrid (enable) set qos acl map HTTP-traffic 5/10
ACL HTTP-traffic is successfully mapped to port 5/10.
hybrid (enable) show qos info config 5/10
QoS setting in NVRAM:
QoS is enabled
Policy Source of port 5/10: COPS
Tx port type of port 5/10 : 2q2t
Rx port type of port 5/10 : 1q4t
Interface type: port-based
ACL attached: HTTP-traffic
The qos trust type is set to untrusted.
Default CoS = 1
(text omitted)

hybrid (enable) show qos policer config aggregate HTTP-police
QoS aggregate policers:
Aggregate name                   Avg. rate (kbps) Burst size (kb) Normal action
-------------------------------- ---------------- --------------- -------------
HTTP-police                                 1000              13 policed-dscp
                           Excess rate (kbps) Excess burst size (kb) Excess actio
                               ------------------ ---------------------- --------
                                            2000                     13 drop
```

```
                                ACL attached
                                ------------------------------------
                                HTTP-traffic
hybrid (enable) show qos statistics l3stats
Packets dropped due to policing:          212586
IP packets with ToS changed:                1276
IP packets with CoS changed:              150636
Non-IP packets with CoS changed:               0
hybrid (enable) show qos statistics aggregate-policer HTTP-police
QoS aggregate-policer statistics:
Aggregate policer               Allowed packet Packets exceed Packets exceed
                                count          normal rate    excess rate
------------------------------- -------------- -------------- --------------
HTTP-police                             109443         205768         212586
```

Example 8-34. Configuring an Aggregate Two-Rate Policer in Native Mod

```
native(config)#access-list 101 permit tcp any any eq www
native(config)#class-map HTTP-traffic
native(config-cmap)#match access-group 101
native(config)#policy-map HTTP-police
native(config-pmap)#class HTTP-traffic
native(config-pmap-c)#police 1000000 2000 2000 pir 2000000 conform-action transmit
➥exceed-action policed-dscp-transmit violate-action drop
native#show mls qos
  QoS is enabled globally
  Microflow policing is enabled globally
```

```
(text omitted)
 ----- Module [1] -----
  QoS global counters:
    Total packets: 291876
    IP shortcut packets: 0
    Packets dropped by policing: 297504
    IP packets with TOS changed by policing: 151
    IP packets with COS changed by policing: 282786
    Non-IP packets with COS changed by policing: 0
native#show mls qos ip fastEthernet 6/10
   [In] Policy map is HTTP-police   [Out] Default.
 QoS Summary [IP]:      (* - shared aggregates, Mod - switch module)
Int            Mod Dir Cl-map DSCP AgId Trust FlId   AgForward-Pk   AgPoliced-Pk
--------------------------------------------------------------------------------
Fa6/10           1 I HTTP-traf  0    2   dscp    0         153203         297504
```

Congestion Management and Congestion Avoidance

After a frame has been classified, processed through all the applicable QoS policies, and a forwardi
decision has been made, the frame is then forwarded to the appropriate egress queue. At the trans
interface, the Catalyst 6500 employs congestion management and congestion avoidance features t
priority traffic has precedence when accessing the network during periods of link oversubscription.
buffers and more complex queuing mechanisms are normally found at egress interfaces. This is be
during peak traffic periods congestion is more common at these points. Congestion within the cam
attributed to link-speed mismatches and to aggregation points within the network. Although input
ensures certain critical traffic is given priority to the switch bus or fabric, congestion upon ingress i
common occurrence and does not provide guarantees to traffic exiting the switch. Because it is mo
to have either a single Gigabit Ethernet connection flowing downstream to a single Fast Ethernet co
or 24 to 48 Fast Ethernet connections flowing upstream to a single Gigabit Ethernet connection, it i
imperative to ensure mission-critical applications and voice traffic are guaranteed access to the net
these points, minimizing the end-to-end latency and jitter. For these examples, the switch backpla
not serve as a bottleneck for the network traffic; instead, the bottleneck is the outbound interface.
result, it is more efficient and critical to properly deploy congestion management and congestion av
techniques at the egress interface. Table 8-16 summarizes some of the egress congestion manager
congestion avoidance port capabilities.

Table 8-16. Congestion Management and Avoidance Mechanisms per M

Module	Transmit Ports	SupportsWRED	Priority Queue	Number ofWRRQue
WS-X6024	2q2t	No	No	2
WS-X6148	2q2t	No	No	2
WS-X6224	2q2t	No	No	2
WS-X6248	2q2t	No	No	2
WS-X6316	1p2q2t	Yes	Yes	2
WS-X6324	2q2t	No	No	2
WS-X6348	2q2t	No	No	2
WS-X6408	2q2t	No	No	2
WS-X6408A	1p2q2t	Yes	Yes	2
WS-X6416	1p2q2t	Yes	Yes	2
WS-X6501	1p2q1t	Yes	Yes	2
WS-X6502	1p2q1t	Yes	Yes	2
WS-X6516	1p2q2t	Yes	Yes	2
WS-X6524	1p3q1t	Yes	Yes	3
WS-X6548	1p3q1t	Yes	Yes	3
WS-X6816	1p2q2t	Yes	Yes	2

Congestion management on the 6500 involves associating different CoS levels with the available tr queues on the egress port. It also pertains to how the various transmit queues are scheduled and t frequency they are serviced. Figure 8-2 depicts the various congestion management and congestio avoidance features available at the output port. The Catalyst 6500 utilizes *weighted round-robin* (W output scheduling mechanism between the various queues. Congestion avoidance allows the switch monitor and manage the buffer utilization within the queue. The Catalyst 6500 implements two cor avoidance mechanisms, tail drop and WRED.

The purpose of this section is to cover the various congestion management and congestion avoidan techniques available within the Catalyst 6500. Congestion management focuses on mapping CoS v transmit queues and thresholds, configuring output queue scheduling WRR weighting factors, and c the transmit queue size ratio. The section concludes with congestion avoidance strategies, includin configuring tail-drop thresholds for transmit queues and configuring WRED thresholds for transmit Throughout the discussion, examples are provided demonstrating the necessary configuration steps behavior for Hybrid and Native modes.

Congestion Management

As was the case with the input queues and scheduling, output scheduling and congestion managem accomplished using individual port ASICs. Also similar to input scheduling, the Catalyst 6500 utiliz values to determine which transmit queue a departing frame is assigned. As demonstrated in the n section, the egress CoS value is derived from the internal DSCP value. In addition, at the same tim DSCP is mapped to the CoS, the ToS field in the IP header is rewritten with the internal DSCP valu result, upon egress the QoS setting is sustained in the Layer 2 trunk header and the Layer 3 IP pac header.

Refer back to Table 8-2, which depicts the transmit queue capabilities of the different linecards ava the Catalyst 6500. Many of the earlier 10/100 and Gigabit modules have two transmit queues, with configurable thresholds assigned to each queue. This configuration is denoted as 2q2t. More recent linecards incorporate an additional strict-priority queue. This queue preemptively services all frame with CoS 5 by default. As long as the priority queue is void of packets, the lower queues are servic Therefore, by default all voice traffic is sent to the priority queue and given preference over traffic i other queues. The addition of the priority queue, in this instance, changes the transmit port type to For newer 10/100 and 100-Mb modules, the queue structure differs slightly. These cards have four one strict-priority queue and three normal queues. In this instance, however, each queue only utili WRED threshold. This port type is represented by 1p3q1t. Finally, the 10 Gigabit linecards offer yet transmit port type. Specified as 1p2q1t, these port types support strict-priority queuing and two no queues, each with one WRED threshold.

Mapping CoS Values to Transmit Queues and Thresholds

After the linecard queuing structure has been determined, it is possible to modify how the CoS valu mapped to the various queues. Table 8-17 displays the default CoS distribution across the various queues.

NOTE

To verify the queuing capabilities of a specific port or interface, issue the show port capabili

{*mod* [/*port*]} in Hybrid or show interface capabilities [module {*mod#*}] in Native mode.

Table 8-17. Default CoS Queue Assignments for Egress Port Type

CoS Values	Transmit Queue
Default Queue Assignments: 2q2t Port Types	
0–1	1 Threshold 1
2–3	1 Threshold 2
4–5	2 Threshold 1
6–7	2 Threshold 2
Default Queue Assignments: 1p2q2t and 1p2q1t Port Types	
0–1	1 Threshold 1
2–3	1 Threshold 2
4,6	2 Threshold 1
7	2 Threshold 2
5	3
Default Queue Assignments: 1p3q1t Port Types	
0–1	1
2–4	2
6–7	3
5	4

As mentioned, the CoS values mapped to the various queues are derived from the internal DSCP va
as the frame traverses the switch. The DSCP value, in turn, originates from the classification policy
to the ingress port. As a result, it is imperative to carefully consider any ingress QoS policies, beca
ultimately impact how the frame is processed upon egress. To modify which queue a particular CoS
mapped to and verify those changes, use the following commands:

```
(Hybrid)

set qos map {port-type}tx {queue#} {thr #}cos {cos-list}

show qos info config {port-type}tx

(Native)

wrr-queue cos-map {queue#} {thr #} {CoS1 [CoS2] [CoS3] [CoS4] [CoS5] [CoS6] [CoS7]

show queueing interface {type num} [ | begin queue thresh]
```

> **NOTE**
>
> As you may recall from the input scheduling section, modifying the ingress CoS mapping characteristics for 1q4t port types affects the CoS mappings for 2q2t port types. The opposite holds true when modifying the 2q2t port types.

In Hybrid, when modifying the ingress and egress queue mappings, the configuration changes app ports associated with the designated port type. Therefore, if you modify the CoS assignments for 1 types, all modules with this transmit capability inherit the specified parameters. Examples 8-35 an demonstrate altering the default CoS assignments and then validating the changes made.

Example 8-35. Mapping CoS Value to Transmit Queue and Threshold in H Mode

```
hybrid (enable) set qos map 2q2t tx 2 1 cos 3

QoS tx priority queue and threshold mapped to cos successfully.


hybrid (enable) show qos info config 2q2t tx

QoS setting in NVRAM for 2q2t transmit:

QoS is enabled

Queue and Threshold Mapping for 2q2t (tx):

Queue Threshold CoS

----- --------- ---------------
```

```
1     1         0 1

1     2         2

2     1         3 4 5

2     2         6 7

(text omitted)

**Note:  Receive queue configuration matches transmit queue.


hybrid (enable) show qos info config 1q4t rx

QoS setting in NVRAM for 1q4t receive:

QoS is enabled

Queue and Threshold Mapping for 1q4t (rx):

Queue Threshold CoS

----- --------- ---------------

1     1         0 1

1     2         2

1     3         3 4 5

1     4         6 7

 (text omitted)
```

The preceding example represents the recommended configuration for ports without a priority queu voice-deployed environment. With no strict-priority queue, frames with CoS 5 remain mapped to th queue and first threshold. In this instance, frames marked with CoS 3 are mapped to the same que threshold as frames marked with CoS 4 and 5. Depending on the traffic levels, this configuration af frames marked with CoS 5 the least amount of potential delay, jitter, and drop probability (becaus the second, or high, queue is scheduled more frequently than traffic in the lower queue). This ensu frames with low delay tolerances are expedited, whereas traffic in the lower queue is buffered. Also depicted in the example, the CoS mapping parameters for 2q2t port types are duplicated for 1q4t p on the receive side.

Example 8-36. Mapping CoS Value to Transmit Queue and Threshold in N Mode

```
native(config-if)#wrr-queue cos-map 2 1 3
```

```
  cos-map configured on:  Gi2/1 Gi2/2 Gi2/3 Gi2/4 Gi2/5 Gi2/6 Gi2/7 Gi2/8

native(config-if)#wrr-queue cos-map 2 2 6

  cos-map configured on:  Gi2/1 Gi2/2 Gi2/3 Gi2/4 Gi2/5 Gi2/6 Gi2/7 Gi2/8


native#show queueing interface gigabitEthernet 2/1 | begin queue thresh

    queue thresh cos-map

    ---------------------------------------

    1     1      0 1

    1     2      2

    2     1      3 4

    2     2      6 7

    3     1      5

(text omitted)
```

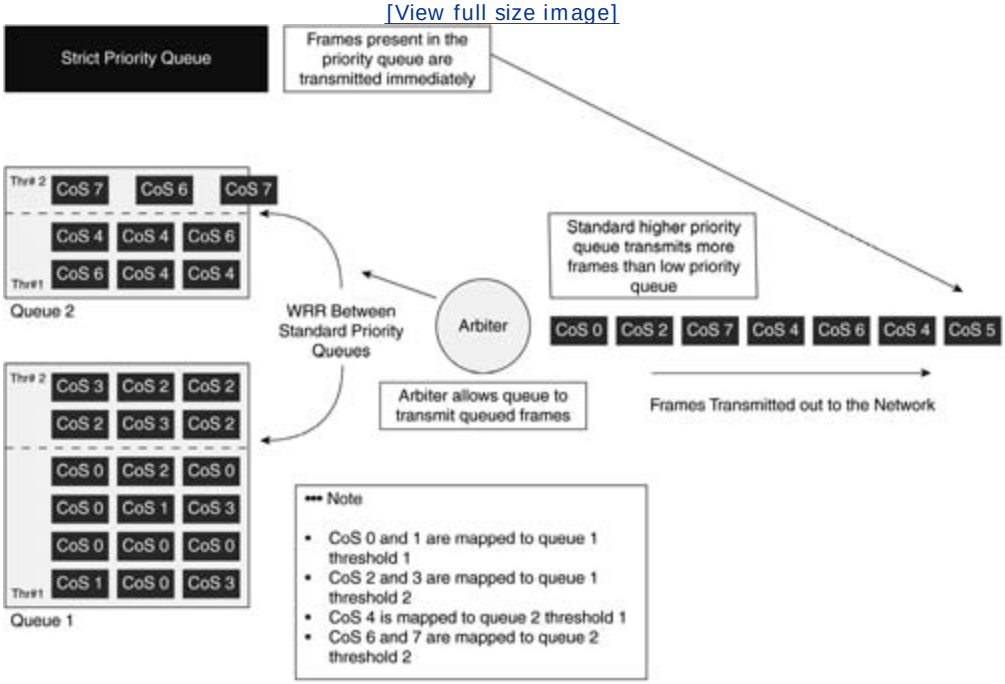
As noted earlier in the chapter, when you configure QoS on the Catalyst 6500, the ASICs are being programmed directly. Therefore, when ports and interfaces share the same ASIC, they inherit or sh configuration parameters. Example 8-36 demonstrates this behavior. Also the strict-priority queue directly above. By default, frames marked with CoS 5 are mapped to the strict-priority queue. To n CoS values to the strict-priority queue, utilize priority-queue cos-map 1 {*CoS*} for Native mode qos map {*port type*}tx {*queue#*} {*thr#*}cos {*CoS*} when running Hybrid. In Hybrid, queues are numbered starting with the lowest-priority queue and ending with the highest-priority queue. As a there are three available queues for 1p2q2t port types. The first queue represents the lowest stand priority queue available and offers the least amount of insurance for network traffic. However, que this case is the strict-priority queue, and as a result offers a low drop probability and minimal dela periods of congestion. After the CoS values have been mapped to a corresponding queue and thres important to analyze the scheduling between the different queues. This analysis ensures high-prior mission-critical traffic is receiving sufficient bandwidth during periods of congestion.

Configuring Output Queue Scheduling WRR Weighting Factors

Covered in Chapter 2, WRR is a scheduling mechanism used to determine the frequency with which queues are serviced. Unlike round-robin, which offers each participating queue equal access to the bandwidth, WRR enables the administrator to assign a weighting factor allowing queues to utilize a share of the bandwidth. If a queue has a higher weight, it is permitted to transmit more frames on network. Hence, it is serviced more frequently than other queues. With the exception of the strict-p queue, all port transmit queues participate in the WRR scheduling process. Figure 8-12 shows the operation among multiple queues.

Figure 8-12. Output Scheduling Using WRR

[View full size image]



NOTE

Remember that WRR and other scheduling mechanisms only affect traffic when congestion is experienced on a port. Also WRR does not rate limit traffic during periods of congestion. If onl priority traffic is present in the egress queue, even during congestion, it is permitted full acces the available bandwidth.

As depicted in the figure, the strict-priority queue on the Catalyst 6500 does not participate in WRF scheduling. When a frame enters the strict-priority queue, the queue is immediately given access t network. The switch checks the priority queue after each frame is transmitted from a standard-pric The remaining lower-priority standard queues are not permitted to transmit until the strict-priority void of any traffic. When that occurs, the remaining queues are permitted to transmit, and are sch the specified WRR fashion. Because the lower-priority queues service the lower-priority traffic, whe assigning weights ensure the higher-priority queues always receive a higher weighting factor. This guarantees more critical traffic receives preferential treatment. The following commands enable th administrator to assign different weighting factors to the various queues and to confirm the configu

(Hybrid)

set qos wrr {*port-type*} {*queue1 val*} {*queue2 val*}...{*queueN val*}

show qos info config {*port-type*}**tx**

(Native)

wrr-queue bandwidth {*queue1 val*} {*queue2 val*}...{*queueN val*}

show queueing interface {*type num*}

To determine the bandwidth allotted for a particular queue, use the same formula introduced in Ch

$(W/S) * B = n$

W represents the weight of the queue in question. This value is divided by the sum of all weights (S various transmit queues assigned to this interface. Finally, that value is multiplied by the total ban accessible to the port. The result (*n*) is the total available bandwidth for this transmit queue. By de weights assigned to the standard-priority queues for ports with two transmit queues are 100 for th priority standard queue and 255 for the high-priority standard queue. This implies the lower-priori permitted approximately 30 percent of the bandwidth, whereas the higher-priority queue receives remainder during periods of congestion. For interfaces with three transmit queues, the default valu 100, 150, and 200. This equates to 22 percent, 33 percent, and approximately 45 percent for the l medium-, and high-priority queues, respectively. Examples 8-37 and 8-38 show how to modify the round-robin weighting factors for the various transmit queues.

Example 8-37. Configuring WRR Scheduling Values in Hybrid Mode

```
hybrid (enable) set qos wrr 1p2q2t 85 255

QoS wrr ratio is set successfully.


hybrid (enable) show qos info config 1p2q2t tx

QoS setting in NVRAM for 1p2q2t transmit:

QoS is enabled

(text omitted)

WRR Configuration of ports with 1p2q2t:

Queue #  Ratios

-------  -----------------------------------
```

```
1        85

2        255
```

Example 8-38. Configuring WRR Scheduling Values in Native Mode

```
native(config-if)#wrr-queue bandwidth 70 125 255

  bandwidth configured on all 48 ports on slot 3.


native#show queueing interface fastEthernet 3/1

Interface FastEthernet3/1 queueing strategy:  Weighted Round-Robin

  Port QoS is enabled

  Trust state: trust IP Precedence

  Default COS is 0

  Transmit queues [type = 1p3q1t]:

    Queue Id    Scheduling  Num of thresholds

    -----------------------------------------

       1         WRR                 1

       2         WRR                 1

       3         WRR                 1

       4         Priority            1

    WRR bandwidth ratios:   70[queue 1] 125[queue 2] 255[queue 3]

(text omitted)
```

Configuring the Transmit Queue Size Ratio

Like the receive queue ratio, the transmit queue ratio enables the administrator to define the maxi amount of memory a queue is permitted to occupy for buffering egress traffic. The queues can be t meet the requirements of the specific flows traversing the port. If multiple queues are available on port, each queue must share the total amount of buffer space with the other queues. When allocati for the various queues in Hybrid, the values are expressed as percentages. As a result, the sum of

percentages must amount to 100 percent of the total available buffer space. In Native mode, the v
relative weights. Each queue is assigned a weighting factor, which can be used to calculate the app
percentage assigned to the particular transmit queue. For example, by default each 10/100 port or
X6348 or WS-X6148 linecard shares 112 KB among all queues for outbound traffic. Gigabit Etherne
share 439 KB of memory among all queues. To modify the percentage of buffer space or weights as
the various transmit queues, use the following commands:

(Hybrid)

set qos txq-ratio {*port-type*} {*queue1*} {*queue2*}...{*queueN*}

show qos info config {*port-type*}

(Native IOS)

wrr-queue queue-limit {*low-priority queue*} {*high-priority queue*}

show queueing interface {*type num*}

Table 8-18 depicts the default transmit queue ratio settings for 2q2t, 1p2q2t, and 1p2q1t port type

Table 8-18. Default Transmit Queue Ration Settings

Port Type	Standard Low-Priority Queue	Standard High-Priority Queue	Strict-Priori
2q2t (Hybrid)	80	20	N/A
2q2t (Native)	70	30	N/A
1p2q2t[*]	70	15	15
1p2q1t[*]	50	30	20
(Values assigned to queues in Hybrid mode are expressed as percentages. Values assigned to que Native mode are expressed as weights.)			

[*] Values apply to queues for both Hybrid and Native modes.

As shown in Table 8-18, a considerable portion of the buffer is assigned to the standard low-priorit
because the higher-priority queues receive more access to the available bandwidth, which is deterr
the WRR weighting factors. Because minimizing the total end-to-end delay is critical for real-time a

and voice, the higher-priority queues need to be serviced more frequently. As a result, they do not much buffer space, because buffering introduces additional delay and jitter. For low-priority queue buffer space must be allocated to ensure low-priority outbound frames are accommodated with mi drops, allowing sufficient time to service the higher-priority queues. When configuring the transmit ratio in Hybrid, the command line enables the administrator to configure all queues, including the priority queue. For Native IOS, it is only possible to explicitly define the weights for the standard lo high-priority queues. For 1p2q2t port types, the strict-priority queue must be configured to match for the standard high-priority queue. This requirement applies to switches operating in Hybrid mod Native IOS. Although in Hybrid the administrator can configure different percentages for the strict- and the standard high-priority queue, the switch still programs the values to be equal. This behavi hardware limitation applicable to 1p2q2t port types. The following examples demonstrate how to c the transmit queue ratio for the applicable port types in Hybrid and Cisco IOS.

Example 8-39. Configuring the Transmit Queue Ratio in Hybrid Mode

```
hybrid (enable) set qos txq-ratio 1p2q2t 60 20 20

QoS txq-ratio is set successfully.


hybrid (enable) show qos info config 1p2q2t tx

QoS setting in NVRAM for 1p2q2t transmit:

QoS is enabled

(text omitted)

Tx queue size ratio:

Queue #  Sizes - percentage

-------  ------------------------------------

1        60%

2        20%

3        20%
```

Example 8-40. Configuring the Transmit Queue Ratio in Native Mode

```
native(config-if)#wrr-queue queue-limit 60 25
```

```
native#show queueing interface tenGigabitEthernet 8/1

Interface TenGigabitEthernet8/1 queueing strategy:  Weighted Round-Robin

  Port QoS is enabled

(text omitted)

    queue-limit ratios:     60[queue 1]  25[queue 2]
```

Congestion Avoidance

Congestion avoidance involves proactively managing queues. The purpose is to avoid buffer exhau subsequently tail-dropping frames. The Catalyst 6500 offers *Weigthed Random Early Detection* (WF congestion avoidance mechanism. Table 8-16 lists the various linecards supporting WRED. WRED c very similar to *Random Early Detection* (RED). It utilizes configured thresholds to determine when randomly drop frames in a queue. Unlike RED, however, WRED is QoS aware. It uses the CoS valu assigned to the various frames to determine which frames to randomly drop. WRED is most efficien instances where multiple TCP streams are traversing the same port. WRED exploits TCP's windowin mechanism to avert congestion. WRED also helps prevent a phenomenon called global synchroniza Speaking in broad terms, *global synchronization* is a side effect of tail drop, where all TCP flows rec subsequently increase, their window size at the same time. A detailed discussion regarding global synchronization is beyond the scope of this text. For more information regarding the operation of W toChapter 2.

One other queue management technique the Catalyst 6500 employs is assigning tail-drop threshol available transmit queues. This enables the administrator to assign maximum high-level marks wit queue for specified marked frames. When CoS values are mapped to a specific queue and threshol demonstrated earlier, if the amount of frames exceeds the maximum specified threshold level, all s frames queued for transmission are tail dropped.

Configuring Tail-Drop Thresholds for Transmit Queues

The commands utilized for configuring the tail-drop thresholds for the receive queues are the same transmit queues.

```
(Hybrid)

set qos drop-threshold {port-type}tx queue {queue#} {thr1} {thr2}...{thrN}

show qos info config {port-type}tx

(Native)
```

```
wrr-queue threshold {queue#} {thr1} {thr2}...{thrN}

show queueing interface {type num} [ | begin queue tail-drop]
```

Currently, only 2q2t port types provide configurable tail-drop thresholds. Table 8-19 lists the defau threshold values.

Table 8-19. Default Tail-Drop Thresholds for 2q2t Port Types

Standard Low-Priority Queue	Standard High-Priority Queue
Threshold 2: 100% capacity	Threshold 2: 100% capacity
Threshold 1: 80% capacity	Threshold 1: 80% capacity

The transmit queue ratio defines the amount of memory allocated to each queue. The tail-drop thr define the maximum levels within the allotted space for frames marked with a particular CoS. Usin default values, consider frames with CoS values mapped to the first threshold in the low-priority qu When the available buffer reaches 80-percent capacity, all subsequent frames mapped to this thres tail dropped. The same applies to frames mapped to the second threshold. As soon as the queue re 100-percent capacity, all consecutive frames mapped to this queue are tail dropped. In addition to interfaces, strict-priority queues utilize tail drop to manage the queue. However, the thresholds for queue are nonconfigurable. The level is fixed at 100-percent capacity. Examples 8-41 and 8-42 de configuring and verifying the tail-drop thresholds for 2q2t type ports.

Example 8-41. Configuring Transmit Queue Tail-Drop Thresholds in Hybr

```
hybrid (enable) set qos drop-threshold 2q2t tx queue 1 65 100

Transmit drop thresholds for queue 1 set at 65% 100% .

hybrid (enable) set qos drop-threshold 2q2t tx queue 2 85 100

Transmit drop thresholds for queue 2 set at 85% 100% .

hybrid (enable) show qos info config 2q2t tx

QoS setting in NVRAM for 2q2t transmit:

QoS is enabled

(text omitted)
```

```
Tx drop thresholds:

Queue #  Thresholds - percentage

-------  ---------------------------------------

1        65% 100%

2        85% 100%
```

Example 8-42. Configuring Transmit Queue Tail-Drop Thresholds in Nativ

```
native(config-if)#wrr-queue threshold 1 65 100

  threshold configured on:  Gi8/1 Gi8/2 Gi8/3 Gi8/4 Gi8/5 Gi8/6 Gi8/7 Gi8/8

native(config-if)#wrr-queue threshold 2 85 100

  threshold configured on:  Gi8/1 Gi8/2 Gi8/3 Gi8/4 Gi8/5 Gi8/6 Gi8/7 Gi8/8

native#show queueing interface gigabitEthernet 8/1

Interface GigabitEthernet8/1 queueing strategy:  Weighted Round-Robin

  Port QoS is enabled

  Port is untrusted

  Default COS is 0

  Transmit queues [type = 2q2t]:

(text omitted)

queue tail-drop-thresholds

 --------------------------

 1     65[1] 100[2]

 2     85[1] 100[2]
```

One limitation expressed earlier regarding tail drop is its interaction with TCP flows. Tail drop is no conducive to TCP traffic, because it could potentially drop multiple frames from numerous TCP stre could lead to global synchronization. The result of this behavior is suboptimal bandwidth utilization attempts to address this issue.

Configuring WRED Thresholds for Transmit Queues

WRED consists of two different thresholds. In addition to a maximum tail-drop threshold, this cong avoidance mechanism enables the administrator to configure a lower threshold. The lower threshol the level to initiate randomly dropping specified frames. Frames are not dropped if buffer utilizatio the lower bound. When the buffer usage for the queue reaches the specified lower threshold, it sta dropping frames marked with the CoS mapped to the particular threshold. As the buffer continues drop rate increases at a linear rate, until it reaches the maximum defined threshold. When the ma level is attained, all additional frames assigned to the queue and WRED threshold are tail dropped. following commands configure the WRED thresholds for the designated queues:

(Hybrid)

set qos wred {*port-type*}**tx queue** {[*thr1Lo*:]*thr1Hi* [[*thr2Lo*:]*thr2Hi*]}

show qos info config {*port-type*}**tx**

(Native)

wrr-queue random-detect min-threshold {*queue#*} {*thr1*} [{*thr2*}]

wrr-queue random-detect max-threshold {*queue#*} {*thr1*} [{*thr2*}]

show queueing interface {*type num*} [**| begin random-detect**]

Because UDP streams do not incorporate any flow-control mechanisms, WRED has very little effect type of traffic. The default WRED thresholds assigned to 1p2q2t type interfaces are depicted in the

Table 8-20. Default WRED Thresholds for 1p2q2t Standard Queue

Standard Low-Priority Queue	Standard High-Priority Queue
Threshold 2 Low: 70 High: 100	Threshold 2 Low: 70 High: 100
Threshold 1 Low: 40 High: 70	Threshold 1 Low: 40 High: 70

For 1p2q1t and 1p3q1t port types, 70 percent is the default minimum threshold, and 100 percent i default maximum threshold for each queue. When configuring the WRED thresholds, the high value lower than the low value. However, the two values can be equal. By configuring the high and low v equal, you create a tail-drop threshold. This tail-drop threshold may be useful, for reasons previou mentioned, if the traffic mapped to a queue and threshold is composed solely of UDP type traffic. A priority queue defaults to a nonconfigurable 100-percent tail-drop threshold. The following exampl how to configure the WRED thresholds for the applicable queues.

Example 8-43. Configuring WRED Transmit Queue Thresholds in Hybrid M

```
hybrid (enable) set qos wred 1p2q2t tx queue 1 65:85 80:100

WRED thresholds for queue 1 set to 65:85 and 80:100 on all WRED-capable 1p2q2t por

hybrid (enable) set qos wred 1p2q2t tx queue 2 70:90 80:100

WRED thresholds for queue 2 set to 70:90 and 80:100 on all WRED-capable 1p2q2t por


hybrid (enable) show qos info config 1p2q2t tx

QoS setting in NVRAM for 1p2q2t transmit:

QoS is enabled

(text omitted)

Tx WRED thresholds:

Queue #  Thresholds - percentage

-------  ---------------------------------------

1        65%:85% 80%:100%

2        70%:90% 80%:100%
```

Example 8-44. Configuring WRED Transmit Queue Thresholds in Native M

```
native(config-if)#wrr-queue random-detect min-threshold 1 75

  WRED_threshold configured on all 48 ports on slot 3.

native(config-if)#wrr-queue random-detect min-threshold 2 75

  WRED_threshold configured on all 48 ports on slot 3.
```

```
native(config-if)#wrr-queue random-detect min-threshold 3 80
  WRED_threshold configured on all 48 ports on slot 3.

native#show queueing interface fastEthernet 3/1
Interface FastEthernet3/1 queueing strategy:  Weighted Round-Robin
  Port QoS is enabled
  Trust state: trust IP Precedence
  Default COS is 0
  Transmit queues [type = 1p3q1t]:
(text omitted)
    queue random-detect-min-thresholds
    ----------------------------------
      1    75[1]
      2    75[1]
      3    80[1]
```

Automatic QoS

Auto-QoS facilitates the QoS configuration process for the Catalyst 6500. At the time of writing, Auto-QoS is available with CatOS Software Release 7.5. Auto-QoS assists the administrator in configuring classification, congestion management, mapping, and congestion avoidance. The Auto-QoS feature on the Catalyst 6500 is divided into two components. Global Auto-QoS provides recommended QoS parameters on a switch-wide basis, whereas port-specific Auto-QoS configures parameters on a per-interface basis. With CatOS Release 7.5, Auto-QoS commands are specifically focused on voice-related applications. However, future software releases will provide additional functionality and capabilities.

> NOTE
>
> At the time of writing, Auto-QoS is not available in Cisco IOS for the Catalyst 6500. However, there are plans to integrate the feature in future Cisco IOS releases.

Global Auto-QoS

The global Auto-QoS command focuses on QoS parameters affecting the entire switch. To enable global Auto-QoS, enter the following command in enable mode on the supervisor:

```
set qos autoqos
```

When you enter the preceding command, any QoS commands previously configured are modified. Therefore, you should enable any Auto-QoS commands prior to manually altering the switch QoS properties. set qos autoqos first enables QoS on the switch, if it is not already enabled. The global Auto-QoS command does not alter any default QoS configurations applied at the port level. Because congestion management, congestion avoidance, and CoS-to-queue mappings depend on port types rather than specific ports, commands related to these mechanisms are incorporated into the global Auto-QoS command. Refer to Table 8-2 for a detailed list of the various modules and the corresponding port types, and review the section "Catalyst Feature Overview" in Chapter 3, which provides additional information on port type nomenclature.Tables 8-21 and 8-22 list the resulting commands executed when configuring global Auto-QoS.

Table 8-21. Added Global Auto-QoS Configuration Commands

[*] set qos enable

set qos policy-source local

set qos acl default-action ip dscp 0

set qos ipprec-dscp-map 0 10 18 24 34 46 48 56

set qos cos-dscp-map 0 10 18 24 34 46 48 56

set qos dscp-cos-map 0-7:0 8-15:1 16-23:2 24-31:3 32-39:4 40-47:5 48-55:6 56-63:7

[**] set qos policed-dscp-map 26:0

[**] set qos policed-dscp-map 46:0

[*] Command is enabled on the switch if it was not previously enabled.

[**] All normal and excess rate policed DSCP map values are configured for default values. The exception is DSCP 26 and DSCP 46.

Table 8-22. Added Global Auto-QoS Port Type Configuration Commands

2q2t/1q4t

set qos map 2q2t tx queue 2 2 cos 5,6,7

set qos map 2q2t tx queue 2 1 cos 1,2,3,4

set qos map 2q2t tx queue 1 1 cos 0

set qos drop-threshold 2q2t tx queue 1 100 100

set qos drop-threshold 2q2t tx queue 2 80 100

set qos drop-threshold 1q4t rx queue 1 50 60 80 100

set qos txq-ratio 2q2t 80 20

set qos wrr 2q2t 100 255

1p3q1t/1p1q0t

set qos map 1p3q1t tx 1 1 cos 0

set qos map 1p3q1t tx 2 1 cos 1,2

set qos map 1p3q1t tx 3 1 cos 3,4

set qos map 1p3q1t tx 3 0 cos 6,7

set qos map 1p3q1t tx 4 cos 5

set qos wrr 1p3q1t 20 100 200

set qos wred 1p3q1t queue 1 70:100

set qos wred 1p3q1t queue 2 70:100

set qos wred 1p3q1t queue 3 70:90

set qos map 1p1q0t rx 1 cos 0,1,2,3,4

```
set qos map 1p1q0t rx 2 cos 5,6,7
set qos rxq-ratio 1p1q0t 80 20
1p2q2t/1p1q4t
set qos map 1p2q2t tx 1 2 cos 0
set qos map 1p2q2t tx 2 1 cos 1,2,3,4
set qos map 1p2q2t tx 2 2 cos 6,7
set qos map 1p2q2t tx 3 cos 5
set qos txq-ratio 1p2q2t 75 15 15
set qos wrr 1p2q2t 50 255
set qos wred 1p2q2t queue 1 1 40:70
set qos wred 1p2q2t queue 1 2 70:100
set qos wred 1p2q2t queue 2 1 40:70
set qos wred 1p2q2t queue 2 2 70:100
set qos map 1p1q4t rx 1 1 cos 0
set qos map 1p1q4t rx 1 3 cos 1,2,3,4
set qos map 1p1q4t rx 1 4 cos 6,7
set qos map 1p1q4t rx 2 cos 5
set qos rxq-ratio 1p1q4t 50 255
set qos drop-threshold 1p1q4t rx queue 1 50 60 80 100
1p2q1t/1p1q8t
set qos map 1p2q1t tx 1 1 cos 0
set qos map 1p2q1t tx 2 1 cos 1,2,3,4
set qos map 1p2q1t tx 2 cos 6,7
set qos map 1p2q1t tx 3 cos 5
set qos txq-ratio 1p2q1t 75 15 15
set qos wrr 1p2q1t 50 255
set qos wred 1p2q1t queue 1 70:100
set qos wred 1p2q1t queue 2 70:100
set qos map 1p1q8t rx 1 1 cos 0
set qos map 1p1q8t rx 1 5 cos 1,2
set qos map 1p1q8t rx 1 8 cos 3,4
set qos map 1p1q8t rx 2 cos 5,6,7
set qos wred 1p1q8t queue 1 1 40:70
set qos wred 1p1q8t queue 1 5 60:90
set qos wred 1p1q8t queue 1 8 70:100
set qos rxq-ratio 1p1q8t 80 20
```

Port-Specific Auto-QoS

The port-specific Auto-QoS functions define, in one of two ways, the classification method used for traffic ingress to a specific port. One way involves specifying the type of device attached to the port, either a Cisco IP Phone or Cisco IP Softphone. The other method requires specifying the trust state for a particular port, either trust DSCP or trust CoS. Configuring port-specific Auto-QoS also assigns extended trust capabilities and CoS assignments to attached IP Phones using *Cisco Discovery Protocol* (CDP) version 2. If CDP version 2 is not enabled, the information cannot be communicated to an attached Cisco IP Phone. Refer to the section "Voice VLANs and Extended Trust" in Chapter 2 for additional information on CDP and extended port-based QoS capabilities. A PFC is a prerequisite for using either the trust DSCP or IP Softphone Auto-QoS option. To configure port-specific Auto-QoS, use one of the following commands:

set port qos {*mod/port*}**autoqos voip ciscoipphone**

set port qos {*mod/port*}**autoqos voip ciscosoftphone**

set port qos {*mod/port*}**autoqos trust dscp**

set port qos {*mod/port*}**autoqos trust cos**

The "Classification and Marking" section within this chapter discusses trust and internal DSCP in further detail. The `trust dscp` and `trust cos` classification keywords are used to classify all traffic based on arriving DSCP or CoS values. You can use the `cos` keyword for trusted uplink ports attached to devices capable of only modifying the trunk header of an Ethernet frame. This ensures previously assigned CoS values are honored at the arriving switch port, and subsequently mapped to a corresponding internal DSCP value. You can use the `dscp` keyword for uplink ports attached to devices capable of altering the DSCP field in the IP header. This capability is relevant, for instance, when connecting to a 3550, another 6500, or even a router. A PFC is required to use the `dscp` keyword. Table 8-21 depicts the CoS-to-DSCP and IP precedence-to-DSCP mapping values if the global Auto-QoS command is executed; Table 8-12 shows the default assigned mapping values.

The`voip ciscoipphone` and `voip ciscosoftphone` keywords simplify the process of configuring voice-specific QoS on a port. The `ciscoipphone` option configures the ingress port to trust the control and voice-bearer traffic received from an IP Phone. By default, this traffic is marked by the phone with CoS 3 and CoS 5, respectively. As mentioned previously, if CDP is enabled and the port terminates into a Cisco IP Phone, additional QoS capabilities are extended to support devices attached to a switchport on the phone. For additional information regarding the Cisco IP Phone, refer to the section "The Cisco IP Phone" in Chapter 2. The `ciscosoftphone` option configures an ingress port to trust voice traffic generated by the Cisco Softphone application. The Softphone application properly marks the control and voice traffic within the IP header. As a result, ACLs trusting the ingress DSCP values are configured for the port (and, based on worst-case performance parameters, policers are configured to regulate control and voice traffic

generated by the application). A PFC is required for configuring the ciscosoftphone keyword.

As discussed in the "Trust" section within this chapter, the WS-X6224/6248 and WS-X6324/6348 have certain hardware limitations that prevent them from supporting certain trust states on a port. As a result, additional intelligence has been added to the port-based Auto-QoS command to account for this limitation. For 1q4t/2q2t port types, when the trust cos, trust dscp, or voip ciscoipphone keywords are used, additional commands and ACLs are configured to support the configured feature. When specifying trust cos or voip ciscoipphone, the following additional commands are added for the port configuration:

```
set port qos {mod/port}trust trust-cos

set qos acl map ACL_IP-PHONES {mod/port}

set qos acl ip ACL_IP-PHONES trust-cos ip any  any

commit qos acl ACL_IP-PHONES
```

When specifying the trust dscp keyword, the following commands are added to complete the port configuration:

```
set port qos {mod/port}trust untrusted

set qos acl map ACL_IP-TRUSTDSCP {mod/port}

set qos acl ip ACL_IP-TRUSTDSCP trust-dscp ip any  any

commit qos acl ACL_IP-TRUSTDSCP
```

NOTE

The WS-X6148 module is capable of supporting trust-cos, trust-dscp, and trust-

ipprec. However, this capabilitiy is integrated in 6.4, and was not included in CatOS Release 7.5.

Tables 8-23 through 8-26 depict the executed commands when configuring port-specific Auto-QoS.

Table 8-23. Port-Specific Auto-Qos Commands for the ciscoipphone Keyword

set port qos {*mod*/*port*}policy-source local

set port qos {*mod*/*port*}port-based

set port qos {*mod*/*port*}cos 0

set port qos {*mod*/*port*}cos-ext 0

set port qos {*mod*/*port*}trust-ext untrusted

set port qos {*mod*/*port*}trust-device ciscoipphone

[*]set port qos {*mod*/*port*}trust trust-cos

[*] Enables receive queue thresholds for 1q4t port types. Additional commands are added for 1q4t/2q2t port types.

Table 8-24. Port-Specific Auto-QoS Commands for the ciscosoftphone Keyword

set port qos {*mod*/*port*}policy-source local

set port qos {*mod*/*port*}port-based

set port qos {*mod*/*port*}cos 0

set port qos {*mod*/*port*}cos-ext 0

set port qos {*mod*/*port*}trust-ext untrusted

set port qos {*mod*/*port*}trust-device none

set port qos {*mod*/*port*}trust untrusted

set qos policer aggregate POLICE_SOFTPHONE-DSCP46-mod-port rate 320 burst 20 policed-dscp

set qos policer aggregate POLICE_SOFTPHONE-DSCP24-mod-port rate 32 burst 8 policed-dscp

set qos acl ip ACL_IP-SOFTPHONE-mod-port trust-dscp aggregate POLICE_SOFTPHONE-DSCP46-mod-port any dscp-field 46

set qos acl ip ACL_IP-SOFTPHONE-mod-port trust-dscp aggregate POLICE_SOFTPHONE-DSCP24-mod-port any dscp-field 24

commit qos acl ACL_IP-SOFTPHONE-mod-port

set qos acl map ACL_IP-SOFTPHONE-mod-port mod/port

Table 8-25. Port Specific Auto-QoS for Trust DSCP

set port qos {*mod*/*port*}policy-source local

set port qos {*mod*/*port*}port-based

set port qos {*mod*/*port*}cos 0

set port qos {*mod*/*port*}cos-ext 0

set port qos {*mod*/*port*}trust-ext untrusted

set port qos {*mod*/*port*}trust-device none

[*]set port qos {*mod*/*port*}trust trust-dscp

[*] Command not executed for modules with 2q2t and 1q4t port types. Additional commands are added for 1q4t/2q2t port types.

Table 8-26. Port-Specific Auto-QoS for Trust CoS

set port qos {*mod*/*port*}policy-source local

set port qos {*mod*/*port*}port-based

set port qos {*mod*/*port*}cos 0

set port qos {*mod*/*port*}cos-ext 0

set port qos {*mod*/*port*}trust-ext untrusted

set port qos {*mod*/*port*}trust-device none

[*]set port qos {*mod*/*port*}trust trust-cos

[*] Enables receive queue thresholds for 1q4t port types. Additional commands are added for 1q4t/2q2t port types.

As demonstrated throughout the chapter, the Catalyst 6500 has a vast array of QoS capabilities. Auto-QoS expedites the configuration of these available QoS features and ensures critical applications, particularly voice traffic, are properly serviced on a switch-wide and per-port basis.

Summary

The Catalyst 6500 Family of switches offers the administrator wide-ranging flexibility for deploying QoS in the campus environment. The Catalyst 6500 series is fully capable of supporting the demands of today's converged campus networks. Given the versatility and broad range of QoS mechanisms of the Catalyst 6500s, you can position these devices at all layers of the campus topology. In addition, because the QoS functions are performed in hardware, the application of QoS policies results in no additional impact to the packet-forwarding rate. The following list summarizes the features and capabilities available on the Catalyst 6500 series devices:

- Includes QoS support for both Hybrid and Cisco IOS.
- Supports input scheduling based on trusted CoS.
- Offers input strict-priority queuing for expeditious handling of voice traffic on specific linecards.
- Classifies traffic based on CoS, IP precedence, and DSCP values.
- Performs classification and marking based on Layer 2, 3, and 4 defined ACLs.
- Supports up to 63 ingress microflow and 1023 ingress aggregate policers. At the time of this writing, egress policing is not supported.
- Provides single-rate policing and dual-rate policing capabilities, depending on PFC revision.
- Accomplishes complex egress congestion avoidance and congestion management techniques, including WRED, egress strict-priority queuing, and WRR scheduling.
- Automated QoS to facilitate the configuration process to support voice and other mission critical data on the network.

This chapter focused on the QoS capabilities of the supervisor engine, the PFC, and the various linecards.Chapter 9 further details the QoS functionality of the Catalyst 6500 family, by discussing the QoS features available when a FlexWAN and MSFC are incorporated into the switch.

Chapter 9. QoS Support on the Catalyst 6500 MSFC and FlexWAN

Chapter 8, "QoS Support on the Catalyst 6500," detailed the Catalyst 6500's *quality of service* (QoS) capabilities from a campus LAN perspective. Moreover, the chapter further detailed the QoS capabilities by describing the architecture and behavior of the individual port *application-specific integrated circuits* (ASICs) and the *Policy Feature Card* (PFC) on the Catalyst 6500. As discussed in Chapter 8, when equipped with specific linecards and a PFC, the Catalyst 6500 is highly versatile and offers many QoS mechanisms. These features help maintain the end-to-end QoS policies enforced in the LAN environment, ensuring the timely delivery of mission-critical applications.

This chapter focuses on the QoS mechanisms available on the *Multilayer Switch Feature Card* (MSFC) and FlexWAN. The MSFC and FlexWAN extend the 6500's QoS capabilities to the *metropolitan-area network* (MAN) and *wide-area network* (WAN). The intended use of the MSFC forwarding engine is strictly for switching packets that are not hardware switched by the PFC. Examples of software-switched traffic processed by the MSFC include packet flows requiring *Network Address Translation* (NAT), encryption, policy routing, and broadcast forwarding in specific software versions. The FlexWAN module, designated as part number WS-X6182, enables WAN capabilities on the Catalyst 6500. Integrating the MSFC and FlexWAN module into a Catalyst 6500 enables the administrator to consolidate QoS policies for WAN and LAN routers within one platform, simplifying the configuration process. This integration also offers the possibility for end-to-end QoS deployment, sustaining service levels for mission-critical applications across the entire network without the use of external routers, multiple platforms, and multiple versions of Cisco IOS.

The FlexWAN and MSFC employ IOS-based QoS concepts introduced in Chapter 2, "End-to-End QoS: Quality of Service at Layer 3 and Layer 2." This chapter discusses how the FlexWAN and MSFC support QoS features, and includes information on the following topics:

- MSFC and FlexWAN Architectural Overview
- QoS Support on the MSFC and FlexWAN
- Classification
- Marking
- Policing and Shaping
- Congestion Management and Scheduling
- Congestion Avoidance
- Summary

This chapter focuses on these QoS features available to software-switched traffic on the Catalyst 6500 and FlexWAN traffic flows. It also discusses the mechanisms used to ensure reliable delivery of traffic throughout the entire network via the use of the FlexWAN and MSFC. Prior to readingChapter 9, it is strongly recommended to review Chapter 2 and Chapter 5, "Introduction to the Modular QoS Command-Line Interface," to obtain the appropriate background.

MSFC and FlexWAN Architectural Overview

This section expands on some of the concepts presented in Chapter 8 within the section titled "Catalyst 6500 Architectural Overview." As discussed in Chapter 8, the MSFC, in conjunction with the PFC, is responsible for Layer 3 forwarding within the Catalyst 6500. With a Supervisor I Engine, the first packet in a flow is software switched by the MSFC. When the first packet is forwarded, the forwarding decision made by the MSFC is also programmed into hardware ASICs on the supervisor engine. This process is referred to as hardware-based *multilayer switching* (MLS). When the initial packet is forwarded in software by the MSFC, and the MLS flow is completed in hardware, all subsequent packets are switched by the PFC. With a Supervisor II Engine, the MSFC is not primarily responsible for forwarding packets. The MSFC builds the Layer 3 forwarding information, which is passed via an out-of-band channel to the PFC on the supervisor engine. The MSFC builds a *Cisco Express Forwarding* (CEF) table, which is copied directly into hardware on the PFC. Copying the CEF table to the PFC permits all forwarding decisions to be made in hardware. In the event an entry does not exist in the PFC for an arriving packet, it is then software switched by the MSFC. The switching process on the Supervisor II Engine is referred to as *CEF-based Layer 3 switching*.

> NOTE
>
> For additional information on MLS-based switching on the Supervisor I Engine, refer to the following technical document at Cisco.com:
>
> "Configuring IP Unicast Layer 3 Switching on Supervisor Engine I"
>
> For additional information on CEF-based switching on the Supervisor II Engine, refer to the following technical document at Cisco.com:
>
> "Configuring CEF for PFC2"

In addition to building the Layer 3 forwarding information, the MSFC is also responsible for applying configurations to the FlexWAN module. In addition, the MSFC switches and encapsulates non-IP traffic for both ingress and egress packets traversing the FlexWAN. The FlexWAN module binds to the active or designated MSFC within the 6500. As stated in the introduction, the FlexWAN extends the Catalyst 6500's reachability to the MAN and WAN. The FlexWAN is a single-slot module that can be integrated into the Catalyst 6500. The physical appearance of the FlexWAN module can be compared to a *Versatile Interface Processor* (VIP) for a 7500. Similar to the VIP, the FlexWAN has two bays, which accommodate two modular WAN port adapters. From the MSFC, the FlexWAN port adapters are configured identically to the way Cisco 7200 or 7500 port adapters are configured. The FlexWAN contains two VIPs capable of supporting one port adapter for each VIP. The FlexWAN port adapters are configurable strictly from the MSFC and not recognized by the switch in Hybrid mode. If running the Cisco Native IOS, the FlexWAN module interfaces are configured from the command line similar to any other available interface. Table 9-3 depicts the 7200/7500 WAN port adapters supported by the FlexWAN.

Unlike the VIP, which only incorporates a single processor and memory to control both bays, the FlexWAN module is comprised of two VIPs. Therefore, the FlexWAN module services the two installed port adapters with a dedicated processor and memory for each bay. Each processor

performs the encapsulation and QoS functions independently for its assigned slot. Also the MSFC communicates with each processor on the FlexWAN independently through the *Ethernet out-of-band channel* (EOBC). However, only system control information is exchanged via the EOBC. Routing and forwarding decisions are still maintained by the MSFC and central PFC. As a result, the FlexWAN uses the central *data bus* (D-bus) and *results bus* (R-bus) for forwarding and receiving data packets. This even applies for packets being forwarded out a different subinterface on the same port adapter. The D-bus, R-bus, and EOBC are introduced in Chapter 8.

For more detailed information regarding the architecture of the Catalyst 6500, consult the following technical document at Cisco.com:

> "Catalyst 6000 and 6500 Series Architecture"

Hardware and Software Requirements

As described in Chapter 8, the administrator has two software options available to support operations on the Catalyst 6500. The first option requires the administrator to load two separate software versions. One version supports the supervisor module, and the other version supports the MSFC. This configuration is referred to as *Hybrid mode*. The alternative arrangement requires only one version of software to be positioned on the platform. This one version sustains both the supervisor and the MSFC. This version of software is referred to as *Cisco Native IOS*.

For additional information on naming conventions and differentiating the various software versions for the Catalyst 6500, refer to the section titled "Identifying the Catalyst Software" in Chapter 8.

The MSFC is supported in both Native IOS and Hybrid. The only hardware requirement for supporting an MSFC is that the supervisor engine must incorporate a PFC. Table 9-1 depicts the minimum versions of software required for both Native IOS and Hybrid to support an MSFC with the respective supervisor engine.

Table 9-1. Minimum Software Versions for MSFC Support

	Sup I MSFC I	Sup I MSFC II	Sup II MSFC II
Native IOS	12.0(7)XE	12.1(2)E	12.1(8a)E
Hybrid	CatOS: 5.3(1)CSX Cisco IOS: 12.0(3)XE1	CatOS: 5.4(3) Cisco IOS: 12.1(2)E	CatOS: 6.1(1) Cisco IOS: 12.1(3a)E1

The FlexWAN requires both a PFC and an MSFC to be installed within the 6500. As stated previously, the FlexWAN is associated with the designated MSFC. In the presence of a redundant MSFC configuration, the FlexWAN interfaces appear only on the designated MSFC. As a result, a failover to the backup MSFC must occur for the FlexWAN interfaces to appear on the redundant MSFC. When the interfaces are available on the alternate MSFC, they can be configured and saved to memory. If a failover occurs in the future, the saved configuration on the redundant MSFC is used. Similar to the MSFC, the FlexWAN module is supported in either Hybrid or Native

IOS.Table 9-2 provides the minimum software versions for both Native IOS and Hybrid to support the FlexWAN module.

Table 9-2. Minimum Software Versions for FlexWAN Support

	Sup IA MSFCI with FlexWAN	Sup IA MSFCII with FlexWAN	Sup II MSFCII with FlexWAN
Native IOS	12.1(5a)E1	12.1(5a)E1	12.1(8a)E
Hybrid	CatOS: 5.4(2) Cisco IOS: 12.1(1)EX1 and 12.1(1)E	CatOS: 5.4(2) Cisco IOS: 12.1(2)E	CatOS: 6.1(1) Cisco IOS: 12.1(3a)E1

When selecting software to support the FlexWAN module, it is necessary to choose a version with a "v" listed in the feature set field. This is identical to the requirement for supporting a VIP on a 7500. If the intent is to load the Enterprise feature set with support for *Secure Shell* (SSH), Triple Data Encryption Standard (3DES), and the FlexWAN module, the feature set field would appear as jk2sv. This requirement applies to IOS versions released prior to 12.1(5a)E. For 12.1(5a)E and all subsequent releases, FlexWAN support is incorporated into each software version. The exception to this rule begins with IOS Release 12.1(13)E. Starting with this software release, the "LAN-only" feature set is introduced. This specific feature set does not incorporate support for the FlexWAN module. However, this exception only applies to Native IOS; the Hybrid Software version maintains support for the FlexWAN module.

The FlexWAN module supports numerous 7200/7500 WAN port adapters in the individual bays. This support allows for the integration of existing available hardware. However, the FlexWAN does not support LAN port adapters, double-wide port adapters, or service modules, such as the VPN accelerator module. Double-wide port adapters are not supported due to the availability of separate processors for each bay. Table 9-3 shows which 7200/7500 WAN port adapters are supported with the FlexWAN module.

Table 9-3. Supported 7200/7500 WAN Port Adapters

T1/E1	T3/E3	HSSI	ATM	Packet over SONET
PA-4T+	PA-T3	PA-H	PA-A3-T3	PA-POS-OC3MM
PA-8T-V35	PA-2T3	PA-2H	PA-A3-E3	PA-POS-OC2SMI
PA-8T-X21	PA-T3+		PA-A3-OC3MM	PA-POS-OC3SML
PA-8T-232	PA-2T3+		PA-A3-OC3SMI	
PA-MC-4T1	PA-E3		PA-A3-OC3SML	
PA-MC-8T1	PA-2E3		PA-T1-IMA	
PA-MC-8TE1+	PA-MC-T3			
PA-MC-8E1/120	PA-MC-2T3+			
PA-MC-STM-1	PA-MC-E3			

QoS Support on the MSFC and FlexWAN

The MSFC and FlexWAN offer a broad range of QoS mechanisms. The QoS support on the MSFC and FlexWAN are derived from the distributed QoS support for a VIP, available on the 7500. Therefore, the MSFC and FlexWAN provide intelligent distributed QoS services, which allow QoS policies to be extended across the MAN and WAN.

The QoS features available for the MSFC and FlexWAN are primarily configured using the *Modular QoS command-line interface* (MQC), introduced in chapter 5. The MQC enables the administrator to classify traffic, define policies, and apply those policies in a systematic modular fashion. For a more detailed discussion regarding the MQC, refer to Chapter 5.

When a WAN packet arrives or departs a FlexWAN interface, the FlexWAN is responsible for applying its own QoS mechanisms to that packet. When the packet is received on the WAN interface, any configured ingress QoS policies defined on the interface are applied to the packet. The same behavior applies to egress packets. When the forwarding decision has been made and the packet is encapsulated with the appropriate WAN header, any egress QoS policies defined on the interface are applied to the packet, which is then queued for transmission. This behavior is particularly true for policing functions on the FlexWAN. The FlexWAN is responsible for policing traffic streams on its interfaces, due to the potential discrepancies that may result from the variation in header length between a WAN and Ethernet frame. Although the PFC does not perform any policing or QoS services for the WAN traffic, it does switch WAN traffic at Layer 3. However, the PFC does not modify the Layer 3 *type of service* (ToS) field when processing these packets.

Aside from the FlexWAN performing its own set of QoS functions, there are other things to consider when deploying QoS on a 6500 with an MSFC and FlexWAN. When a frame is sent either to the MSFC or FlexWAN, the Layer 2 CoS settings applied to that frame are not maintained. By default, a 6500 equipped with a PFC I sets the CoS to zero if the frame is processed and forwarded from the MSFC or the FlexWAN. With a PFC II, CoS is derived from the precedence value in the IP header. This feature is not configurable. For IP and IPX packets, the MSFC is not normally involved in the forwarding process. Therefore, careful consideration must be given when applying QoS policies to the MSFC and FlexWAN so that the outcome results in the desired behavior. The QoS policies discussed in this chapter center on the FlexWAN module. Although the MSFC is required for configuring the FlexWAN module, it is not possible to run many of the mechanisms discussed without a FlexWAN module. The exception to this is *network-based application recognition* (NBAR). With Cisco IOS Release 12.1(13)E, software-assisted NBAR is supported without a FlexWAN module. At the time of this writing, hardware-assisted NBAR is not yet supported.

Classification

Classification is the first step in applying QoS policies within a network. If traffic is not classified, policies cannot be applied. Classification categorizes network traffic and assigns those categories to different classes of service. When the traffic is classified, QoS mechanisms are used to maintain the appropriate service levels for a particular category or class. Voice traffic, for example, is extremely vulnerable to delays in the network, and as a result requires expeditious handling on an end-to-end basis. Contrary to the voice traffic, HTTP or web-based traffic is not significantly impacted by delays or drops experienced in the network. Therefore, based on the diverse handling requirements, it is necessary to classify these types of traffic differently. When all traffic is assigned to the appropriate class, mechanisms, such as Low Latency Queuing (LLQ) for voice or *Class-Based Weighted Fair Queuing* (CBWFQ) and marking for web-based applications, are applied to accommodate the required service levels.

There are several ways to implement classification. One method is to classify all traffic traveling through a specific interface. However, this is primarily for situations where a homogenous mixture of traffic is present. For those instances, all traffic departing or leaving a particular interface should be provided the same service level. However, this type of classification policy is more the exception than the norm. If the previous policy is applied to an interface where there is a heterogeneous mixture of traffic, any benefits obtained from deploying QoS are negated. This QoS negation results because no distinctions are being made between the different assigned priority levels for the various traffic flows.

Another classification method is to use standard and extended access lists. *Access-control lists* (ACLs) match addressing information, protocol IDs, or Layer 4 port numbers. When configuring ACLs, however, prior knowledge of the actual applications and protocols operating on the network is necessary. Matching values previously specified in the ToS field of the IP header, either IP precedence or *differentiated services codepoint* (DSCP), is yet another classification method. This allows forwarding decisions and policies to be applied based on predetermined values assigned at either the access or distribution layers of the network.

One other classification mechanism implemented on the Catalyst 6500 for the FlexWAN and MSFC is *distributed network-based application recognition* (dNBAR). dNBAR, through the use of *packet description language modules* (PDLMs), recognizes and classifies a wide range of IP-based applications, as well as HTTP traffic found on networks. Not only does dNBAR recognize applications using static port assignments, it is capable of classifying applications that utilize dynamic port assignments, as well as classifying HTTP traffic based on subport characteristics. dNBAR was initially only supported on the Catalyst 6500 with a FlexWAN module with Software Release 12.1(6)E. However, Cisco IOS Software Release 12.1(13)E expanded NBAR support to include LAN interfaces and does not require a FlexWAN module. NBAR is supported only with the MSFC II. This section briefly describes NBAR and demonstrates how NBAR may be deployed on the Catalyst 6500.

> NOTE
>
> PDLMs provide the necessary information to the NBAR inspection process, allowing NBAR to recognize the various applications. PDLMs can be loaded into Flash, and do not require downtime for the system. As new PDLMs become available, they can be loaded on the Catalyst 6500 for additional protocol support. PDLMs are only available through Cisco.

For a current list of protocols supported by NBAR at the time of this writing, refer to the following document at Cisco.com:

"Cisco IOS Software Release 12.2T Network-Based Application Recognition"

Or download PDLMs directly from the following website:

www.cisco.com/cgi-bin/tablebuild.pl/pdlm

NBAR Protocol Discovery

The first step in being able to classify network traffic is to actually know what protocols and applications are running on the network. This knowledge enables administrators to prioritize business-critical information and applications over less-important applications. Unfortunately, to configure ACLs to classify network traffic you must have prior knowledge of the network applications, as well as their associated protocol or port numbers. One option for discovering the protocols currently traversing an interface within the network is using NBAR protocol discovery. NBAR is capable of recognizing any protocol included within the PDLM file. Protocol discovery is applied to the desired interface or group of interfaces using the following command at each intended interface:

```
ip nbar protocol-discovery
```

When protocol discovery is applied to the interface, statistics are gathered depicting the active protocols traversing the interface. To view the results of the protocol discovery process, use the following command:

```
show ip nbar protocol-discovery [interface type num]
```

Example 9-1 demonstrates the behavior of the NBAR protocol discovery process.

Example 9-1. Configuring and Verifying NBAR Protocol Discovery

```
MSFC#configure terminal
MSFC(config)#interface serial 3/1/0
MSFC(config-if)#ip nbar protocol-discovery
MSFC#show ip nbar protocol-discovery interface serial 3/0/0

 Serial3/0/0
                              Input                     Output
   Protocol                   Packet Count              Packet Count
                              Byte Count                Byte Count
                              30 second bit rate (bps)  30 second bit rate (bps)
   ------------------------   ------------------------  ------------------------
   fasttrack                  1142                      25636899
                              53674                     7691069400
                              0                         1988000
   secure-http                2227281                   32046128
                              104682300                 6409225600
                              27000                     1657000
   ssh                        209780                    22432288
                              9859648                   5608071500
                              0                         1449000
   realaudio                  19227675                  217434
                              4806918750                10219398
                              1242000                   2000
   ntp                        2990024                   6409226
                              140531614                 961383900
                              38000                     249000
```

```
   icmp                    36144                   102
                           2170488                 10608
                           0                       0
   eigrp                   19540                   9724
                           1242576                 620451
                           0                       0
   bgp                     812                     406
                           39788                   17864
                           0                       0
(text omitted)
   Total                   5702157                 140751114
                           268799486               34224484573
                           67000                   6585000
```

NBAR Classification

For low-speed serial connections, it is essential to ensure critical applications are given precedence to the available bandwidth. You can use NBAR protocol discovery to discover what applications are utilizing network resources, as well as roughly estimate the bandwidth consumption of those protocols. You can use this information to determine effective policies to sustain end-to-end service levels.

In the preceding show ip nbar protocol-discovery output, for example, NBAR recognizes Fasttrack as one of the applications utilizing considerable bandwidth on this connection. Fasttrack is a protocol used for peer-to-peer applications, such as Kazaa and Grokster. In this example, protocols matching these descriptions are not mission-critical and are deemed low-priority. To restrict the bandwidth used by the protocols matching these descriptions, the following example uses a policer. As a result, access to the available bandwidth is restricted for these applications. The policer is configured using the MQC and is verified using show policy-map interface {*type num*}.

Example 9-2. Configuring Distributed NBAR Classification and Verifying Configuration

```
MSFC#configure terminal

MSFC(config)#class-map match-all Fasttrack
```

```
MSFC(config-cmap)#match protocol fasttrack
MSFC(config)#policy-map Non-critical-apps
MSFC(config-pmap)#class Fasttrack
MSFC(config-pmap-c)#police 128000 1500 1500 conform-action set-prec-transmit 0
➥exceed-action drop
MSFC#show policy-map interface serial 3/0/0
 Serial3/0/0
  service-policy output: Non-critical-apps
    class-map: Fasttrack (match-all)
      190667 packets, 57200100 bytes
      30 second offered rate 1345000 bps, drop rate 10000 bps
      match: protocol fasttrack
      police:
        128000 bps, 1500 limit, 1500 extended limit
        conformed 18263 packets, 5478900 bytes; action: set-prec-transmit 0
        exceeded 5 packets, 1500 bytes; action: drop
        violated 76902 packets, 23070600 bytes; action: drop
        conformed 129000 bps, exceed 0 bps violate 548000 bps
    class-map: class-default (match-any)
      640676 packets, 143760844 bytes
      30 second offered rate 3387000 bps, drop rate 0 bps
      match: any
        640676 packets, 143760844 bytes
        30 second rate 3387000 bps
```

In the preceding configuration, NBAR classifies Fasttrack traffic. The match protocol statement enables the administrator to specify one of the protocols recognized by NBAR as match criteria. In the example, the fasttrack keyword is selected. Although it is not depicted, multiple protocols may be specified as match criteria. Also if multiple protocols are listed, the class map

may be configured to match all conditions, or any one of the conditions configured by selecting either the match-any or match-all keyword. The example demonstrates the match-all keyword. Within the policy map fasttrack, the MSFC polices traffic to limit the utilized bandwidth for these protocols. Additionally, the MSFC marks packets conforming to the policing contract with an IP precedence value of zero. This provides only Best Effort delivery for these packets and ensures the MSFC does not favor these packets over more mission-critical applications when competing for bandwidth on congested interfaces.

As demonstrated in the preceding example, NBAR is not an all-encompassing QoS mechanism. Rather, it is a classification tool used for classifying IP-based traffic. NBAR works in conjunction with other available QoS tools such as policing. Another mechanism NBAR operates with is class-based marking. The following section discusses class-based marking for the MSFC and FlexWAN module.

Marking

The purpose of marking is to assign different priority levels to various traffic flows. It allows downs devices to differentiate higher-priority traffic from lower-priority traffic and perform predefined pol based on assigned precedence values or specific bits set within the particular header. The FlexWAN is capable of marking traffic using class-based marking, *committed access rate* (CAR), or class-bas policing to identify various traffic streams. Class-based marking enables the administrator to speci precedence or DSCP values within the IP header, assign incoming packets to a local QoS group, or MPLS experimental bits. CAR and class-based policing can be configured to solely mark traffic mat the configured classification criteria. Instead of discarding violating packets, the packets are marke and forwarded. This section explains class-based marking, CAR, and class-based policing as marki mechanisms and how they are configured on the FlexWAN module.

Class-Based Marking

Class-based marking is a mechanism used to identify and mark various traffic flows in the network Different devices then use these set values to prioritize traffic when congestion is experienced in th network. The administrator assigns values to the various traffic flows, based on specific classificati criteria. When the individual flows or groups of flows are assigned the appropriate marking parame devices in the network are able to act on those packets based on their assigned marking. This actio downstream devices to differentiate among the various high- and low-priority protocols and applic and deterministically drop or forward packets to maintain defined service levels. When configuring based marking on the FlexWAN module, you have three possible marking options. The following co syntax shows the three available options:

```
set {{ip {dscp {dscp} | precedence {prec}}} | {qos-group {group#}} | {mpls experim
➥{exp}}}
```

One option is to assign a value to the ToS field. The administrator assigns either an IP precedence value within the IP header. The second option is to assign the traffic to a QoS group. This provides additional granularity beyond the 64 possible DSCP values. set qos-group can be used in network have a significant number of different classes of traffic, and can scale up to 100 different assigned Because QoS groups are assigned to ingress traffic, the FlexWAN module uses the QoS group value prioritize traffic for transmission. Finally, the FlexWAN module also supports set mpls experimen egress traffic. Often, packets are marked based on IP precedence or DSCP values. DSCP is the recommended alternative if all devices in the network support DSCP. DSCP provides substantially r granularity than IP precedence, permitting up to 64 different service levels to be defined. Example through Example 9-6 demonstrate and explain the various steps of configuring class-based marking FlexWAN module, including the following:

- Defining classes and grouping application by class (Example 9-3)
- Configuring policies based on essential and nonessential traffic (Example 9-4)
- Implementing the service policy statement (Example 9-5)
- Verifying that the configuration appears as intended (Example 9-6)

Example 9-3. Configuring Distributed Class-Based Marking on the FlexWA Module

```
MSFC#configure terminal
MSFC(config)#class-map match-any Non-essential
MSFC(config-cmap)#match protocol http
MSFC(config-cmap)#match protocol fasttrack
MSFC(config)#class-map match-any Low-Priority
MSFC(config-cmap)#match protocol smtp
MSFC(config-cmap)#match protocol secure-http
MSFC(config)#class-map match-any Business-essential
MSFC(config-cmap)#match protocol sqlnet
MSFC(config-cmap)#match protocol sqlserver
MSFC(config)#class-map match-any Video-preso
MSFC(config-cmap)#match protocol netshow
```

The first step in configuring class-based marking is to define the various classes and group the rela applications and protocols into those classes. In this example, the classes are defined based on the they have on business functions. Applications using the Fasttrack protocol and normal web-based t considered nonessential, and as a result are placed in the appropriate class. Mail traffic and secure traffic, although not considered business-critical, are placed in a higher category than the nonesser elements. Applications essential to the business are placed in an even higher category. Because da traffic is time-sensitive, it is critical that any database components receive preferential treatment o other less-important protocols and applications. Finally, a separate category has been configured f video applications. When the traffic has been classified, the policies are configured.

Example 9-4. Configuring Essential and Nonessential Traffic

```
MSFC#configure terminal
MSFC(config)#policy-map CB-Marking
MSFC(config-pmap)#class Non-essential
MSFC(config-pmap-c)#police 256000 1500 1500 conform-action set-dscp-transmit 0
➥exceed-action drop
MSFC(config-pmap-c)#class Low-Priority
MSFC(config-pmap-c)#set ip dscp 8
MSFC(config-pmap-c)#class Business-essential
MSFC(config-pmap-c)#set ip dscp 16
MSFC(config-pmap-c)#class Video-preso
MSFC(config-pmap-c)#set ip dscp 24
```

In addition to marking the "nonessential" traffic with DSCP 0, the traffic is also being policed to 25(Therefore, not only is the nonessential traffic dropped first during periods of congestion, the bandw also limited, allowing more availability to other protocols and applications, such as voice. When the are defined, the configured policy map is applied to the interface with the desired service-policy statement.

Example 9-5. Configuring the service-policy Statement

```
MSFC(config)#interface serial 3/0/0
MSFC(config-if)#service-policy input CB-Marking
MSFC(config-if)#end
```

After all the configuration steps have been taken, you can verify the configuration and performance show policy-map interface {*type num*}.

Example 9-6. Verifying the Configuration

```
MSFC#show policy-map interface serial 3/0/0
 Serial3/0/0
  service-policy input: CB-Marking
    class-map: Non-essential (match-any)
      198987 packets, 61685950 bytes
      30 second offered rate 1233000 bps, drop rate 925000 bps
      match: protocol http
        119392 packets, 41787200 bytes
        30 second rate 834000 bps
      match: protocol fasttrack
        79595 packets, 19898750 bytes
        30 second rate 395000 bps
      police:
        256000 bps, 1500 limit, 1500 extended limit
        conformed 49311 packets, 12694050 bytes; action: set-dscp-transmit 0
        exceeded 8 packets, 2600 bytes; action: drop
        violated 149668 packets, 48989300 bytes; action: drop
        conformed 252000 bps, exceed 0 bps violate 978000 bps
    class-map: Low-Priority (match-any)
      129343 packets, 35071825 bytes
      30 second offered rate 699000 bps, drop rate 0 bps
      match: protocol smtp
        39798 packets, 5969700 bytes
        30 second rate 118000 bps
      match: protocol secure-http
        89545 packets, 29102125 bytes
        30 second rate 580000 bps
      set:
```

```
    ip dscp 8
class-map: Business-essential (match-any)
  139292 packets, 23878575 bytes
  30 second offered rate 476000 bps, drop rate 0 bps
  match: protocol sqlnet
    79595 packets, 17908875 bytes
    30 second rate 355000 bps
  match: protocol sqlserver
    59697 packets, 5969700 bytes
    30 second rate 118000 bps
  set:
    ip dscp 16
class-map: Video-preso (match-any)
  159189 packets, 55716150 bytes
  30 second offered rate 1113000 bps, drop rate 0 bps
  match: protocol netshow
    159189 packets, 55716150 bytes
    30 second rate 1113000 bps
  set:
    ip dscp 24
class-map: class-default (match-any)
  176 packets, 11174 bytes
  30 second offered rate 0 bps, drop rate 0 bps
  match: any
    176 packets, 11174 bytes
    30 second rate 0 bps
```

Marking Using Committed Access Rate (CAR)

CAR is a legacy QoS mechanism and is not generally recommended for new deployments. For the s
completeness, an explanation of CAR's configuration and functionality is included in this chapter. A
CAR is primarily used as a policing mechanism, you can also use CAR to mark traffic. CAR is config
match traffic using an ACL, a pre-established DSCP value, a QoS group, or CAR matches all ingress
egress traffic traversing an interface, based on the direction the command is applied. The following
command applies CAR to the desired interface:

rate-limit {**input** | **output**} [[**access-group**[**rate-limit**]*list #*] | [**qos-group** *group#*

[**dscp***dscp*]] {*rate*} {*normal burst*} {*excess burst*}**conform-action** {*conform-action*}

exceed-action {*exceed-action*}

Therate-limit command is applied to a specific interface and configured in interface configuration
The required input or output option specifies the direction the rate-limit command is applied, rel
the traffic flow. The next set of options allows traffic to be matched against a predetermined list or
assigned value. The specified rate is measured in bits per seconds, and the burst values are measu
bytes. The conform-action and exceed-action determine what actions are taken for conforming
nonconforming packets, respectively. The following output displays the configurable actions. The k
shown are available for both conform-action and exceed-action.

Example 9-7. Configurable Options for the rate-limit Command

```
MSFC(config-if)#rate-limit input 1000000 187500 375000 conform-action ?
  continue           scan other rate limits
  drop               drop packet
  set-dscp-continue  set dscp, scan other rate limits
  set-dscp-transmit  set dscp and send it
  set-prec-continue  rewrite packet precedence, scan other rate limits
  set-prec-transmit  rewrite packet precedence and send it
  set-qos-continue   set qos-group, scan other rate limits
```

```
  set-qos-transmit   set qos-group and send it
  transmit           transmit packet
```

If the intent is to use CAR to mark packets, as opposed to police, the drop keyword is not used. In set action is specified to modify the ToS field in the IP header, or set the local QoS group value for packet. The transmit keyword is yet another option, which allows a packet to be forwarded withou modifying any existing settings. One additional feature with CAR is the flexibility to configure multi rate-limit statements on the same interface. By using the continue keyword, independently or wi set action, packets can be processed through multiple rate-limit statements until a match is found event a match is not found, the default action is to transmit. Therefore, in the absence of a match, packet is just forwarded. Example 9-8 demonstrates configuring CAR to mark traffic. In the examp extended ACL is configured specifying the traffic to be considered. In this instance, secure web traf being forwarded to the serial interface noted in the example. However, the traffic is not marked wi value that conforms to the QoS policy in place. The intent is to mark all secure web traffic conformi the configured 1-Mbps rate with DSCP 8. Any traffic exceeding this rate is marked with DSCP 0.

Example 9-8. Marking Secure Web Traffic with CAR

```
MSFC#configure terminal
MSFC(config)#access-list 101 permit tcp any any eq 443
MSFC(config)#interface serial 3/0/0
MSFC(config-if)#rate-limit input access-group 101 1000000 187500 375000 conform-ac
➥set-dscp-transmit 8 exceed-action set-dscp-transmit 0
MSFC(config-if)#end

MSFC#show interface serial 3/0/0 rate-limit
Serial3/0/0
  Input
    matches: access-group 101
      params:  1000000 bps, 187500 limit, 375000 extended limit
      conformed 115046 packets, 46018400 bytes; action: set-dscp-transmit 8
      exceeded 56927 packets, 22770800 bytes; action: set-dscp-transmit 0
      last packet: 4ms ago, current burst: 281100 bytes
      last cleared 00:06:12 ago, conformed 988000 bps, exceeded 489000 bps
```

The output verifies traffic conforming to the configured contract is marked with DSCP 8, whereas tr violating the contract is marked with DSCP 0. The following section demonstrates using the class-b policer to accomplish the same results provided in Example 9-8.

Marking Using a Class-Based Policer

Similar to CAR, but preferred for new deployments, the class-based policer marks traffic without er any policing actions. However, the class-based policer has three different actions it enforces on tra CAR, it has a conform- and an exceed-action, however the class-based policer also has a third acti violate. The violate-action applies an additional set of actions to traffic violating the configured rate exceeding the assigned conform and excess burst values. Class-based policer operation is discusse further detail in the "Policing and Shaping" section of this chapter.

Example 9-9. Configuring the Distributed Class-Based Policer for Marking

```
MSFC#configure terminal

MSFC(config)#class-map match-all Secure-Web

MSFC(config-cmap)#match protocol secure-http

MSFC(config)#policy-map Marking-policy

MSFC(config-pmap)#class Secure-Web

MSFC(config-pmap-c)#police 1000000 1500 1500 conform-action set-dscp-transmit 1
➥exceed-action set-dscp-transmit 0 violate-action set-dscp-transmit 0

MSFC(config)#interface serial 3/0/0

MSFC(config-if)#service-policy input Marking-policy

MSFC(config-if)#end

MSFC#show policy-map interface serial 3/0/0
 Serial3/0/0
  service-policy input: Marking-policy
    class-map: Secure-Web (match-all)
      89929 packets, 35971600 bytes
      30 second offered rate 1497000 bps, drop rate 0 bps
```

```
    match: protocol secure-http
    police:
      1000000 bps, 1500 limit, 1500 extended limit
      conformed 59733 packets, 23893200 bytes; action: set-dscp-transmit 1
      exceeded 41 packets, 16400 bytes; action: set-dscp-transmit 0
      violated 30155 packets, 12062000 bytes; action: set-dscp-transmit 0
      conformed 994000 bps, exceed 0 bps violate 500000 bps
  class-map: class-default (match-any)
    91 packets, 5764 bytes
    30 second offered rate 0 bps, drop rate 0 bps
    match: any
      91 packets, 5764 bytes
      30 second rate 0 bps
```

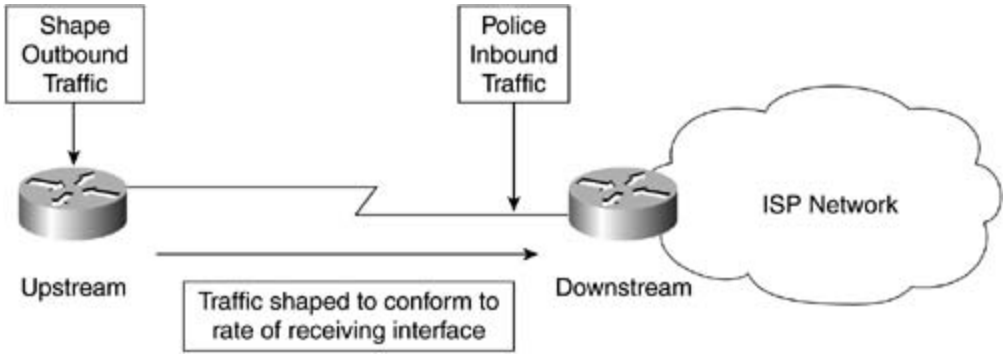
Similar to classification, marking is not a mechanism used independently. As demonstrated in one previous examples, NBAR can be a classification method used in conjunction with class-based mark However, the important aspect of marking is it also allows other devices in the network to be able differentiate between critical and noncritical traffic, based on the marking values. When the values determined, different policies can be applied to the traffic, including Priority Queuing, policing, or s The following section discusses policing and shaping on the FlexWAN module.

Policing and Shaping

Chapter 2 introduced policing and shaping. As discussed, both features use a token bucket mechan operation. Based on the token bucket analogy, when there are enough tokens available in the buck service an entire arriving packet, the packet is permitted to proceed. However, there is a difference how the policer and shaper function. The shaper adapts to bursty traffic. The shaper allocates a fin amount of buffer space to accommodate burstiness. This prevents traffic exceeding the average rat being dropped and allows packets to wait in a buffer until a sufficient amount of tokens are availab service the entire packet. The end result of the shaper is it smoothes traffic spikes down to the ave configured rate. However, although shaping utilizes buffers to prevent excessive drops, the bufferi introduces latency, which adversely impacts delay-sensitive applications. As mentioned in Chapter shaping is one way to prevent higher-speed interfaces from overrunning potentially lower-speed downstream interfaces. (The example used is in a Frame Relay environment.) Before deploying a s converged environment, however, you should fully understand the effects it has on traffic in the ne

Although policing uses the same token bucket scheme, policing does not buffer traffic. As a result, available tokens are exhausted, packets are dropped, instead of being buffered. It is also possible packets using a shaper. However, packets are only dropped when the buffer space is depleted. Wh deploying policers and shapers in the network, the recommendation is to shape on the upstream in and police on the downstream interface. A situation where this is practiced is when end customers a service provider. The end customer does not want the service provider to determine what traffic i be randomly dropped. Therefore, the customer configures a shaper to conform to the policing cont configured on the service provider's receiving interface. This permits the customer to prioritize and deterministically drop traffic during periods of congestion, to ensure all mission-critical traffic is no in the service provider's network. The FlexWAN module implements two types of policers: CAR and based policer. The "Marking" section of this chapter demonstrated how CAR and class-based polici be used to mark traffic. The following two sections discuss how these mechanisms are utilized as p This section concludes with deploying distributed traffic shaping on the FlexWAN.

Figure 9-1. Policing and Shaping in the Network



NOTE

This section does not provide an in-depth exhausted discussion on the various policing and sh techniques. It is meant to provide a fundamental overview regarding the operation of the vari

mechanisms, as well as provide examples and instances where the various functions can be applied. For a more thorough discussion regarding policing and shaping, refer to Chapter 2 or to the following document at Cisco.com:

"Policing and Shaping Overview"

Committed Access Rate (CAR)

Prior to class-based policing and the release of the Cisco MQC, CAR was the preferred method for p traffic. CAR does not allocate buffer space for queuing oversubscribed packets, nor does CAR guara minimum amounts of bandwidth for applications. The purpose of CAR, like most policers, is to limit the available bandwidth. This enables administrators, through designing specific policies, to ensure priority applications do not starve out mission-critical applications. This facilitates network traffic fl more deterministically through the network. The command syntax displayed previously in Example used to configure CAR as a policing mechanism:

```
rate-limit {input | output} [[access-group[rate-limit]list #] | [qos-groupgroup#]
[dscpdscp]] {rate} {normal burst} {excess burst}conform-action {conform-action}
exceed-action{exceed-action}
```

CAR works using a token bucket mechanism. The committed rate, measured in bits per second, det token arrival rate. As long as there are sufficient tokens available to service an entire packet, traffi through the policer. When the tokens are depleted, or there aren't enough available to accommoda entire packet, that packet and possibly subsequent packets are dropped. The token bucket depth is by the burst parameters. CAR defines a *committed burst* (Bc) and an *excess burst* (Be). When defi burst values, in order to use the excess burst capability, the Be value must be greater than Bc. If B Bc, excess burst is not defined, and there are no additional tokens available when the Bc bucket is Also it is not possible to configure the Be value to be less than Bc. In the event this does occur, Cis sets the Be equal to Bc. The burst values are measured in bytes.

Table 9-4 demonstrates CAR's operational behavior however; a few assumptions have been made. demonstration purposes, token replenishment does not occur. Also all packets in the table arrived same instance on the receiving interface. Packet size is fixed at 250 bytes per packet. Finally, the v the rate, committed burst, and excess burst are 8000 bits per second, 1500 bytes, and 3000 bytes respectively.

NOTE

For Table 9-4, the Be value is configured for 3000 bytes. When configuring the Be parameter fo CAR, it is necessary to consider the Bc value. The configured Be value is actually the sum of th and Be. Therefore, for Table 9-4, although the configured Be is 3000 bytes, only 1500 additior bytes are available for the Be bucket.

Table 9-4. CAR Operation (Actual and Compounded Debt)

Time	Packet Size	Actual Debt (ai)	Compounded Debt (Dc)	
0	0	0	0	<-- Committed burst bucket deple
1	250	250	250	
2	250	500	750	
3	250	750	1500	
4	250	1000	2500	<-- Packet dropped. (Dc exceeds Be.)
4		750	0	<-- Tokens not removed from exce bucket.
5	250	1000	1000	
6	250	1250	2250	<-- Packet dropped. (Dc exceeds Be.)
6		1000	0	
7	250	1250	1250	
8	250	1500	2750	<-- Packet dropped. (Dc exceeds Be.)
8		1250	0	
9	250	1500	1500	<-- Packet transmitted.
After Time 9, all subsequent packets experience a tail-drop scenario (ai > Dc), until the bucket is replenished with tokens.				

The table demonstrates the interaction between actual and compounded debt. As conforming packe processed by the CAR mechanism, the actual and compounded debt values accrue. When a packet

the compounded debt value to exceed the excess burst, that packet is dropped. Because the packe
discarded, however, no tokens are removed from the bucket and the compounded debt is reset to

Example 9-10. Configuring Distributed CAR for Ingress Traffic Flows

```
MSFC#configure terminal
MSFC(config)#access-list 110 remark Non-essential traffic
MSFC(config)#access-list 110 permit tcp any eq 1214 any
MSFC(config)#access-list 110 permit tcp any e 80 any
MSFC(config)#access-list 111 remark Low-priority traffic
MSFC(config)#access-list 111 permit tcp any eq 25 any
MSFC(config)#access-list 111 permit tcp any eq 443 any
MSFC(config)#access-list 112 remark Mission-Critical traffic
MSFC(config)#access-list 112 permit tcp any eq 1521 any
MSFC(config)#access-list 112 permit tcp any eq 1433 any
MSFC(config)#access-list 113 remark Video-applications
MSFC(config)#access-list 113 permit tcp any eq 1755 any
MSFC(config)#access-list 114 permit ip any any
MSFC(config)#interface serial 3/0/0
MSFC(config-if)#rate-limit input access-group 110 344000 65625 131250 conform-acti
➥set-dscp-transmit 0 exceed-action continue
MSFC(config-if)#rate-limit input access-group 111 400000 75000 150000 conform-acti
➥set-dscp-transmit 6 exceed-action continue
MSFC(config-if)#rate-limit input access-group 112 400000 75000 150000 conform-acti
➥set-dscp-transmit 16 exceed-action continue
MSFC(config-if)#rate-limit input access-group 113 1000000 187500 375000 conform-ac
➥set-dscp-transmit 26 exceed-action continue
MSFC(config-if)#rate-limit input access-group 114 4000000 750000 1500000 conform-a
➥drop exceed-action drop
```

```
MSFC(config-if)#end

MSFC#show interface serial 3/0/0 rate-limit
Serial3/0/0
  Input
    matches: access-group 110
      params:  344000 bps, 65625 limit, 131250 extended limit
      conformed 294270 packets, 80617800 bytes; action: set-dscp-transmit 0
      exceeded 641561 packets, 209489650 bytes; action: continue
      last packet: 4ms ago, current burst: 131010 bytes
      last cleared 00:31:56 ago, conformed 336000 bps, exceeded 874000 bps
    matches: access-group 111
      params:  400000 bps, 75000 limit, 150000 extended limit
      conformed 315610 packets, 93739250 bytes; action: set-dscp-transmit 6
      exceeded 529878 packets, 197663700 bytes; action: continue
      last packet: 4ms ago, current burst: 149850 bytes
      last cleared 00:31:56 ago, conformed 391000 bps, exceeded 825000 bps
    matches: access-group 112
      params:  400000 bps, 75000 limit, 150000 extended limit
      conformed 550546 packets, 93677725 bytes; action: set-dscp-transmit 16
      exceeded 104531 packets, 18620850 bytes; action: continue
      last packet: 1ms ago, current burst: 88350 bytes
      last cleared 00:31:56 ago, conformed 391000 bps, exceeded 77000 bps
    matches: access-group 113
      params:  1000000 bps, 187500 limit, 375000 extended limit
      conformed 658258 packets, 230390300 bytes; action: set-dscp-transmit 26
      exceeded 0 packets, 0 bytes; action: continue
      last packet: 4ms ago, current burst: 250 bytes
```

```
    last cleared 00:31:58 ago, conformed 960000 bps, exceeded 0 bps
  matches: access-group 114
    params:  4000000 bps, 750000 limit, 1500000 extended limit
    conformed 1276414 packets, 425802136 bytes; action: drop
    exceeded 0 packets, 0 bytes; action: drop
    last packet: 4ms ago, current burst: 400 bytes
    last cleared 00:31:58 ago, conformed 1775000 bps, exceeded 0 bps
```

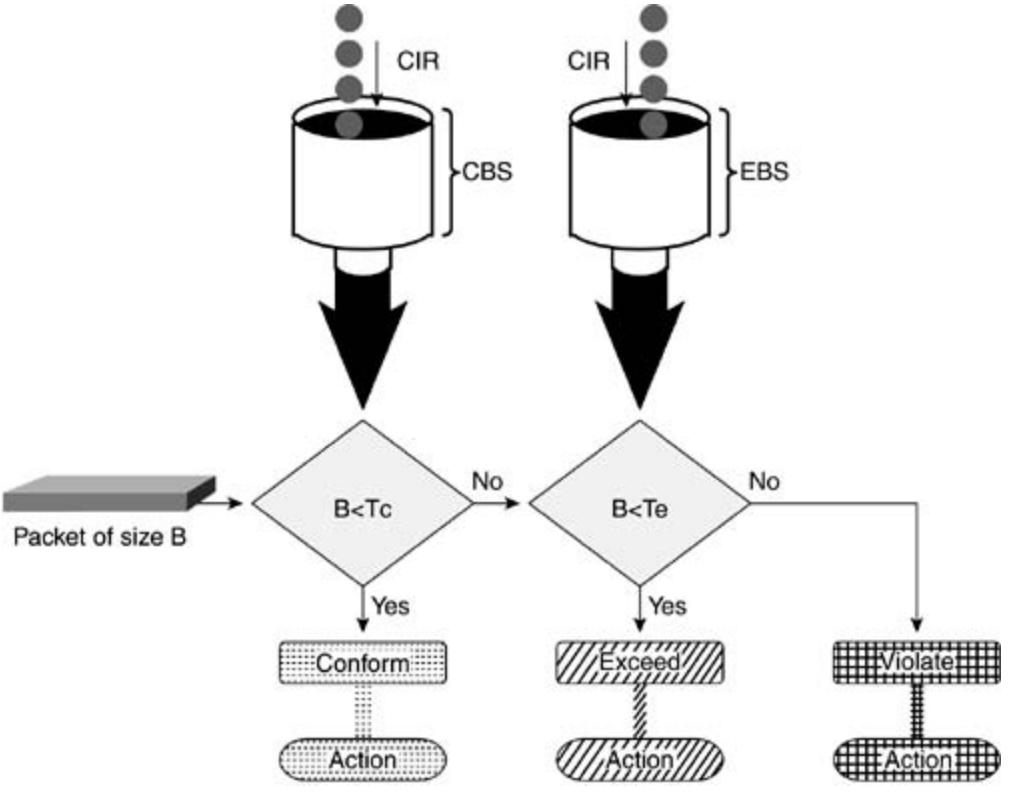
Class-Based Policer

The class-based policer is an alternative policing mechanism available for the FlexWAN module. Be
the development of the MQC, the class-based policer is the recommended policer for newer deploy
Similar to CAR, the class-based policer uses a token bucket mechanism to perform the policing acti
Compared to CAR, however, the class-based policer is more versatile and actually employs the use
token buckets rather than one. The two token buckets for the class-based policer represent the *con*
burst size (CBS) and the *excess burst size* (EBS). The committed information rate is responsible fo
how the rate tokens are replenished in both token buckets. As a result of using the additional toker
the class-based policer offers three possible policing actions. These three actions are modeled after
behavior described in RFC 2697, which discusses the single-rate three-color policer. The various av
actions—conform, exceed, and violate—represent the three different colors green, yellow, and red,
respectively.

The class-based policer is configured using the MQC. The policing parameters are configured under
policy map class using the following command:

```
police {rate} [normal burst] [excess burst]conform-action {conform-action} [exceed
➥{exceed-action}] [violate-action {violate-action}]
```

Figure 9-2. Class-Based Policer



The rate is measured in bits per second, and the optional burst parameters are measured in bytes. options are available for the conform-action; drop, transmit, set-dscp-transmit, set-prec-tran set-qos-transmit. These same options are available for the exceed-action and violate-action as w packets conforming to the configured committed information rate are forwarded based on the forw action. When a packet is forwarded, the number of tokens equal to the size of the packet is remove the Bc bucket. If a packet exceeds the committed information rate, meaning, there are not enough the Bc bucket to service the entire packet, the exceed-action is enforced. When a packet is forward on the exceed-action, the appropriate amount of tokens are depleted from the Be bucket. Finally, i exceeds the configured rate and there are not enough tokens available in the Be bucket to accomm entire packet, the packet violates the configured contract. As a result, the violate-action is enforced may result in marking down the packet's DSCP value or just dropping the violating packet. You car configure the class-based policer to emulate a single-bucket CAR implementation. By only configur conform-action and exceed-action, the class-based policer will behave based on a single-bucket po mechanism. If the violate-action is not specified in the configuration, the Be bucket is not configure therefore not used. Also specifying the drop keyword as the preferred action at any point results ir subsequent actions being configured to drop. Example 9-11 shows a class-based policer configurat how to verify the behavior.

Example 9-11. Configuring and Verifying Distributed Class-Based Policing

```
MSFC(config)#class-map match-any Non-essential

MSFC(config-cmap)#match protocol http

MSFC(config-cmap)#match protocol fasttrack
```

```
MSFC(config)#class-map match-any Low-Priority
MSFC(config-cmap)#match protocol smtp
MSFC(config-cmap)#match protocol secure-http
MSFC(config)#class-map match-any Business-essential
MSFC(config-cmap)#match protocol sqlnet
MSFC(config-cmap)#match protocol sqlserver
MSFC(config)#class-map match-any Video-preso
MSFC(config-cmap)#match protocol netshow
MSFC(config-cmap)#exit
MSFC(config)#policy-map CB-Policing
MSFC(config-pmap)#class Non-essential
MSFC(config-pmap-c)#police 344000 65625 65625 conform-action set-dscp-transmit 0
➥exceed-action drop
MSFC(config-pmap-c)#exit
MSFC(config-pmap)#class Low-Priority
MSFC(config-pmap-c)#police 400000 75000 75000 conform-action set-dscp-transmit 6
➥exceed-action set-dscp-transmit 0 violate-action drop
MSFC(config-pmap-c)#exit
MSFC(config-pmap)#class Business-essential
MSFC(config-pmap-c)#police 400000 75000 75000 conform-action set-dscp-transmit 16
➥exceed-action set-dscp-transmit 8 violate-action drop
MSFC(config-pmap-c)#exit
MSFC(config-pmap)#class Video-preso
MSFC(config-pmap-c)#police 1000000 187500 187500 conform-action set-dscp-transmit
➥exceed-action set-dscp-transmit 18 violate-action drop
MSFC(config-pmap-c)#exit
MSFC(config-pmap)#exit
MSFC(config)#interface serial 3/0/0
```

```
MSFC(config-if)#service-policy input CB-Policing
MSFC(config-if)#end

MSFC#show policy-map interface serial 3/0/0
 Serial3/0/0
 service-policy input: CB-Policing
   class-map: Non-essential (match-any)
     475684 packets, 147459900 bytes
     30 second offered rate 1230000 bps, drop rate 650000 bps
     match: protocol http
       285389 packets, 99886150 bytes
       30 second rate 833000 bps
     match: protocol fasttrack
       190295 packets, 47573750 bytes
       30 second rate 395000 bps
     police:
       344000 bps, 65625 limit, 65625 extended limit
       conformed 151166 packets, 41049400 bytes; action: set-dscp-transmit 0
       exceeded 405 packets, 130850 bytes; action: drop
       violated 324112 packets, 106279300 bytes; action: drop
       conformed 341000 bps, exceed 0 bps violate 887000 bps
   class-map: Low-Priority (match-any)
     394109 packets, 132133750 bytes
     30 second offered rate 707000 bps, drop rate 226000 bps
     match: protocol smtp
       95140 packets, 14271000 bytes
       30 second rate 117000 bps
     match: protocol secure-http
```

```
    298969 packets, 117862750 bytes
    30 second rate 587000 bps
  police:
    400000 bps, 75000 limit, 75000 extended limit
    conformed 178233 packets, 48220925 bytes; action: set-dscp-transmit 6
    exceeded 439 packets, 149825 bytes; action: set-dscp-transmit 0
    violated 218631 packets, 84629025 bytes; action: drop
    conformed 396000 bps, exceed 0 bps violate 309000 bps
class-map: Business-essential (match-any)
  332979 packets, 57081025 bytes
  30 second offered rate 475000 bps, drop rate 54000 bps
  match: protocol sqlnet
    190265 packets, 42809625 bytes
    30 second rate 356000 bps
  match: protocol sqlserver
    142714 packets, 14271400 bytes
    30 second rate 117000 bps
  police:
    400000 bps, 75000 limit, 75000 extended limit
    conformed 281382 packets, 48220450 bytes; action: set-dscp-transmit 16
    exceeded 802 packets, 149950 bytes; action: set-dscp-transmit 8
    violated 54235 packets, 9300375 bytes; action: drop
    conformed 396000 bps, exceed 0 bps violate 76000 bps
class-map: Video-preso (match-any)
  316766 packets, 110868100 bytes
  30 second offered rate 1107000 bps, drop rate 79000 bps
  match: protocol netshow
    316766 packets, 110868100 bytes
```

```
      30 second rate 1107000 bps
    police:
      1000000 bps, 187500 limit, 187500 extended limit
      conformed 316429 packets, 110750150 bytes; action: set-dscp-transmit 26
      exceeded 535 packets, 187250 bytes; action: set-dscp-transmit 18
      violated 3734 packets, 1306900 bytes; action: drop
      conformed 992000 bps, exceed 0 bps violate 112000 bps
  class-map: class-default (match-any)
    95643 packets, 19061418 bytes
    30 second offered rate 157000 bps, drop rate 0 bps
    match: any
      95643 packets, 19061418 bytes
      30 second rate 157000 bps
```

Distributed Traffic Shaping

Traffic shaping is a mechanism that regulates the amount of traffic leaving a particular interface. C
the policer, the traffic-shaping mechanism allocates buffers to accommodate traffic exceeding the c
information rate. Buffering allows the traffic shaper to tolerate short bursts in traffic, which are reg
and subsequently transmitted at the committed rate. Shaping is commonly found in Frame Relay
environments, or on interfaces peering to a service provider. Due to potential speed mismatches w
Frame Relay cloud, traffic shaping is implemented to ensure downstream interfaces are not overwh
with traffic. Traffic shaping prevents bottlenecks and congestion from occurring within the network
case of service providers, they provide their customers with specific service-level contracts. Freque
service providers strictly enforce these contracts by policing traffic transmitted toward the provider
Shaping can be applied in this instance to ensure network traffic conforms to the provider's policies
enables the end user to deterministically prioritize their traffic and ensure mission-critical applicati
not left to the discretion of the provider. The FlexWAN module supports *distributed traffic shaping* (
DTS is configured using the MQC and applied under the policy map class using the following comm

shape {**average** | **peak**} {*rate*} [*normal burst*] [*excess burst*]

Rate is expressed in bits per second and represents the average transmission rate for egress traffic burst (Bc) and excess burst (Be) are expressed in bits and represent the number of bits transmitte interval (Bc/rate). When the average keyword is specified, a total of Bc is transmitted per time int thepeak keyword is specified, (Bc + Be) is transmitted per time interval. When establishing burst increasing Bc increases the time between transmissions. This negatively impacts latency for time-s applications if the shaping mechanism is applied to the physical interface. When configuring shapir the MQC, if voice traffic is present, it is normally assigned to the LLQ. When assigned to the strict-p LLQ, voice traffic present in the queue is immediately serviced ahead of other traffic. Therefore, vo streams are not affected by shaping imposed on the other configured queues. If shaping is applied physical interface, however, Bc and its effect on network traffic must be carefully considered, partic voice traffic traverses the same interface.

In addition to the previous shape command, DTS provides mechanisms specific to Frame Relay environments. When congestion is experienced within a Frame Relay cloud, switches within the clo congestion notifications—*forward-explicit congestion notifications* (FECNs) and *backward-explicit co notifications* (BECNs)—to the end devices. These congestion notifications inform the devices along t transmission path that congestion has occurred and transmission rates should be throttled. The firs command listed instructs the receiving interface to send BECNs back to the transmitting device onc is received from the network. The second command specifies what transmission rate the interface s adjust to in the event congestion is detected. Both commands are configured using the MQC and ap under the policy map class:

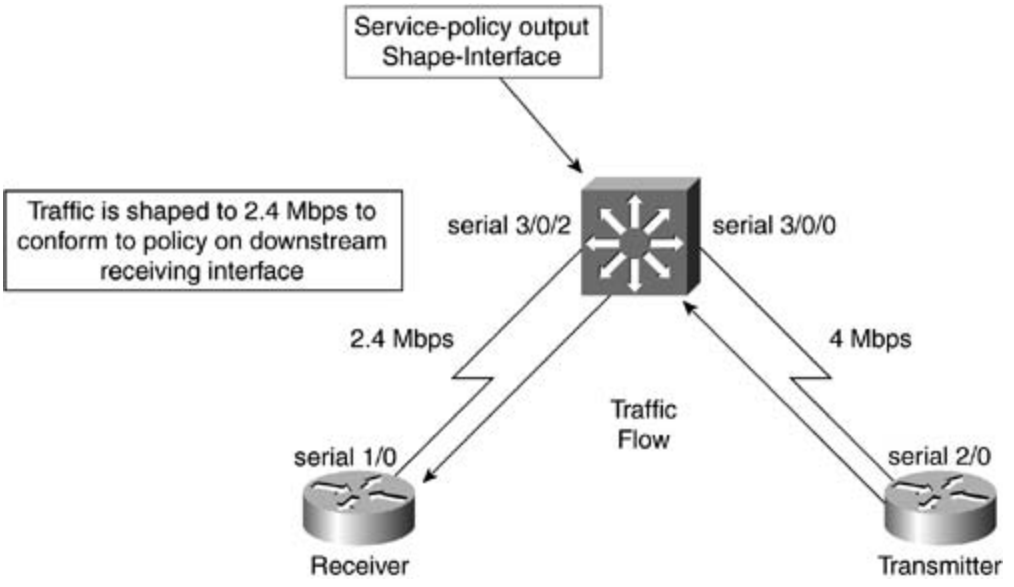
```
shape fecn-adapt

shape adaptive {rate}
```

NOTE

Frame Relay operation is beyond the scope of this book. For more information regarding Fram Relay, refer to the Frame Relay technology overview document at Cisco.com.

The following example demonstrates how to configure DTS on the FlexWAN module. In this examp traffic on serial 3/0/2 is shaped down to 2.4 Mbps. This example shows the upstream transmitting downstream receiving interfaces and the shaping mechanism operation. Figure 9-3 depicts how the devices are connected.

Figure 9-3. Distributed Traffic Shaping



Example 9-12. Configuring and Verifying Distributed Traffic Shaping

```
MSFC#configure terminal
MSFC(config)#access-list 150 permit ip any any
MSFC(config)#class-map All-traffic
MSFC(config-cmap)#match access-group 150
MSFC(config-cmap)#policy-map Shape-Interface
MSFC(config-pmap)#class All-traffic
MSFC(config-pmap-c)#shape average 24000000
MSFC(config)#interface serial 3/0/2
MSFC(config-if)#service-policy output Shape-Interface
MSFC(config-if)#end
Transmitting#show interfaces serial2/0
Serial2/0 is up, line protocol is up
  Hardware is M4T
  Internet address is 192.168.50.2/30
  MTU 1500 bytes, BW 4000 Kbit, DLY 20000 usec,
```

```
     reliability 255/255, txload 255/255, rxload 26/255
  Encapsulation HDLC, crc 16, loopback not set
  Keepalive set (10 sec)
  Last input 00:00:00, output 00:00:00, output hang never
  Last clearing of "show interface" counters 00:22:09
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  30 second input rate 53000 bits/sec, 152 packets/sec
  30 second output rate 2564000 bits/sec, 1320 packets/sec
     199706 packets input, 8838730 bytes, 0 no buffer
     Received 155 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     1709676 packets output, 417003428 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up

MSFC#show policy-map interface serial3/0/2
 Serial3/0/2
  service-policy output: Shape-Interface
    class-map: All-traffic (match-all)
      1374692 packets, 333637211 bytes
      30 second offered rate 2587000 bps, drop rate 51000 bps
      match: access-group 150
      queue size 134, queue limit 500
      packets output 1320791, packet drops 53898
      tail/random drops 53898, no buffer drops 0, other drops 0
      shape: cir 2400000,  Bc 9600,  Be 9600
```

```
        output bytes 312025836, shape rate 2419000 bps
      fair-queue: per-flow queue limit 125
    class-map: class-default (match-any)
      0 packets, 0 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      match: any
        0 packets, 0 bytes
        30 second rate 0 bps
```

```
Downstream#show interfaces serial1/0
Serial1/0 is up, line protocol is up
  Hardware is M4T
  Internet address is 192.168.60.2/30
  MTU 1500 bytes, BW 4000 Kbit, DLY 20000 usec,
     reliability 255/255, txload 3/255, rxload 152/255
  Encapsulation HDLC, crc 16, loopback not set
  Keepalive set (10 sec)
  Last input 00:00:00, output 00:00:00, output hang never
  Last clearing of "show interface" counters 00:07:35
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 39/75, 0 drops
  30 second input rate 2397000 bits/sec, 1269 packets/sec
  30 second output rate 55000 bits/sec, 154 packets/sec
     577710 packets input, 136495374 bytes, 0 no buffer
     Received 53 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     69483 packets output, 3082383 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
```

```
0 output buffer failures, 0 output buffers swapped out
0 carrier transitions     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

Congestion Management and Scheduling

Discussed in Chapter 2, congestion management is a technique used to manage traffic flows at poi
congestion within the network. The congestion management process involves the following three st

1. Creating different queues to accommodate network traffic
2. Assigning packets to the various queues based on predefined characteristics
3. Scheduling between the various queues to provide queued packets access to the available ba
when congestion has occurred on the link

This section discusses the various congestion management mechanisms supported on the FlexWAN
They include *distributed Weighted Fair Queuing* (dWFQ), *distributed Class-Based Weighted Fair Que*
(dCBWFQ), and *distributed Low Latency Queuing* (dLLQ). Examples are provided in the section
demonstrating the necessary configuration steps, as well as show commands to verify functionality

Distributed Weighted Fair Queuing

Distributed Weighted Fair Queuing (dWFQ) is a congestion management mechanism that operates
FlexWAN's VIPs and provides fair treatment for outbound flows on congested interfaces. WFQ is en
default on all interfaces operating at 2.048 Mbps and less. WFQ protects low-bandwidth traffic flow
ensuring high volume conversations do not monopolize the available bandwidth. The mechanism w
assigning each traffic flow to its own queue. A flow is defined as packets possessing the same sour
address, destination IP address, source TCP or UDP port, destination TCP or UDP port, protocol ID,
of service value. Once a packet is assigned to a queue, the WFQ mechanism services each queue e
each queue receives a fair share of the available bandwidth, based on the assigned weight for that

WFQ uses packet size and arrival time in conjunction with a weighting factor to permit access to th
bandwidth. WFQ is QoS aware. If a packet arrives with a high precedence value, the weight assign
packet will be low. Therefore, the higher the precedence value the lower the weight. The lower wei
translates to faster de-queuing and transmission time, which translates to more access to the avail
bandwidth. dWFQ for a VIP or FlexWAN is configured on the physical WAN interface. dWFQ operate
same as WFQ, with the exception weight is not assigned to a packet when executed in a distributec
on one of the VIPs of the FlexWAN module. All traffic flows are provided equal access to the availab
bandwidth. However, low bandwidth flows are still protected from being starved by high bandwidth
conversations. The following commands configure flow-based dWFQ on the target interface. The su
show commands allow the administrator to verify the configuration and monitor the statistics for t
applicable interface.

fair-queue

```
fair-queue [[aggregate-limit {# packets}] | [individual-limit {# packets}]]

show queueing fair [interface {type num}]

show interface {type num}
```

The command fair-queue enables flow-based dWFQ for the configured interface. After dWFQ is en the interface, the optional aggregate-limit keyword in conjunction with the fair-queue command to specify the maximum number of packets that may be present between all queues. Exceeding thi results in enforcement of the individual queue limits and possible discard for subsequent arriving p needing to be queued. The optional individual-limit keyword specifies the maximum number of p that can be queued for an individual flow queue. If the number of packets queued for a per flow qu increases above the number specified by the individual-limit keyword, all subsequent arriving pa dropped. Despite the enforcement of the aggregate and individual limits, packets already placed in queue are not discarded. It is not recommended to alter the aggregate-limit or individual-limit from their default settings. Before deciding to change the values from their defaults, carefully cons these changes will impact network and traffic performance.

dWFQ's benefit is its relative ease to implement. dWFQ functions independently of access-lists or th Although WFQ provides fair access to the available bandwidth, it does not provide specific bandwid guarantees for the various queues or classes during periods of congestion. The next section describ based weighted fair queuing and its benefits.

Distributed Class-Based Weighted Fair Queuing

CBWFQ is the most widely recognized QoS mechanism deployed today. CBWFQ enhances the funct available with WFQ.

CBWFQ provides scalable fair treatment of traffic on a class-by-class basis. However, CBWFQ also a minimum bandwidth guarantees to classes of traffic, to ensure bandwidth is available for critical applications. The minimum bandwidth assigned to a class is expressed as a percentage or a rate, e in kilobits per second. This number guarantees a class a minimum level of bandwidth in the event congestion is experienced on the interface, and assigns a weight to all packets matching the criteri given class. This weight denotes the frequency a class is serviced during periods of congestion, ens fair treatment of all traffic according to the configured policy. Because the weight assigned to a cla percentage based on the total available bandwidth for the assigned interface, it is essential to ensu bandwidth statement for the interface properly depicts what is available. Any remaining bandwidth assigned to one of the configured classes is allocated to the default class. By default, the sum of all bandwidth assigned to the various classes cannot exceed 75 percent of the total available bandwid provisioned for an interface. The remaining 25 percent provides minimum bandwidth guarantees fo Effort, overhead, and network control traffic. Use the following command to assign a minimum ban a class:

bandwidth {{*rate*} | **percent** {*percentage*}}

CAUTION

You can alter the 75 percent maximum bandwidth guarantee allocated for all configured classe using the interface command max-reserved-bandwidth [*percent*]. However, before modifyi this value, careful consideration must be given to ensure the configured classes do not consun the available bandwidth, starving overhead and critical network control traffic.

CBWFQ is configured using the MQC. The minimum bandwidth guarantee assigned to the appropria is configured under the policy-map class. For all administratively defined classes, a minimum band must be assigned to the applicable class before fair queuing is enabled. The exception to this beha default class. A minimum bandwidth guarantee is not required to configure flow-based WFQ for the class. Fair-queuing is configured for a specific class using the command fair-queue [queue-limit #}]. The queue-limit keyword associated with the fair-queue command specifies the maximum r possible per-flow queues for the configured class. Optionally, the command queue-limit {*packets*} configured under the policy-map class, specifies the maximum number of packets that can be queu the associated class. If the number of packets queued for the class increases above the number spe thequeue-limit command, all subsequent packets are tail-dropped. Example 9-9 demonstrates co flow-based WFQ for the default class called "class-default." The default class matches any packets matched by the classification criteria specified in the configured class-maps. Configuring flow-base for the default class allows queues within the class to fairly share the available bandwidth.

Unlike policing and shaping, the minimum bandwidth guarantees configured for CBWFQ do not rep maximum upper-bound limit. If a particular class is not transmitting, or fully utilizing the minimum allocated bandwidth, that bandwidth is available to the other classes, allowing them to transmit ab minimum guarantees. Example 9-13 demonstrates configuring and verifying CBWFQ behavior.

Example 9-13. Configuring Distributed WFQ

```
MSFC#configure terminal

MSFC(config)#class-map match-any Low-Priority

MSFC(config-cmap)#match protocol smtp

MSFC(config-cmap)#match protocol secure-http

MSFC(config)#class-map match-any Business-essential

MSFC(config-cmap)#match protocol sqlnet

MSFC(config-cmap)#match protocol sqlserver

MSFC(config)#class-map match-any Video-preso
```

```
MSFC(config-cmap)#match protocol netshow
MSFC(config-cmap)#exit
MSFC(config)#policy-map dCBWFQ
MSFC(config-pmap)#class Low-Priority
MSFC(config-pmap-c)#bandwidth 400
MSFC(config-pmap-c)#fair-queue
MSFC(config-pmap-c)#exit
MSFC(config-pmap)#class Business-essential
MSFC(config-pmap-c)#bandwidth 400
MSFC(config-pmap-c)#fair-queue
MSFC(config-pmap-c)#class Video-preso
MSFC(config-pmap-c)#bandwidth 1000
MSFC(config-pmap-c)#fair-queue
MSFC(config-pmap)#class class-default
MSFC(config-pmap-c)#fair-queue
MSFC(config-pmap-c)#exit
MSFC(config-pmap)#exit
MSFC(config)#interface serial 3/0/2
MSFC(config-if)#service-policy output dCBWFQ
MSFC(config-if)#end

MSFC#show interfaces serial 3/0/2
Serial3/0/2 is up, line protocol is up
  Hardware is Serial
  Internet address is 192.168.60.1/30
  MTU 1500 bytes, BW 4000 Kbit, DLY 20000 usec,
     reliability 255/255, txload 165/255, rxload 1/255
  Encapsulation HDLC, crc 16, loopback not set
```

```
  Keepalive set (10 sec)
  Last input 00:00:03, output 00:00:01, output hang never
  Last clearing of "show interface" counters 11:15:58
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: VIP-based fair queuing
  Output queue :0/40 (size/max)
  30 second input rate 1000 bits/sec, 3 packets/sec
  30 second output rate 2653000 bits/sec, 1271 packets/sec
     94583 packets input, 5754020 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     44615858 packets output, 2920117041 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
     RTS up, CTS up, DTR up, DCD up, DSR up

MSFC#show policy-map interface serial 3/0/2
 Serial3/0/2
  service-policy output: dCBWFQ
    class-map: Low-Priority (match-any)
      104897 packets, 26826925 bytes
      30 second offered rate 374000 bps, drop rate 0 bps
      match: protocol smtp
        41512 packets, 6226800 bytes
        30 second rate 97000 bps
      match: protocol secure-http
        63385 packets, 20600125 bytes
```

```
    30 second rate 274000 bps
  queue size 0, queue limit 100
  packets output 105295, packet drops 0
  tail/random drops 0, no buffer drops 0, other drops 0
  bandwidth: kbps 400, weight 10
  fair-queue: per-flow queue limit 25
class-map: Business-essential (match-any)
  147389 packets, 26827900 bytes
  30 second offered rate 374000 bps, drop rate 0 bps
  match: protocol sqlnet
    96712 packets, 21760200 bytes
    30 second rate 278000 bps
  match: protocol sqlserver
    50677 packets, 5067700 bytes
    30 second rate 94000 bps
  queue size 0, queue limit 100
  packets output 147934, packet drops 0
  tail/random drops 0, no buffer drops 0, other drops 0
  bandwidth: kbps 400, weight 10
  fair-queue: per-flow queue limit 25
class-map: Video-preso (match-any)
  191646 packets, 67076100 bytes
  30 second offered rate 937000 bps, drop rate 0 bps
  match: protocol netshow
    191646 packets, 67076100 bytes
    30 second rate 937000 bps
  queue size 0, queue limit 250
  packets output 192407, packet drops 0
```

```
      tail/random drops 0, no buffer drops 0, other drops 0
      bandwidth: kbps 1000, weight 25
      fair-queue: per-flow queue limit 62
    class-map: class-default (match-any)
      138172 packets, 33814224 bytes
      30 second offered rate 472000 bps, drop rate 0 bps
      match: any
        138172 packets, 33814224 bytes
        30 second rate 472000 bps
      queue size 0, queue limit 550
      packets output 138803, packet drops 0
      tail/random drops 0, no buffer drops 0, other drops 0
      fair-queue: per-flow queue limit 137
MSFC#show queueing fair interface serial 3/0/2
Current fair queue configuration:
 Serial3/0/2 queue size 0
        pkts output 1002701, wfq drops 0, nobuffer drops 0
 WFQ: aggregate queue limit 1000 max available buffers 1000
     Class 0: weight 55 limit 550 qsize 0 pkts output 235046 drops 0
     Class 2: weight 25 limit 250 qsize 0 pkts output 326801 drops 0
     Class 8: weight 10 limit 100 qsize 0 pkts output 180851 drops 0
     Class 11: weight 10 limit 100 qsize 0 pkts output 260001 drops 0
```

NOTE

The FlexWAN module supports configuring dCBWFQ on a per-VC basis. This is supported for *available bit rate* (ABR) and *variable bit rate* (VBR) classes of service. *Unspecified bit rate* (UBI and UBR+ do not provided bandwidth guarantees. dCBWFQ are configured using the MQC. As result, service policies are applied to individual *virtual circuits* (VCs) and individual members c VC bundle. To implement flow-based WFQ on a per-VC basis, the default class is configured fo

fair-queue under the policy map. So long as traffic is not matched by other classification crite the traffic defaults to the default class where flow-based WFQ is applied. For additional information on IP-to-ATM CoS, refer to the following technical document available at Cisco.co

"Configuring IP to ATM Class of Service"

Distributed Low Latency Queuing

Although WFQ protects low-bandwidth traffic, such as voice, from being starved of network resourc periods of congestion, WFQ cannot guarantee consistent delay variations between packets. Althoug can affect overall voice quality, as long as the delay is consistent the end user might not notice a d in service. However, variation in arrival time between voice packets, referred to as *jitter*, can quick degrade the performance of voice quality. dLLQ is specifically targeted for voice traffic. LLQ assigns priority queue for voice traffic. The benefit of the priority queue is that instead of waiting for the sc to service the various other queues, the strict-priority queue is provided immediate access to the transmission media. Traffic in the strict-priority queue is immediately forwarded regardless of cong the interface. When a packet is placed in the strict-priority queue, the scheduler immediately servic packet.Example 9-14 demonstrates configuring dLLQ on the FlexWAN module.

Example 9-14. Configuring and Verifying dLLQ

```
MSFC#configure terminal
MSFC(config)#ip access-list extended Voice-Control
MSFC(config-ext-nacl)#remark Permit Voice Control Traffic
MSFC(config-ext-nacl)#permit tcp 192.168.20.0 0.0.0.255 host 192.168.60.2 eq 2000
MSFC(config-ext-nacl)#permit tcp 192.168.40.0 0.0.0.255 host 192.168.60.2 eq 2748
MSFC(config-ext-nacl)#permit udp 192.168.20.0 0.0.0.255 host 192.168.60.2 eq tftp
MSFC(config)#class-map Voice-Control
MSFC(config-cmap)#match access-group name Voice-Control
MSFC(config)#ip access-list extended Voice-traffic
MSFC(config-ext-nacl)#remark Permit Phones to Phone Communication
MSFC(config-ext-nacl)#permit udp 192.168.20.0 0.0.0.255 192.168.70.0 0.0.0.255 ran
➥2767
MSFC(config-ext-nacl)#permit udp 192.168.40.0 0.0.0.255 192.168.70.0 0.0.0.255 ran
➥2767
MSFC(config)#class-map Voice-traffic
```

```
MSFC(config-cmap)#match access-group name Voice-traffic
MSFC(config)#access-list 101 permit tcp any any eq 443
MSFC(config)#class-map match-all Secure-HTTP
MSFC(config-cmap)#match access-group 101
MSFC(config-cmap)#class-map match-any Business-essential
MSFC(config-cmap)#match protocol sqlnet
MSFC(config-cmap)#match protocol sqlserver
MSFC(config)#policy-map LLQ-Policy
MSFC(config-pmap)#class Voice-Control
MSFC(config-pmap-c)#set ip dscp 26
MSFC(config-pmap-c)#bandwidth percent 20
MSFC(config-pmap-c)#class Business-essential
MSFC(config-pmap-c)#bandwidth percent 20
MSFC(config-pmap-c)#class Secure-HTTP
MSFC(config-pmap-c)#bandwidth percent 20
MSFC(config-pmap-c)#class Voice-traffic
MSFC(config-pmap-c)#set ip dscp ef
MSFC(config-pmap-c)#priority 500 1500
MSFC(config)#interface serial 3/0/2
MSFC(config-if)#service-policy output LLQ-Policy
MSFC#(config)end

MSFC#show policy-map interface serial 3/0/2
 Serial3/0/2
  service-policy output: LLQ-Policy
    queue stats for all priority classes:
      queue size 0, queue limit 250
      packets output 71195, packet drops 0
```

```
  tail/random drops 0, no buffer drops 0, other drops 0
class-map: Business-essential (match-any)
  124590 packets, 21358250 bytes
  30 second offered rate 452000 bps, drop rate 0 bps
  match: protocol sqlnet
    71194 packets, 16018650 bytes
    30 second rate 339000 bps
  match: protocol sqlserver
    53396 packets, 5339600 bytes
    30 second rate 113000 bps
  queue size 0, queue limit 50
  packets output 124591, packet drops 0
  tail/random drops 0, no buffer drops 0, other drops 0
  bandwidth: 20%, kbps 400
class-map: Voice-Control (match-any)
  71195 packets, 6229550 bytes
  30 second offered rate 131000 bps, drop rate 0 bps
  match: access-group 120
    71195 packets, 6229550 bytes
    30 second rate 131000 bps
  queue size 0, queue limit 50
  packets output 73149, packet drops 0
  tail/random drops 0, no buffer drops 0, other drops 0
  bandwidth: 20%, kbps 400
  set:
    ip dscp 26
class-map: Secure-HTTP (match-all)
  167307 packets, 66922800 bytes
```

```
  30 second offered rate 1419000 bps, drop rate 93000 bps
  match: access-group 101
  queue size 48, queue limit 50
  packets output 109971, packet drops 61929
  tail/random drops 61929, no buffer drops 0, other drops 0
  bandwidth: 20%, kbps 400
class-map: Voice-traffic (match-any)
  71195 packets, 14239000 bytes
  30 second offered rate 301000 bps, drop rate 0 bps
  match: access-group 121
    71195 packets, 14239000 bytes
    30 second rate 301000 bps
  Priority: kbps 500, burst bytes 1500, b/w exceed drops: 0
  set:
    ip dscp 46
class-map: class-default (match-any)
  77 packets, 4928 bytes
  30 second offered rate 0 bps, drop rate 0 bps
  match: any
    77 packets, 4928 bytes
    30 second rate 0 bps
  queue size 0, queue limit 100
  packets output 121, packet drops 0
  tail/random drops 0, no buffer drops 0, other drops 0
```

Congestion Avoidance

The purpose of congestion avoidance is to avoid congestion from occurring at bottlenecks within th network. This avoidance is accomplished by proactively managing transmit queues. Congestion avo mechanisms are specifically targeted for TCP-based applications. TCP uses flow-control mechanism manage established communication sessions. As a result, drops in the network impact TCP session are indicators of congestion. Unfortunately, UDP traffic does not employ any flow-control mechanis the protocol level; therefore traffic is not throttled as a result of a dropped packet.

The FlexWAN module employs *distributed Weighted Random Early Detection* (dWRED) as a congest avoidance mechanism for port adapter interfaces. The mechanism's weighting factor enables the administrator to determine the probability a packet is dropped based on its assigned IP precedence DSCP value. The following section discusses dWRED operation on the FlexWAN module.

Distributed Weighted Random Early Detection

dWRED attempts to alleviate congestion within a network. dWRED accomplishes this by proactively monitoring transmit queues. As traffic accumulates in the queues, dWRED randomly discards packe prevent congestion from occurring. dWRED monitors the average length of a transmit queue. Withi queue are established minimum and maximum thresholds associated with the various IP precedenc DSCP values. As long as the average queue length for a queue remains below the minimum WRED threshold, the packet is queued for transmission. If the average queue length exceeds the minimur threshold, however, the packet is potentially discarded. Whether the packet is discarded depends o precedence or DSCP value for the packet. The higher the assigned IP precedence or DSCP value, th the chances for packet discard. When the packet exceeds both the minimum and maximum thresho packets are dropped. The following commands configure dWRED for a particular class using the MC

```
random-detect {dscp-based | precedence-based}

random-detect exponential-weighting-constant {exponent#}

random-detect {{dscp {dscp}} | {precedence{prec}}} {min thr} {max thr} {mark proba
```

The first random-detect command specifies whether random-detect uses IP precedence or DSCP Note that when the DSCP-based or precedence-based method is chosen, all configured classes mus same method. The exponential weighting factor is a configurable value and affects how WRED calcu the average queue size. The higher the weighting values, the less susceptible WRED is to variations queue size. As a result, if the value is set too high, WRED may react slower to signs of congestion, to police the queue. For smaller weighting values, WRED responds to changes in the queue size. If weighting value is set too low, however, packets may be dropped too frequently. It is not recomme alter the exponential weighting factor, because of the dramatic impact it may have on performance

the DSCP or IP precedence values are mapped to a minimum and maximum threshold, measured i packets. In addition, each precedence value or DSCP value is assigned a mark probability. The *ma probability* is a denominator that specifies how many packets are dropped when the maximum thre attained. If the value is 25, for example, at the maximum threshold 1 out of every 25 packets is dr As the average queue size exceeds the minimum threshold, the drop rate is linear with respect to t probability. As a result, as the queue size increases, the probability linearly increases for packets t discarded. This occurs until the maximum threshold is exceeded and all packets mapped to that th are dropped. Example 9-15 demonstrates configuring dWRED and verifying the behavior.

Example 9-15. Configuring dWRED on the FlexWAN Module

```
MSFC#configure terminal
MSFC(config)#class-map match-any Low-Priority
MSFC(config-cmap)#match protocol smtp
MSFC(config-cmap)#match protocol secure-http
MSFC(config)#class-map match-any Business-essential
MSFC(config-cmap)#match protocol sqlnet
MSFC(config-cmap)#match protocol sqlserver
MSFC(config)#class-map match-any Video-preso
MSFC(config-cmap)#match protocol netshow
MSFC(config-cmap)#exit
MSFC(config)#policy-map dCBWFQ
MSFC(config-pmap)#class Low-Priority
MSFC(config-pmap-c)#bandwidth 400
MSFC(config-pmap-c)#fair-queue
MSFC(config-pmap-c)#random-detect dscp-based
MSFC(config-pmap-c)#exit
MSFC(config-pmap)#class Business-essential
MSFC(config-pmap-c)#bandwidth 400
MSFC(config-pmap-c)#fair-queue
MSFC(config-pmap-c)#random-detect dscp-based
MSFC(config-pmap-c)#class Video-preso
```

```
MSFC(config-pmap-c)#bandwidth 1000
MSFC(config-pmap-c)#fair-queue
MSFC(config-pmap-c)#random-detect dscp-based
MSFC(config-pmap)#class class-default
MSFC(config-pmap-c)#fair-queue
MSFC(config-pmap-c)#random-detect dscp-based
MSFC(config-pmap-c)#random-detect dscp 0 20 100 25
MSFC(config-pmap-c)#random-detect dscp 8 75 250 100
MSFC(config-pmap-c)#exit
MSFC(config-pmap)#exit
MSFC(config)#interface serial 3/0/2
MSFC(config-if)#service-policy output dCBWFQ
MSFC(config-if)#end

MSFC#show policy-map interface serial 3/0/2
 Serial3/0/2
  service-policy output: dCBWFQ
    class-map: Low-Priority (match-any)
      2547111 packets, 645072400 bytes
      30 second offered rate 370000 bps, drop rate 0 bps
      match: protocol smtp
        1044221 packets, 156633150 bytes
        30 second rate 69000 bps
      match: protocol secure-http
        1502890 packets, 488439250 bytes
        30 second rate 298000 bps
      queue size 0, queue limit 100
      packets output 2559341, packet drops 0
```

```
  tail/random drops 0, no buffer drops 0, other drops 0
  bandwidth: kbps 400, weight 10
  fair-queue: per-flow queue limit 25
  random-detect:
    Exp-weight-constant: 9 (1/512)
    Mean queue depth: 0
    Class Random       Tail   Minimum   Maximum     Mark       Output
            drop       drop threshold threshold  probability  packets
    0          0          0        25        50     1/10       117590
class-map: Video-preso (match-any)
  4607686 packets, 1612690100 bytes
  30 second offered rate 923000 bps, drop rate 0 bps
  match: protocol netshow
    4607687 packets, 1612690450 bytes
    30 second rate 923000 bps
  queue size 0, queue limit 250
  packets output 4629910, packet drops 0
  tail/random drops 0, no buffer drops 0, other drops 0
  bandwidth: kbps 1000, weight 25
  fair-queue: per-flow queue limit 62
  random-detect:
    Exp-weight-constant: 9 (1/512)
    Mean queue depth: 0
    Class Random       Tail   Minimum   Maximum     Mark       Output
            drop       drop threshold threshold  probability  packets
    26         0          0       108       125     1/10       130443
class-map: Business-essential (match-any)
  2375525 packets, 408280000 bytes
```

```
  30 second offered rate 370000 bps, drop rate 0 bps
  match: protocol sqlnet
    1365820 packets, 307309500 bytes
    30 second rate 277000 bps
  match: protocol sqlserver
    1009705 packets, 100970500 bytes
    30 second rate 92000 bps
  queue size 0, queue limit 350
  packets output 2389714, packet drops 0
  tail/random drops 0, no buffer drops 0, other drops 0
  queue-limit 350
  bandwidth: kbps 400, weight 10
  fair-queue: per-flow queue limit 87
  random-detect:
    Exp-weight-constant: 9 (1/512)
    Mean queue depth: 0
    Class Random       Tail   Minimum   Maximum     Mark       Output
            drop       drop threshold threshold  probability  packets
    16         0          0        31        50     1/10       116425
class-map: class-default (match-any)
  3304694 packets, 813059382 bytes
  30 second offered rate 464000 bps, drop rate 0 bps
  match: any
    3304694 packets, 813059382 bytes
    30 second rate 464000 bps
  queue size 0, queue limit 550
  packets output 3325430, packet drops 0
  tail/random drops 0, no buffer drops 0, other drops 0
```

```
fair-queue: per-flow queue limit 137

random-detect:

  Exp-weight-constant: 9 (1/512)

  Mean queue depth: 0

  Class Random       Tail   Minimum   Maximum     Mark       Output

          drop       drop threshold threshold  probability  packets

  0          0          0        20       100     1/25        88946

  8          0          0        75       250     1/100           0
```

WRED is specifically targeted for TCP-based applications. When a dropped packet is detected, the T
sender initiates TCP's "slow start" mechanism. This enables the sender to gradually increase its
transmission rate. The random dropping of packets for TCP flows alleviates the possibility for globa
synchronization to occur. Unfortunately, WRED has little effect on applications that do not employ f
control mechanisms. Therefore, for classes composed primarily of UDP traffic, WRED has little effec

Summary

This chapter focused specifically on QoS performance and configuration for the MSFC and FlexWAN module. As demonstrated in the examples, QoS on the MSFC and FlexWAN is configured using the MQC, introduced in Chapter 5. The majority of the features available for the FlexWAN module and the MSFC are based on the QoS features supported on the 7500 with a VIP. In addition, the FlexWAN module and the MSFC extend the 6500's reachability to the MAN and WAN environments. Their integration allows for ease of configuration and management, as well as the capability to extend QoS support across the enterprise. This versatility ensures the appropriate service levels are maintained for mission-critical applications and protocols on an end-to-end basis. This chapter covered the following QoS concepts:

- Classification using dNBAR and NBAR protocol discovery.
- Marking using CAR, class-based policing, and the class-based marker.
- Policing and shaping functions on the FlexWAN module. The chapter demonstrated CAR and the class-based policing behavior, as well as DTS.
- Congestion management mechanisms available on the FlexWAN module. This discussion covered dWFQ, dCBWFQ, and dLLQ.
- Congestion avoidance mechanisms, particularly dWRED.

Chapter 10. End-to-End QoS Case Studies

Earlier chapters of this book focused on specific product lines of Catalyst switches and provided examples based on those product lines. This chapter reviews several common *quality of service* (QoS) features by applying these features to a sample campus network design scaled to six switches. The campus network design characterizes a common campus topology but represents the core, distribution, and access layer using only two switches at each layer. The campus network topology uses various families of Catalyst switches to form a multiplatform design. The campus network design and topology exaggerate the use of QoS features for the purpose of providing examples and leans toward a classification model of trusting access layer ports. Furthermore, some of the configurations in this chapter are similar to those generated by Auto-QoS. Because at time of publication Auto-QoS was not widely available on all platforms, Auto-QoS discussions are not included in this chapter.

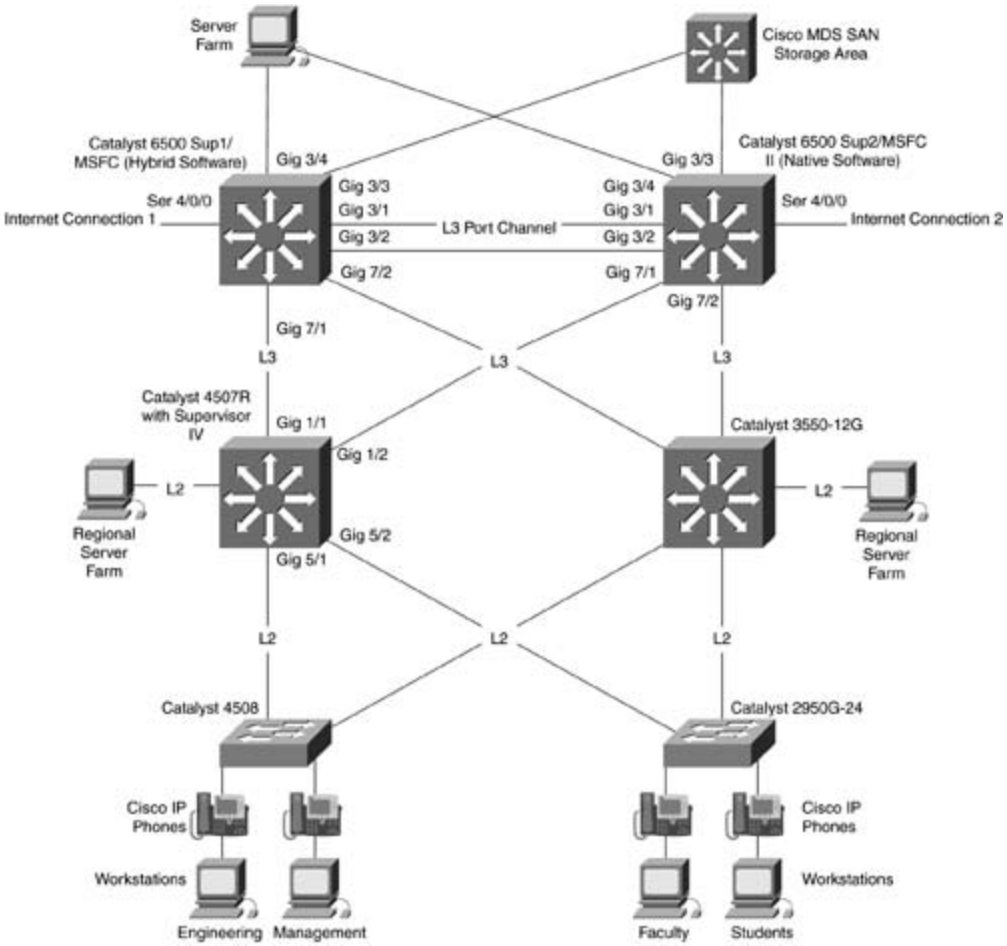
The sample network design includes the following QoS features:

- Input scheduling
- Classification based on trust configuration
- ACL-based classification
- Marking using policy maps
- Rate and markdown policing
- Individual policing
- Congestion management
- Congestion avoidance

This chapter applies these QoS features to the sample campus network topology in Figure 10-1 for review of an end-to-end QoS configuration. This chapter explains the network design and topology shown in Figure 10-1, describes each of the network layers and the QoS configuration, and summarizes the applications for common QoS features. At the conclusion of this chapter, you will understand how to apply common QoS features to a campus network design and topology.

Figure 10-1. Sample Multiplatform Campus Network Topology

[View full size image]


Server Farm
Cisco MDS SAN Storage Area
Catalyst 6500 Sup1/ MSFC (Hybrid Software)
Gig 3/4
Gig 3/3
Catalyst 6500 Sup2/MSFC II (Native Software)
Ser 4/0/0
Internet Connection 1
Gig 3/3
Gig 3/1
Gig 3/2
L3 Port Channel
Gig 3/4
Gig 3/1
Gig 3/2
Ser 4/0/0
Internet Connection 2
Gig 7/2
Gig 7/1
Gig 7/2
Gig 7/1
L3
L3
L3
Catalyst 4507R with Supervisor IV
Gig 1/1
Gig 1/2
Catalyst 3550-12G
L2
L2
Regional Server Farm
Gig 5/2
Gig 5/1
Regional Server Farm
L2
L2
L2
Catalyst 4508
Catalyst 2950G-24
Cisco IP Phones
Cisco IP Phones
Workstations
Workstations
Engineering
Management
Faculty
Students

Chapter Prerequisites and Material Presentation

This chapter assumes the reader has read through all the material in the previous chapters and grasps the basic understanding of all the QoS fundamentals presented in earlier chapters. This chapter presents material from an overview perspective. The chapter does not discuss command-line parameters or configuration specifics. Instead, this chapter presents a sample configuration and then discusses it. The reader should know how to relate the QoS configuration discussions to the sample configurations. The end goal of the chapter is to give the reader an understanding and overview of an end-to-end QoS design and topology.

Furthermore, this chapter refers to applications such as file sharing and databases. In many examples,*access-control lists* (ACLs) define these applications using TCP and UDP port assignments. The ACLs in the examples are not true representations of the necessary ACLs to profile the applications. Instead, the examples include the ACLs for completeness of the *command-line interface* (CLI) syntax.

The material presented in this chapter begins with the access layer and moves to the core. Because the classification begins in the access layer, the explanations begin there as well. To facilitate an easier understanding of the sample configuration presented, the configurations do not illustrate repeated identical interface configurations or infrastructure-related materials, such as spanning-tree parameters, IP routing protocols, and VLAN databases.

Multiplatform Campus Network Design and Topology

Figure 10-1 shows the campus network topology this chapter uses for sample configurations. The principle of the campus network design is to apply QoS features to support multiple *classes of service* among different classes of users and applications. To illustrate a sample campus network topology, the campus network design mimics a university campus consisting of multiple buildings. Each building uses two distribution layer switches to aggregate access layer switches that connect individual workstations and IP Phones in multiple wiring closets per floor of each building. The core aggregates each building by interconnecting the distribution layer switches. In addition to providing aggregation, the core also offers redundant Internet connectivity with firewall filtering, *virtual private networking* (VPN) services, and IP connectivity to iSCSI routers that interconnect a Cisco Multilayer Director and Fabric Switches (MDS) Fibre Channel *storage-area network* (SAN). A core server farm supplies global enterprise services such as e-mail, file storage, web services, and applications.

In the core, one of the Catalyst 6500 switches uses Native IOS Software; the other Catalyst 6500 switch uses Hybrid Software. Chapter 8, "QoS Support on the Catalyst 6500," discusses the differences between Native and Hybrid Software. The topology uses different software for illustration purposes; most campus designs keep the software versions consistent.

Catalyst 4507R and 3550-12G switches create the distribution layer. The distribution layer aggregates access layer switches in buildings throughout the campus network. These switches also connect regional server farms for the purpose of providing services on a per-building basis. Such services include web caching, authentication, and file services. The distribution layer uses Layer 3 connections to the core with Layer 2 connections to the access layer. As a result, the distribution layer switches route between local IP subnets. The core switches advertise default routes to the distribution layer so that the distribution layer can route to other IP subnets residing in other buildings. It is not necessary for the distribution layer to contain all the campus routes in its routing table.

Catalyst 2950 and 4506 switches comprise the wiring closet. These switches have redundant connections to the distribution layer via Layer 2 connections. The switches do not use the *Spanning Tree Protocol* (STP) for link redundancy; instead, these switches use *Hot Standby Router Protocol* (HSRP) for link redundancy. The design achieves link redundancy without the use of STP by designing each access layer switch with unique VLANs. Because there is no trunk between the distribution switches to carry the VLAN traffic between distribution switches, using unique VLANs per access layer results in these VLANs existing only on a single port per distribution switch. As a result, any respective VLAN interface moves to the down state during a link failure between the distribution and access layer. Consider the case when HSRP is in the active state for a VLAN on the Catalyst 4507R switch, and a link failure occurs between the Catalyst 4507R switch and the Catalyst 4506 switch. The VLAN interfaces associated with the link failure move to the down state because the only interface associated with the VLAN interface is no longer in the line protocol up state. Switches move VLAN interfaces to the down state when no ports in the VLAN are in the line protocol up state to prevent routing black holes. After the VLAN interface moves to the down state on the Catalyst 4507R switch, the HSRP peer on the Catalyst 3550-12G switch assumes full responsibility for routing traffic because it no longer hears HSRP hellos from the neighbor peer and therefore becomes the active HSRP router.

Although this design does not use STP for redundancy, STP is still utilized to prevent Layer 2 loops and broadcast storm that are caused by incorrect cabling, faulty hardware, or software defects. This design uses STP features such as Portfast and Trunkfast to expedite convergence; consult the Cisco website for more details on these features. An alternative to this wiring closet design is to use STP for redundancy. Although the STP method is currently the most popular, the current design trend is to use the method previously described.

For additional information on HSRP, refer to the following technical documents at Cisco.com:

"Using HSRP for Fault Tolerate IP Routing"

"HSRP Technical Tips"

Each access layer switch uses a least two VLANs, one for data and another for voice. The access layer connects various types of users in the campus. In this example, the campus network classifies traffic on access layer switches based on the following predefined types of users:

- Faculty
- Students
- Engineering
- Management

Access Layer Switches

The access layer switches connect individual workstations and IP Phones of various user types. In this example, the switches used for access layer switches are the Catalyst 2950G-24 and Catalyst 4506 switch. The next sections explore a sample QoS configuration for the Catalyst 2950G-24. Following the discussion of the QoS configuration for the Catalyst 2950G-24 is a discussion for the QoS configuration of the Catalyst 4506.

In brief, the next two sections discuss the following QoS features applied on access layer switches:

- Trust Cisco IP Phone
- Untagged packet classification
- Ingress port policing
- Congestion management
- Weighted round-robin
- Strict-priority queuing
- Egress CoS-to-transmit queue mapping

Catalyst 2950G-24

Example 10-1 illustrates the sample configuration applied to Figure 10-1.

Example 10-1. Catalyst 2950 Sample Configuration for Figure 10-1

```
Switch-2950#show running-config
Building configuration...
Current configuration : 2304 bytes
!
version 12.1
(text deleted)
!
!
wrr-queue bandwidth 10 40 30 20
```

```
wrr-queue cos-map 1 0 1
wrr-queue cos-map 2 2 3
wrr-queue cos-map 3 4
wrr-queue cos-map 4 5 6 7
!
class-map match-all Faculty_Profile
  match access-group 100
class-map match-all Student_Profile
  match access-group 101
!
!
policy-map STUDENT
  class Student_Profile
    police 4000000 16384 exceed-action drop
policy-map FACULTY
  class Faculty_Profile
    police 1000000 8192 exceed-action drop
    set ip dscp 0
!
(text deleted)
!
!
interface FastEthernet0/1
 switchport access vlan 2
 switchport voice vlan 700
 no ip address
 service-policy input FACULTY
 mls qos trust device cisco-phone
```

```
 spanning-tree portfast trunk
!
(text deleted)
!
interface FastEthernet0/13
 switchport access vlan 3
 no ip address
 service-policy input STUDENT
 mls qos cos 1
 mls qos cos override
 spanning-tree portfast
!
(text deleted)
!
interface GigabitEthernet0/1
 switchport mode trunk
 switchport trunk encap dot1q
 no ip address
 mls qos trust dscp
!
interface GigabitEthernet0/2
 switchport mode trunk
 switchport trunk encap dot1q
 no ip address
 mls qos trust dscp
!
(text deleted)
!
```

```
access-list 100 permit tcp any any eq 25000

access-list 100 permit udp any any eq 25000

access-list 101 permit ip any any

!

(text deleted)

!

end
```

For classification, the Catalyst 2950 switch uses two different classification models based on the front-panel port. The campus faculty and administrators use interfaces 1 through 12 for Cisco IP Phones and workstations. Ports 13 through 24 connect directly to student buildings, labs, and offices. Generally, network designs apply classification models on a per-interface basis. However, alternative methods of classification exist, such as using ACLs and policy maps where classification models utilize Layer 3 and 4 ingress packet information rather than the front panel.

The interfaces designated for faculty workstations and Cisco IP Phones use the trust DSCP based on the connected Cisco IP Phone method for classification. The switch trusts DSCP on ingress packets based on whether the interface learns that a Cisco IP Phone is connected. The switch learns of attached Cisco IP Phones via the *Cisco Discovery Protocol* (CDP). For switch interfaces that do not learn of an attached Cisco IP Phone, the switch classifies all ingress frames with a CoS value of zero. These switches map the classified CoS or *differentiated services codepoint* (DSCP) value to internal DSCP, which determines the policing, scheduling, and queuing behavior in packet processing.

Furthermore, the interface configuration consists of the default extended trust parameters. The default parameters inform the Cisco IP Phone to rewrite all incoming CoS values to zero for any ingress packets into the PC port of the phone. For more information regarding these classification techniques, refer to Chapter 2, "End-to-End QoS: Quality of Service at Layer 3 and Layer 2," and Chapter 3, "Overview of QoS Support on Catalyst Platforms and Exploring QoS on the Catalyst 2900XL, 3500XL, and Catalyst 4000 CatOS Family of Switches."

For interfaces designated for student use, the switch classifies ingress packets with the default port CoS of zero because of the default classification configuration of the Catalyst 2950G-24 switch. By default, the Catalyst 2950 Family of switches uses the untrusted configuration for all interfaces. The command `mls qos cos override` configures the switch to also classify 802.1q tagged frames with the default port CoS of zero. As a result, the interface classification state is untrusted for both 802.1q tagged and untagged frames.

To prevent the faculty workstations from consuming excessive bandwidth for file-sharing applications, the switch polices all traffic for file-sharing applications to 1 Mbps on ingress per port using the FACULTY policy map. The configuration uses ACL 100 to define file-sharing traffic. The policy map action reclassifies and marks all traffic matching the file-sharing class to a DSCP value of zero. This reclassification ensures the switch does not schedule or queue packets for file-sharing applications before any other higher-priority data traffic.

For student workstations, the STUDENT policy map limits the total traffic ingress per port to 4 Mbps. The policy map does not take into account any traffic profile; instead, the policy map restricts bandwidth for all traffic.

For congestion management, the switch uses *weighted round-robin* (WRR) with custom bandwidth configurations rather than strict-priority queuing. Because the Catalyst 2950 switch uses four transmit queues and the WRR parameters are global, each queue, 1 through 4, receives the following global bandwidth assignments: 10 percent, 40 percent, 30 percent, and 20 percent. In this manner, the lower-priority queues receive the lowest scheduling bandwidth while queues 2 and 3 receive higher bandwidth. Queue 4 is the highest-priority queue; however, most traffic in the network occupies lower queues and assigning a large WRR bandwidth value to queue 4 is unnecessary. Furthermore, the CoS-to-transmit queue mapping configuration places traffic for CoS 5 into queue 4. The default configuration is to place CoS 5 traffic into queue 3. Because voice traffic from Cisco IP Phones has a CoS value of 5 by default, placing voice traffic in queue 4 yields better performance because WRR services queue 4 before other queues.

Catalyst 4506

Example 10-2 shows the Catalyst 4506 sample configuration applied to Figure 10-1. In this example, the Catalyst 4506 is populated with a Supervisor III Engine.

Example 10-2. Catalyst 4506 Sample Configuration for Figure 10-1

```
Switch-4506#show running-config

Building configuration...

Current configuration : 4658 bytes

!

version 12.1

(text deleted)

!

qos

(text deleted)

!

!

class-map match-all Management_Profile

  match access-group 101

class-map match-all Engineering_Profile

  match access-group 100

!
```

```
!
policy-map MANAGEMENT
  class Management_Profile
    police 1 mbps 16000 byte conform-action transmit exceed-action drop
policy-map ENGINEERING
  class Engineering_Profile
    police 5 mbps 1600 byte conform-action transmit exceed-action drop
!
(text deleted)
!
interface GigabitEthernet1/1
 switchport mode trunk
 switchport encap dot1q
 qos trust dscp
 tx-queue 1
   bandwidth 400 mbps
 tx-queue 2
   bandwidth 200 mbps
 tx-queue 3
   bandwidth 200 mbps
   priority high
 tx-queue 4
   bandwidth 200 mbps
!
interface GigabitEthernet1/2
 switchport mode trunk
 switchport encap dot1q
qos trust dscp
```

```
 tx-queue 1
   bandwidth 400 mbps
 tx-queue 2
   bandwidth 200 mbps
 tx-queue 3
   bandwidth 200 mbps
   priority high
 tx-queue 4
   bandwidth 200 mbps
!
(text deleted)
!
interface FastEthernet6/1
 description Engineering
 switchport access vlan 4
 switchport voice vlan 701
 qos trust dscp
 service-policy input ENGINEERING
 tx-queue 3
   priority high
 spanning-tree portfast
!
(text deleted)
!
interface FastEthernet6/25
 switchport access vlan 5
 switchport voice vlan 701
 qos trust dscp
```

```
 service-policy input MANAGEMENT
 tx-queue 3
   priority high
 spanning-tree portfast
!
(text deleted)
!
access-list 100 permit tcp any any range 25000 30000
access-list 100 permit udp any any range 25000 30000
access-list 101 deny   ip any 10.0.0.0 0.255.255.255
access-list 101 permit ip any any
!
(text deleted)
!
end
```

As with the Catalyst 2950 switch, the Catalyst 4506 switch uses two different classification models based on the front-panel port. The campus engineering staff uses interfaces 1 through 24 for Cisco IP Phones and workstations. The campus management staff connects to interfaces 25 through 48.

The interfaces designated for either engineering or management employ the trust DSCP classification method. The switch just trusts the DSCP value of all ingress frames. As a result, the switch maps any packet's ingress DSCP directly to the internal DSCP value. Furthermore, the interface configuration consists of the default extended trust parameters. The default parameters inform the Cisco IP Phone to rewrite all incoming CoS values to zero for any ingress packets into the PC port of the phone.

For interfaces designed for use by the engineering staff, the switch polices all file-sharing and web-gaming traffic to 5 Mbps on ingress. The ENGINEERING policy map defines this behavior. The class map Engineering_Profile maps the ACL that defines the file-sharing and web-gaming traffic profile to the policy map.

For interfaces used by the management staff, the switch polices all Internet-bound traffic to 1 Mbps. The campus network connects to the Internet via two T3s(45 Mbps). Limiting the Internet-bound traffic for these ports to 1 Mbps is sufficient. The MANAGEMENT policy map defines this behavior. The class map Management_Profile maps the ACL that defines the file-sharing and web-gaming traffic profile to the policy map. The ACL defines Internet-bound traffic by noting which destination IP addresses are internal. All internal IP addressing in the campus network

resides on the 10.0.0.0/8 subnet.

For congestion management on the Gigabit Ethernet interfaces uplinked to the core, the switch uses WRR and Priority Queuing on a per-interface basis. The Catalyst 4506 switch uses four transmit queues per interface. For interfaces connecting the management and engineering workstations and IP Phones, the switch applies strict priority to queue 3 while implementing the default behavior of WRR sharing per queue. Each queue receives 250 Mbps as a share value by default. In conjunction with the strict-priority queuing, the switch transmits traffic out of queue 3 before servicing any other queue up until the share value of 250 Mbps. On the Catalyst 4000 Family of switches, queue 3 is the only queue available for strict-priority scheduling. By default, IP Phone voice traffic with CoS values 5 and DSCP values of 46 occupy queue 3 on egress. As a result, no mapping of voice traffic is necessary by default in this example.

For the uplinks connecting to the distribution layer, the congestion management differs slightly. Although queue 3 is a strict-priority queue, the queue only receives a share value of 200 Mbps. Queues 1, 2, and 4 receive 400 Mbps, 200 Mbps, and 200 Mbps as share values, respectively. As a result, egress scheduling is slightly different per queue than the default configuration. The principle behind the alternate scheduling configuration is that the higher-priority queues do not need more than 200 Mbps.

Distribution Layer

The distribution layer switches dual connect access layer switches via Gigabit Ethernet interfaces. The distribution layer switches used in this example are the Catalyst 4507R and the Catalyst 3550-12G switches. The next two sections explores a sample QoS configuration for both of these switche

In brief, the next two sections discuss the following QoS features applied on distribution layer switches:

- Trust DSCP classification
- Reclassification using policy maps
- WRR
- Strict-priority queuing

Catalyst 3550-12G

Example 10-3 illustrates the Catalyst 3550-12G sample configuration applied to Figure 10-1.

Example 10-3. Catalyst 3550-12G Sample Configuration for Figure 10-1

```
Switch-3550#show running-config
Building configuration...
Current configuration : 3460 bytes
!
version 12.1
(text deleted)
!
mls qos
!
class-map match-all VOIP
  match access-group 100
class-map match-all Database_iSCSI
  match access-group 102
```

```
class-map match-all Other
  match any
class-map match-all VIDEO
  match access-group 101
!
!
policy-map RECLASSIFY
  class VOIP
    trust dscp
  class VIDEO
    set ip dscp 34
  class Database_iSCSI
    set ip dscp 25
  class Other
    set ip dscp 0
!
(text deleted)
!
interface GigabitEthernet0/1
 no switchport
 ip address 10.2.2.1 255.255.255.252
 wrr-queue bandwidth 400 200 250 100
 wrr-queue cos-map 4 5
 priority-queue out
!
interface GigabitEthernet0/2
 no switchport
 ip address 10.2.2.5 255.255.255.252
```

```
 wrr-queue bandwidth 400 200 250 100
 wrr-queue cos-map 4 5
 priority-queue out
!
interface GigabitEthernet0/3
 switchport trunk encapsulation dot1q
 switchport trunk allowed vlan 2,3,700,1002-1005
 switchport mode trunk
 no ip address
 wrr-queue bandwidth 400 200 250 100
 wrr-queue cos-map 4 5
 service-policy input RECLASSIFY
 priority-queue out
!
interface GigabitEthernet0/4
 switchport trunk encapsulation dot1q
 switchport trunk allowed vlan 4,5,701,1002-1005
 switchport mode trunk
 no ip address
 wrr-queue bandwidth 400 200 250 100
 wrr-queue cos-map 4 5
 service-policy input RECLASSIFY
 priority-queue out
!
interface GigabitEthernet0/5
 switchport access vlan 201
  no ip address
 wrr-queue bandwidth 400 200 250 100
```

```
 wrr-queue cos-map 4 5
 service-policy input RECLASSIFY
 priority-queue out
(text deleted)
!
interface Vlan2
 ip address 10.0.2.3 255.255.255.0
 no ip redirects
 standby 2 ip 10.0.2.1
 standby 2 priority 100
 standby 2 preempt
!
(text deleted)
!
access-list 100 permit ip 10.200.0.0 0.0.255.255 10.200.0.0 0.0.255.255
access-list 101 permit tcp 10.0.0.0 0.255.255.255 10.0.0.0 0.255.255.255 range 250
  25999
access-list 101 permit udp 10.0.0.0 0.255.255.255 10.0.0.0 0.255.255.255 range 250
  25999
access-list 102 permit tcp 10.0.0.0 0.255.255.255 10.0.0.0 0.255.255.255 eq 3260
!
(text deleted)
!
end
```

In the access layer, the switches defined two levels of service. The switches classified traffic from II Phones as trusted, and the switch classified all other traffic with a CoS value of zero. The campus network design reclassifies traffic from the access layer and defines multiple levels of service.

The RECLASSIFY policy map reclassifies traffic into four categories: VOIP, VIDEO, Database_iSCSI

and Other. The *Voice over IP* (VoIP) traffic represents any traffic to and from Cisco IP Phones and their respective servers. These servers include the Cisco Call Manager and Unity servers. Because t access layer preserved the classification applied by the individual IP Phones, the RECLASSIFY polic map just trusts DSCP for any frames traversing the voice VLANs. ACL 100 defines the IP subnets th define the voice VLANs. This method actually provides multiple levels of services because the signaling and voice frames to and from IP Phones use different DSCP values.

The VIDEO class map defines traffic for video and voice applications. ACL 101 defines the applications by TCP and UDP ports numbers. This example uses arbitrary port numbers for brevity the configuration. The policy map classifies traffic for the VIDEO class map with a DSCP value of 34 This value is less than the Cisco IP Phone packet default DSCP value of 46.

The Database_iSCSI class map characterizes traffic for database applications and traffic that matches protocol ports used by the iSCSI protocol in ACL 102. As with any ACL in this chapter, ACL 102 is not a complete representation of an ACL that would define all database and iSCSI traffic. Th policy map classifies traffic for this class map with a DSCP value of 25. Finally, the policy map classifies all traffic that does not match any other profile to a DSCP value of 0. This configuration step is for illustration purpose because the ingress port is untrusted, traffic not matching any of the reclassification uses an internal DSCP value of 0, nonetheless.

For congestion management, the switch uses WRR with custom bandwidth configurations and a strict-priority queue designation on all interfaces connecting to other switches. The Catalyst 3550 Family of switches uses four transmit queues. In this example, each queue, 1 through 4, receives t following interface WRR bandwidth assignments: 400, 200, 250, and 100, respectively. The configuration of the WRR bandwidth is a relative value and not a bandwidth parameter in bits per second. Because the sum of the bandwidth assignments is 1000 and each interface is 1 Gbps, the values represent bandwidth weights that map directly to megabits per second. Queue 4 is the highest-priority queue; however, most traffic in the network resides in lower queues and configurir a large WRR bandwidth value to queue 4 is unnecessary. In addition, the CoS-to-transmit queue mapping places egress traffic for CoS 5 into queue 4 rather than queue 3, the default configuration because voice traffic uses CoS value 5 by default. Furthermore, queue 4 is a priority queue. Using the priority queue configuration forces the switch to service queue 4 before servicing any other queue.

Catalyst 4507R

Example 10-4 shows the Catalyst 4507R sample configuration applied to Figure 10-1.

Example 10-4. Catalyst 4507R Sample Configuration for Figure 10-1

```
Switch#show running-config

Building configuration...

Current configuration : 4241 bytes

!

version 12.1

(text deleted)
```

```
!
qos
(text deleted)
!
!
class-map match-all VOIP
  match access-group 100
class-map match-all Database_iSCSI
  match access-group 102
class-map match-all Other
  match any
class-map match-all VIDEO
  match access-group 101
!
!
policy-map RECLASSIFY
  class VOIP
    trust dscp
  class VIDEO
    set ip dscp 34
  class Database_iSCSI
    set ip dscp 25
  class Other
    set ip dscp 0
!
!
(text deleted)
!
```

```
interface GigabitEthernet1/1
 no switchport
 ip address 10.2.1.1 255.255.255.252
 qos trust dscp
 tx-queue 1
   bandwidth 400 mbps
 tx-queue 3
   bandwidth 200 mbps
   priority high
 tx-queue 4
   bandwidth 150 mbps
!
interface GigabitEthernet1/2
 no switchport
 ip address 10.2.1.5 255.255.255.252
 qos trust dscp
 tx-queue 1
   bandwidth 400 mbps
 tx-queue 3
   bandwidth 200 mbps
   priority high
 tx-queue 4
   bandwidth 150 mbps
!
interface GigabitEthernet 2/1
 switchport access vlan 201
 service-policy input RECLASSIFY
 tx-queue 3
```

```
   priority high
spanning-tree portfast
(text deleted)
!
interface GigabitEthernet5/1
 switchport trunk encapsulation dot1q
 switchport trunk allowed vlan 2,3,700
 switchport mode trunk
 service-policy input RECLASSIFY
 tx-queue 3
   priority high
!
interface GigabitEthernet5/2
 switchport trunk encapsulation dot1q
 switchport trunk allowed vlan 4,5,701
 switchport mode trunk
 service-policy input RECLASSIFY
 tx-queue 3
   priority high
!
(text deleted)
!
interface Vlan2
 ip address 10.0.2.2 255.255.255.0
 no ip redirects
 shutdown
 standby 2 ip 10.0.2.1
 standby 2 priority 150
```

```
 standby 2 preempt
!
(text deleted)
!
!
access-list 100 permit ip 10.200.0.0 0.0.255.255 10.200.0.0 0.0.255.255
access-list 101 permit tcp 10.0.0.0 0.255.255.255 10.0.0.0 0.255.255.255 range 250
  25999
access-list 101 permit udp 10.0.0.0 0.255.255.255 10.0.0.0 0.255.255.255 range 250
  25999
access-list 102 permit tcp 10.0.0.0 0.255.255.255 10.0.0.0 0.255.255.255 eq 3260
!
(text deleted)
!
end
```

As with the Catalyst 3550-12G in the distribution layer, the Catalyst 4507R reclassifies all ingress traffic from access layer switches. The fundamental QoS configurations between the Catalyst 3550-12G switch and the Catalyst 4507R switch are identical; however, there are a few configuration syntax and operational differences in congestion management between the two families of switches

With regard to congestion management, the WRR bandwidth parameters configure per transmit queue. The default bandwidth parameter is 250 Mbps. Furthermore, transmit queue 3, not queue 4 as with the Catalyst 3550 Family of switches, is designated as the priority queue. As a result, a CoS to-transmit queue configuration is not necessary because the default CoS-to-transmit queue mapping places voice traffic from Cisco IP Phones into queue 3. Other than these few syntax and operational differences, both the Catalyst 3550-12 and Catalyst 4507R behave identically using these distribution layer configurations.

Core Layer

The core layer switches dual connect distribution switches via Gigabit Ethernet interfaces. The swit
in this example for core layer switches is the Catalyst 6500. Each Catalyst 6500 operates using diff
software. One switch uses Hybrid Software; the other switch uses Native IOS. The next two section
explore a sample QoS configuration applied identically to both these switches.

In brief, the next two sections discuss the following QoS features applied on distribution layer swit

- Trust DSCP classification
- Reclassification using policy maps
- Policing to mark down traffic
- WRR
- Strict-priority queuing
- WRED congestion avoidance
- Transmit queue size manipulation

Catalyst 6500 Hybrid OS

Example 10-5 shows the Catalyst 6500 Hybrid OS sample configuration applied to Figure 10-1.

Example 10-5. Catalyst 6500 CatOS Sample Configuration for Figure 10-1

```
6k-Hybrid> (enable) show config

This command shows non-default configurations only.

Use 'show config all' to show both default and non-default configurations.

..............

(text deleted)

begin

!

# ***** NON-DEFAULT CONFIGURATION *****

!

!
```

```
(text deleted)
!
#qos
set qos enable
set qos wrr 1p2q2t 30 70
set qos txq-ratio 1p2q2t 60 20 20
set qos wred 1p2q2t tx queue 1 50:75 70:100
set qos wred 1p2q2t tx queue 2 60:80 75:100
set qos ipprec-dscp-map 0 8 16 26 34 46 48 56
set qos cos-dscp-map 0 8 16 26 34 46 48 56
set qos policed-dscp-map 0:0
set qos policed-dscp-map 1:1
set qos policed-dscp-map 2:2
set qos policed-dscp-map 3:3
set qos policed-dscp-map 4:4
set qos policed-dscp-map 5:5
set qos policed-dscp-map 6:6
set qos policed-dscp-map 7:7
set qos policed-dscp-map 8:8
set qos policed-dscp-map 9:9
set qos policed-dscp-map 10:10
set qos policed-dscp-map 11:11
set qos policed-dscp-map 12:12
set qos policed-dscp-map 13:13
set qos policed-dscp-map 14:14
set qos policed-dscp-map 15,25:0
set qos policed-dscp-map 16:16
set qos policed-dscp-map 17:17
```

```
set qos policed-dscp-map 18:18
set qos policed-dscp-map 19:19
set qos policed-dscp-map 20:20
set qos policed-dscp-map 21:21
set qos policed-dscp-map 22:22
set qos policed-dscp-map 23:23
set qos policed-dscp-map 24:24
set qos policed-dscp-map 26:26
set qos policed-dscp-map 27:27
set qos policed-dscp-map 28:28
set qos policed-dscp-map 29:29
set qos policed-dscp-map 30:30
set qos policed-dscp-map 31:31
set qos policed-dscp-map 32:32
set qos policed-dscp-map 33:33
set qos policed-dscp-map 34:34
set qos policed-dscp-map 35:35
set qos policed-dscp-map 36:36
set qos policed-dscp-map 37:37
set qos policed-dscp-map 38:38
set qos policed-dscp-map 39:39
set qos policed-dscp-map 40:40
set qos policed-dscp-map 41:41
set qos policed-dscp-map 42:42
set qos policed-dscp-map 43:43
set qos policed-dscp-map 44:44
set qos policed-dscp-map 45:45
set qos policed-dscp-map 46:46
```

```
set qos policed-dscp-map 47:47

set qos policed-dscp-map 48:48

set qos policed-dscp-map 49:49

set qos policed-dscp-map 50:50

set qos policed-dscp-map 51:51

set qos policed-dscp-map 52:52

set qos policed-dscp-map 53:53

set qos policed-dscp-map 54:54

set qos policed-dscp-map 55:55

set qos policed-dscp-map 56:56

set qos policed-dscp-map 57:57

set qos policed-dscp-map 58:58

set qos policed-dscp-map 59:59

set qos policed-dscp-map 60:60

set qos policed-dscp-map 61:61

set qos policed-dscp-map 62:62

set qos policed-dscp-map 63:63

set qos policer microflow iSCSI_mark_down rate 100000 burst 32000 policed-dscp

clear qos acl all

#VTC-Server

set qos acl ip VTC-Server trust-ipprec ip any  any

#iSCSI-Traffic_1

set qos acl ip iSCSI-Traffic_1 trust-dscp microflow iSCSI_mark_down tcp 10.0.0.0

  255.0.0.0 10.0.0.0 255.0.0.0 eq 3260

#iSCSI-Traffic_2

set qos acl ip iSCSI-Traffic_2 dscp 25 tcp 10.0.0.0 255.0.0.0 eq 3260 10.0.0.0 255

#

commit qos acl all
```

```
!
(text deleted)
!
#module 3 : 16-port 1000BaseX Ethernet
set module name    3
set vlan 100  3/1-2
set vlan 101  3/4
set vlan 102  3/5
set vlan 600  3/3
(text deleted)
set port qos 3/1-2,3/4-16 trust trust-dscp
set qos acl map iSCSI-Traffic_1 3/1
set qos acl map iSCSI-Traffic_1 3/2
set qos acl map iSCSI-Traffic_2 3/3
(text deleted)
set port channel 3/1-2 mode desirable silent
!
#module 4 : 0-port FlexWAN Module
!
#module 5 : 0-port Switch Fabric Module
!
(text deleted)
!
#module 7 : 16-port 1000BaseX Ethernet
set module name    7
set vlan 1071 7/1
set vlan 1072 7/2
(text deleted)
```

```
set port qos 7/1-16 trust trust-dscp

set qos acl map iSCSI-Traffic_1 7/1

set qos acl map iSCSI-Traffic_1 7/2

(text deleted)

!

end
```

The distribution layer classifies traffic into multiple levels depending on the protocol and applicatio not necessarily for the core switch to reclassify traffic from the distribution layer. As a result, all int that connect to other switches just classify traffic based on the trust DSCP classification mechanism core switches connect to other switches using interfaces on module three.

Furthermore, the global server farms consist of Cisco Call Manager and Unity servers. The servers global farm have the DSCP values, IP precedence, and CoS values administered securely and corre Using the trust DSCP classification mechanism is sufficient for these interfaces as well. The core sw use module seven for connecting the global servers farm.

The SAN, which connects to the core switches on module seven port one, does not mark traffic on e As a result, the core switches classify the traffic using an ACL. The goal of the ACL is to mark only t for iSCSI packet flows. As a result, the iSCSI-Traffic_2 policy ACL defines traffic from the iSCSI SA classification. The SAN network connects to port 3/3, and hence, the set qos acl map iSCSI-Traff 3/3 configuration command. The switch treats all other ingress traffic on this port as untrusted.

Furthermore, the core switches use an ingress microflow policer, iSCSI_mark_down, on all ports connecting to other switches to mark down iSCSI traffic exceeding 100 Mbps. This policer prevents building in the campus from overloading the egress transmit on the interface connecting to the SA network. The iSCSI protocol uses TCP as the transport; therefore, the out-of-order packets that ma for different scheduling of frames over the policing rate are not an issue. Because the distribution l marks iSCSI frames with a DSCP value of 25, the policed DSCP mapping table configuration forces switch to mark down frames above the policing with a DSCP value of 25 to 0.

In terms of congestion management and avoidance, all Catalyst 6500 switch ports in this design ut priority eqress queue and two standard eqress queues, each with two configurable *Weighted Rando Detection* (WRED)-drop thresholds (1p2q2t). Because of the priority queue designation, the switch all traffic in the priority before servicing the standard queues. The switches service the standard qu using WRR.

The configuration applies bandwidth values of 30 and 70 for each of the standard queues, respectiv priority queue does not use a bandwidth designation because the switch services that queue whene traffic exists in the queue regardless of WRR scheduling. The bandwidth values use weights betwee 255 for bandwidth assignment. In this example, queue 1 receives 30 percent of the bandwidth and 2 receives 70 percent of the bandwidth.

The switches assign the transmit queue size for the egress queues, 1 through 3, as 60 percent, 20 and 20 percent, respectively. Transmit queue 3 represents the strict-priority queue. Higher-priority generally do not have a large amount to transmit. Furthermore, queuing large amounts of high-pri traffic is generally unnecessary due to the time-sensitive nature of higher-priority traffic.

In terms of congestion avoidance, the Catalyst 6500s use WRED. Each transmit queue uses two thr

values. The threshold values are not the same for each transmit queue. For transmit queue 1, the f threshold uses a minimum value of 50 percent and a maximum value of 75 percent; the second thr uses 70 percent and 100 percent, respectively. In this manner, queue 1 has two thresholds at whic randomly and totally drop frames.

This example uses the default CoS-to-egress queue mapping. In addition, all other QoS mapping ta also use the default configuration.

Catalyst 6500 Hybrid MSFC

Example 10-6 illustrates the Catalyst 6500 Hybrid *Multilayer Switch Feature Card* (MSFC) sample configuration applied to Figure 10-1.

Example 10-6. Catalyst 6500 MSFC Sample Configuration for Figure 10-1

```
C6k-Hybrid-MSFC#show running-config

Building configuration...

Current configuration : 8949 bytes

!

version 12.1

!

(text deleted)

!

class-map match-any Internet-traffic

  match any

!

!

policy-map Shape-Internet

  class Internet-traffic

    shape average 45000000 180000 180000

!

(text deleted)

!
```

```
interface Serial4/0/0
 bandwidth 45000
 no ip address
service-policy output Shape-Internet
 dsu bandwidth 44210
 framing c-bit
 cablelength 10
!
(text deleted)
end
```

The campus network design uses redundant Internet connections. A FlexWAN module in the Cataly connects the Internet directly to T3s. To drop excess traffic transmitted on the interface, the FlexW a shaper. The policy map uses a class map that matches all traffic to shape the egress traffic out th interface to 45 Mbps. In this example, the shaper uses a committed burst rate of 180 kbps and an burst rate of 180 kpbs.

Catalyst 6500 Native IOS

Example 10-7 shows the Catalyst 6500 Native IOS sample configuration applied to Figure 10-1.

Example 10-7. Catalyst 6500 Native IOS Sample Configuration for Figure

```
C6k-Native#show running-config
Building configuration...
Current configuration : 11129 bytes
!
version 12.1
(text deleted)
!
!
```

```
class-map match-all iSCSI-Traffic_1
  match access-group 151
class-map match-all iSCSI-Traffic_2
  match access-group 152
!
!
policy-map Mark_DSCP_iSCSI
  class iSCSI-Traffic_2
    set ip dscp 25
policy-map iSCSI_mark_down
  class iSCSI-Traffic_1
     police 96000 32000 32000 conform-action transmit exceed-action policed-dscp-t
!
(text deleted)
!
mls qos map policed-dscp normal-burst 25 to 0
mls qos map ip-prec-dscp 0 8 16 26 34 46 48 56
mls qos map cos-dscp-map 0 8 16 26 34 46 48 56
mls qos
!
(text deleted)
!
interface Port-channel1
 description Port-Channel (G3/1 G3/2) to Core(Hybrid) 6500
 ip address 10.0.0.2 255.255.255.252
 mls qos trust dscp
 service-policy input iSCSI_mark_down
!
```

```
(text deleted)
!
interface GigabitEthernet3/1
 no ip address
 wrr-queue queue-limit 60 20
 wrr-queue random-detect min-threshold 1 50 70
 wrr-queue random-detect min-threshold 2 60 75
 wrr-queue random-detect max-threshold 1 75 100
 wrr-queue random-detect max-threshold 2 80 100
 mls qos trust dscp
 channel-group 1 mode desirable
!
interface GigabitEthernet3/2
 no ip address
 wrr-queue queue-limit 60 20
 wrr-queue random-detect min-threshold 1 50 70
 wrr-queue random-detect min-threshold 2 60 75
 wrr-queue random-detect max-threshold 1 75 100
 wrr-queue random-detect max-threshold 2 80 100
 mls qos trust dscp
 channel-group 1 mode desirable
!
interface GigabitEthernet3/3
 description Gigabit Connection to SAN
 switch access vlan 600
 wrr-queue queue-limit 60 20
 wrr-queue random-detect min-threshold 1 50 70
 wrr-queue random-detect min-threshold 2 60 75
```

```
 wrr-queue random-detect max-threshold 1 75 100
 wrr-queue random-detect max-threshold 2 80 100
 service-policy input Mark_DSCP_iSCSI
 mls qos trust dscp
!
(text deleted)
!
interface Serial4/0/0
 no ip address
 rate-limit output 45000000 8437500 16875000 conform-action transmit exceed-action
 shutdown
 dsu bandwidth 44210
 framing c-bit
 cablelength 10
!
(text deleted)
!
interface GigabitEthernet7/1
 ip address 10.2.1.6 255.255.255.252
 service-policy input iSCSI_mark_down
 mls qos trust dscp
!
interface GigabitEthernet7/2
 ip address 10.2.2.6 255.255.255.252
 service-policy input iSCSI_mark_down
 mls qos trust dscp
!
(text deleted)
```

```
!
access-list 101 permit ip any any
access-list 151 permit tcp 10.0.0.0 0.255.255.255 10.0.0.0 0.255.255.255 eq 3620
access-list 152 permit tcp 10.0.0.0 0.255.255.255 eq 3620 10.0.0.0 0.255.255.255
!
(text deleted)
!
end
```

Although the configuration syntax differs, the Native IOS QoS features used in Example 10-7 are id to the Hybrid CatOS and MSFC IOS configurations from Examples 10-5 and 10-6. The Native IOS configuration models the CLI from the Catalyst 3550 and Catalyst 4500 Family of switches. Becaus the Catalyst 6500s in Figure 10-1 use identical line modules, both switches operate identically in al aspects of QoS.

Summary

This chapter reviewed some of the common QoS features applied to a sample campus network topology. Many alternative configurations exist for campus network designs; however, this chapter evaluated the common features in application. Although not explicitly stated, this chapter highlighted the following QoS principles:

- As evident from the configurations, understanding packet flow and traffic profile is essential to building any Campus QoS design and topology.
- Classification on access layer switch interfaces is viable using policy maps with ACLs or trusting based on an attached Cisco IP Phone.
- Deploy reclassification and marking as needed throughout the campus to differentiate service in more levels than possible with access layer switches.
- Use policers to restrict unwanted traffic flows such as Internet gaming and file sharing.
- Deploying WAN interfaces on Catalyst switches eases configuration and provides for additional QoS features.
- Use a method of Priority Queuing when scheduling voice packets from transmit queues.
- Carefully administer QoS maps to maintain desired queuing and scheduling behavior.
- Use congestion avoidance techniques on any interface where congestion is common.

management:congestion:Auto-QoS

AVVID (Architecture for Voice, Video, and Integrated Data)

deployment:AVVID 2nd